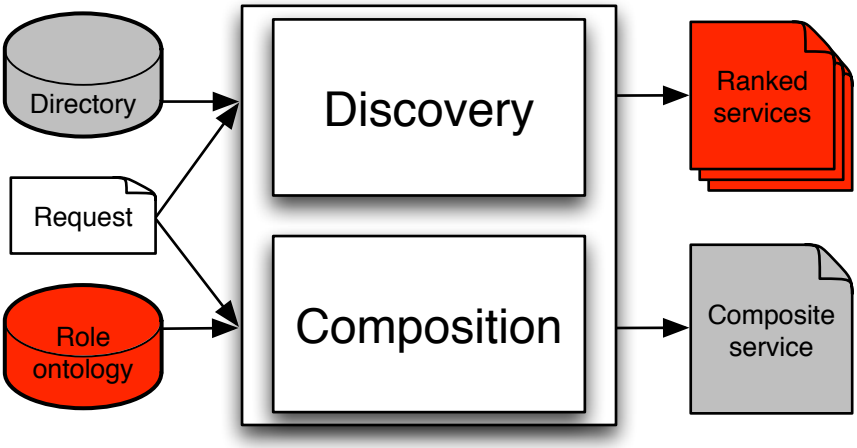
Alberto Fernández Gil

2

Descripción, Descubrimiento y Composición de Servicios en Entornos Multiagente Abiertos

Un Enfoque Organizacional

CETINIA



Publicaciones CETINIA
Universidad Rey Juan Carlos

Descripción, Descubrimiento y Composición de Servicios en Entornos Multiagente Abiertos. Un Enfoque Organizacional

Descripción, Descubrimiento y Composición de Servicios en Entornos Multiagente Abiertos. Un Enfoque Organizacional

Tesis Doctoral presentada por

Alberto Fernández Gil

para la obtención del título de Doctor en Informática

Departamento de Arquitectura y Tecnología de Computadores y Ciencias de la Computación e Inteligencia Artificial

Universidad Rey Juan Carlos

2007

A Lucía y Ángel

Agradecimientos

Sirvan estas líneas para agradecer a algunas personas cuya presencia de una u otra manera ha contribuido a que esta tesis sea realidad.

En primer lugar, a Sascha Ossowski, director de esta tesis doctoral, por su trabajo de dirección, ánimos y confianza desde que me incorporé a esta Universidad Rey Juan Carlos allá por el año 2000.

Quiero también hacer un agradecimiento especial a Matteo Vasirani, que ha compartido conmigo muchos momentos de esta tesis, y en especial por su labor de implementación de las técnicas propuestas.

Al resto de mis compañeros en la URJC. A Ana, Juanma y Holger, además de ser los que más tiempo llevamos juntos, también les agradezco su flexibilidad con los temas docentes, es un privilegio compartir asignaturas con vosotros durante tantos años. También a los demás compañeros César, David, Ingo, José Miguel, Oswaldo, Ramón, Roberto, Rubén (gracias también por cubrir parte de mi docencia últimamente) y Sergio. A todos les doy las gracias por su apoyo, ánimos y amistad.

A Eduardo Alonso por acogerme en la City durante mi estancia allí. A Axel Polleres y Juergen Dunkel por sus comentarios sobre la tesis.

Finalmente, a mis amigos y mis familiares, especialmente a Ángel, Lucía, Jose, Cristina, Miguel, Lucía e Irene. Espero poder dedicarles ahora algo más del poco tiempo que lo he hecho últimamente.

Resumen

En los sistemas multi-agente (SMA) abiertos, los agentes se suelen concebir como entidades software capaces de anunciar los servicios que proporcionan, localizar otros proveedores que ofrecen servicios potencialmente interesantes para ellos, y negociar acuerdos respecto a la ejecución de servicios. En los SMA orientados a servicios, los agentes no son simplemente encapsuladores de la interfaz de los servicios que proveen sino que, además de ejecutar un servicio, deberían ser capaces de entablar diferentes tipos de interacciones sociales durante la provisión del servicio.

Aunque los modelos organizacionales son frecuentes en las metodologías de diseño orientado a agentes, aún no han sido utilizados en los mecanismos de descripción de servicios de agentes y, en consecuencia, no son explotados explícitamente por las técnicas actuales para el descubrimiento y composición de servicios. En este trabajo se propone un enfoque que usa conceptos organizacionales, como los roles e interacciones, para extender las tendencias actuales en la descripción semántica, equiparación y composición de servicios en sistemas multi-agente.

Los agentes utilizan un lenguaje de **descripción de servicios** para especificar las características relevantes de los servicios que proveen. La mayoría de los enfoques actuales basan sus descripciones de servicios en sus entradas y salidas; algunos también utilizan sus precondiciones y efectos, u otros parámetros adicionales como la categoría del servicio. En este trabajo se propone un enfoque que, además de esta información, explota los conceptos organizacionales para caracterizar los contextos en los que pueden usarse los servicios. Para ello se utiliza una ontología de roles e interacciones.

En esta tesis se ha desarrollado un algoritmo de **equiparación**[1] que toma como entrada una solicitud y un anuncio de servicio, y devuelve el grado de ajuste entre ambos, un número real entre 0 y 1 mediante el cual los proveedores de servicios pueden ser ordenados para ser seleccionados (quizá junto con otros criterios). El encaje semántico entre dos roles se calcula en base a una ontología de roles, y es una función que depende de la relación de subsunción entre ellos y la distancia entre ellos en la taxonomía. El grado de encaje entre dos servicios combina los valores de encajes entre roles, que son los elementos básicos de las descripciones de servicios en este enfoque.

Si ninguno de los servicios disponibles encaja con uno concreto buscado, se puede

[1] ajuste, encaje o *matching*

utilizar una funcionalidad de **planificación** para construir servicios compuestos. En SMA abiertos de gran escala orientados a servicios, un enfoque de planificación con técnicas puras de Inteligencia Artificial puede ser impracticable debido al enorme número de servicios (operadores) que normalmente están registrados en el directorio. En esta tesis se propone un método genérico de filtrado de aquellos servicios que probablemente sean irrelevantes al proceso de planificación. Nuestra heurística se basa en la dimensión de los planes y en el número de ocurrencias de servicios en planes. Se puede aproximar esta información almacenando y procesado los planes creados en el pasado. Sin embargo, el número de servicios y posibles consultas puede ser muy grande y la repetición continua de una solicitud de un servicio concreto es bastante improbable. Para superar este inconveniente, agrupamos servicios en clases en base a ciertas propiedades. En particular, obtenemos información sobre las clases de los servicios a partir de las taxonomías de roles e interacciones que se derivan del modelo organizacional del SMA.

Los métodos propuestos en esta tesis han sido instrumentados en componentes software que han sido validados mediante pruebas empíricas que han permitido comprobar la validez de nuestro enfoque. Además, estos componentes han sido integrados como parte la arquitectura del proyecto europeo CASCOM[2]. Este proyecto consiste en el desarrollo, implementación y validación de una infraestructura para coordinación de servicios basada en agentes para el descubrimiento, composición y ejecución de servicios Web semánticos en redes de servicios peer-to-peer fijas y móviles. En particular, como componentes de la infraestructura, forman parte de una aplicación en el dominio de asistencia en emergencias médicas que se ha evaluado en el Tyrolean Hospital Consortium en Austria.

[2]http://www.ist-cascom.org/

Abstract

In open multi-agent systems (MAS), agents are often conceived as software entities capable of advertizing the services they provide, locating other service providers that offer services potentially of interest to them, and negotiating agreements regarding service enactment. In service-oriented MAS, agents need to be more than mere wrappers of the service interface: besides executing a service, they should be able to engage in different types of social interactions when providing services.

Although organizational models are usually present in agent-oriented design methodologies, they have not yet found their way into service description mechanisms for agents and consequently are not explicitly exploited by today's service discovery and composition mechanisms. We propose an approach that bridges this gap by making use of organizational concepts such as roles and interactions to extend current trends in semantic service description, matchmaking and composition in MAS.

Agents use a service description language to specify relevant characteristics of the service(s) they provide. Most current approaches base their service descriptions on inputs, outputs, preconditions and effects. Sometimes additional parameters, like service category, are taken into account. In this work we propose an approach that, in addition to this information, exploits organizational concepts to further characterize the contexts in which services can be used.

We have developed a role-based matching algorithm that takes as input a service request and a service advertisement, and returns the degree of match between them. This is a real number between 0 and 1 by which service providers can be ranked for selection. The semantic match of two roles is made based on the ontology of roles. It is a function that depends on the subsumption relation between the two roles in the ontology, and the distance between them in the taxonomy. The semantic match between two services combines the semantic match between two roles, which are the basic elements in service descriptions.

If no adequate services are available for a specific request, a planning functionality can be used to build up composite services. In open large-scale service-oriented MAS, a pure AI planning approach can become impracticable due to the vast number of services (operators) that are usually registered in the directory. In this thesis we propose a generic method for filtering out those services that are probably irrelevant to the planning process. Our heuristic is based on the plan dimension and on the number of occurrences of services in plans: the more important a service is, the greater

the number of plans for which it is necessary and the shorter the plans for which it is required. We can approximate this information by storing and processing the plans historically created. However, the number of services and possible queries may become too large and the continual repetition of one particular service request is rather unlikely. To overcome this drawback, we cluster services into classes based on certain properties. In particular, we obtain service class information from the role and interaction taxonomies that are derived from the MAS organizational model.

The methods proposed in this thesis have been implemented and evaluated experimentally through lab tests. In addition, the approach presented here has been integrated as part of the European IST project CASCOM[3] infrastructure. This project has developed, implemented and validated an agent-based service coordination infrastructure for the discovery, composition and execution of innovative Semantic Web services across mobile and fixed peer-to-peer service networks. In particular, as a component of the infrastructure, it is part of a field trial in medical emergency assistance that has been tested at the Tyrolean Hospital Consortium in Austria.

[3] http://www.ist-cascom.org/

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

La Computación Orientada a Servicios (SOC) [59, 60] es un nuevo paradigma de computación que utiliza los servicios como los constructores básicos para el desarrollo rápido, a bajo coste y de fácil composición de aplicaciones distribuidas incluso en entornos heterogéneos. Los servicios son entidades computacionales que pueden ser descritos, publicados, descubiertos y ensamblados dinámicamente para el desarrollo de sistemas masivamente distribuidos, interoperables y evolutivos. Los servicios llevan a cabo funciones que pueden ir desde responder preguntas sencillas hasta ejecutar sofisticados procesos de negocio que requieren relaciones peer-to-peer posiblemente entre múltiples niveles de consumidores y proveedores. El enfoque de programación orientado a servicios se basa en la idea de componer aplicaciones mediante el descubrimiento e invocación de servicios disponibles a través de la red en lugar de construir nuevas aplicaciones o mediante invocación de aplicaciones para llevar a cabo algunas tareas [93]. Normalmente los servicios se construyen de forma que sean independientes del contexto en el que son usados, por lo que hay un acoplamiento débil entre el proveedor y el consumidor.

Este enfoque "orientado a servicios" es independiente de lenguajes de programación y sistemas operativos específicos. Permite a las organizaciones exponer sus competencias por medio de programas en Internet o una variedad de redes

(p.e. cable, UMTS, Bluetooth, etc.), usando lenguajes y protocolos estándares (basados en XML), e implementando una interfaz autodescriptiva.

La promesa visionaria de la Computación Orientada a Servicios es un mundo de servicios que cooperan donde los componentes de las aplicaciones se ensamblan con poco esfuerzo en una red de servicios que puede ser débilmente acoplada para crear procesos de negocio flexibles y dinámicos, y aplicaciones ágiles que pueden abarcar organizaciones y plataformas de computación [72]. Los servicios mantienen la promesa de moverse más allá del simple intercambio de información (el mecanismo dominante actualmente para la integración de aplicaciones) al concepto de acceso, programación e integración de servicios que son encapsulados en viejas y nuevas aplicaciones. Un importante beneficio económico del paradigma de la SOC es que permite a los desarrolladores construir aplicaciones más rápidamente que nunca antes, creando aplicaciones compuestas que usan recursos software existentes internos a la organización en combinación con componentes externos posiblemente residentes en redes remotas. Las aplicaciones aisladas disponibles dentro de una organización se pueden convertir ahora en arquitecturas basadas en servicios y ser integradas más efectivamente que cuando dependían de tecnologías de integración punto a punto. El resultado final es que es más fácil de crear nuevas aplicaciones compuestas que usan piezas de aplicación y/o datos que residen en los sistemas existentes. Esto representa un cambio fundamental en la estructura socioeconómica de la comunidad de desarrollo de software que mejora la efectividad y productividad en las actividades de desarrollo de software y permite a las empresas poner nuevos productos y servicios en el mercado más rápidamente [71].

Clave para este concepto es la Arquitectura Orientada a Servicios (SOA) [60]. La SOA es una forma lógica de diseñar un sistema software para proporcionar servicios tanto para aplicaciones finales como para otros servicios distribuidos en una red, vía interfaces publicados y que se pueden encontrar. Una arquitectura orientada a servicios bien construida y basada en estándares puede dotar a un entorno con una infraestructura y entorno de proceso flexible. La SOA consigue esto proporcionando procesos y funciones independientes, reutiliza-

bles y automáticos como servicios. La eficiencia en el diseño, implementación y operación de sistemas basados en SOA pueden permitir a las organizaciones adaptarse mucho más rápidamente a un entorno cambiante.

Los Servicios Web es la tecnología actual basada en el concepto de SOC más prometedora [122]. Los servicios Web proporcionan la base para el desarrollo de procesos de negocio que están distribuidos por la red y disponibles vía interfaces y protocolos estándar. Pueden usar Internet como medio de comunicación (así como otros protocolos de transporte) y estándares abiertos basados en Internet, como SOAP (Simple Object Access Protocol) [121] como medio de transmisión, WSDL (Web Services Description Language) [25] para la descripción del servicio y BPEL4WS (Business Process Execution Language for Web Services) [15] para la orquestación de servicios.

Los servicios Web aportan un primer paso importante a la funcionalidad a la actual Web hacia una integración sin fisuras de componentes software distribuidos utilizando estándares de la Web. La tecnología actual de servicios Web permite el intercambio de mensajes entre servicios Web (SOAP), la descripción de la interfaz para consumir un servicio Web (WSDL), y soporta el anuncio de servicios Web en registros (UDDI [8]). Sin embargo, estas tecnologías no incluyen descripciones explícitas de la funcionalidad de un servicio Web. Además, estas tecnologías operan a un nivel sintáctico y por tanto no representan el significado de la información a intercambiar, requiriendo en gran medida de la intervención humana para la programación de aplicaciones que utilicen los servicios Web.

El desarrollo de la Web Semántica [10] está permitiendo dotar a los servicios Web de información semántica lo cual facilita la automatización de su descubrimiento, composición e invocación. La Web Semántica va a dirigida a la mejora del entendimiento automático del contenido Web, en donde las ontologías se han identificado como un aspecto clave. Varios lenguajes para representar ontologías se construyen sobre XML, como RDF [97], RDF Schema [98] y OWL [90]. La Web Semántica y los Servicios Web están siendo las tecnologías claves para la siguiente generación de las aplicaciones Web.

Los Servicios Web Semánticos (SWS) aplican tecnologías de la Web Semántica a los servicios Web mediante el uso de descripciones semánticas que permitan la provisión de mecanismos inteligentes para el descubrimiento, composición, contratación, y ejecución de servicios Web. Se suele utilizar OWL-S [77] como lenguaje de descripción de servicios. OWL-S es una ontología escrita en OWL cuyo objetivo es hacer que los servicios Web sean interpretables por un programa, es decir, descritos con suficiente información para permitir, de forma automatizada, el descubrimiento, invocación, composición y monitorización de la ejecución de servicios.

Desde hace dos décadas, los Agentes Software y Sistemas Multi-Agente [123] han crecido como un área de investigación muy activa. Esta tecnología tiene cierta similitud y se ha dedicado en parte a aspectos similares a los que tiene como objetivo la computación orientada a servicios. A pesar de la aparente similitud entre ambos enfoques, existes aspectos importantes que los diferencian. Un agente es un sistema que está situado en un entorno y que es capaz de actuar de forma flexible y autónoma en ese entorno para conseguir unos objetivos [126]. Las principales características del concepto agente son [128]: autonomía (operan sin la intervención directa de humanos o de otros agentes), reactividad (son capaces de percibir estímulos de su entorno y reaccionar en un instante pequeño de tiempo a los cambios que se produzcan en él), proactividad (son capaces de exhibir un comportamiento dirigido por objetivos tomando iniciativa) y sociabilidad (son capaces de comunicarse con otros agentes). Por el contrario, un servicio Web típicamente realiza procedimientos computacionales más bien sencillos, y tiene un comportamiento pasivo y predecible. Por tanto, los agentes proporcionan un nivel de abstracción mayor que los objetos o los servicios Web.

La tecnología de agentes proporciona a los diseñadores un enfoque orientado a las interacciones para diseñar sistemas software abiertos [76]. Existe una consciencia generalizada de que los modelos organizacionales son fundamentales para regular los sistemas multi-agente abiertos para inculcar las propiedades deseadas. Así lo demuestra el hecho de que prácticamente todas las metodologías de diseño de SMA están centradas o hacen uso de este tipo de modelos

(por ejemplo [107, 129, 133]).

La relación entre los agentes y los servicios existe desde diferentes perspectivas [78]:

1. Los agentes usan servicios. En esta vista, los servicios individuales pueden permanecer relativamente simples, son proveedores de capacidades discretas accedidas a través del intercambio de mensajes, sin exhibir pro-actividad, autonomía u otros atributos sofisticados de los agentes. Es más, en esta vista, algunos de los desafíos más fuertes asociados con los servicios Web, como la composición automática de servicios, podría relegarse finalmente a la competencia de los agentes, y dejarlo fuera del ámbito de los estándares de los servicios Web. En general, esta vista no impone requerimientos a la infraestructura que soporta los servicios más que los atributos de los agentes.

2. Los servicios son agentes, aunque actualmente de un tipo limitado. En esta vista, que es la más ambiciosa respecto al futuro de la tecnología de servicios Web, los servicios individuales finalmente serán libres de mostrar la autonomía, proactividad, racionalidad, etc. que define la noción de agente, y colecciones de servicios interactuarán con la flexible colaboración que define la esencia de los SMA. Actualmente, sin embargo, la mayoría del trabajo en servicios no encaja en esta visión. En particular, los servicios individuales se conciben mayormente como reactivos, efímeros, y destinados a entablar sólo dos partes en un estilo de uso proveedor/solicitante. Aunque algunos trabajos esperan romper estas limitaciones, todavía hay un largo camino por recorrer.

3. Los agentes se componen de, se implantan como, y se extienden dinámicamente por servicios. Esta vista, que mantiene que los agentes se construyen con servicios Web como bloques de construcción [17], se puede ver como un enfoque intermedio entre (1) y (2). Permite una noción de servicio más limitada que en (2), pero que existe en un marco conceptual más extenso del requerido en (1).

Igual que ocurre en muchos ámbitos de las sociedades humanas, los agentes intermediarios [31] juegan un papel importante en las sociedades de agentes artificiales. Estos agentes proporcionan medios para facilitar la coordinación entre agentes en entornos abiertos, mediante servicios de localización y comunicación entre agentes que proporcionan servicios y agentes que los solicitan. En general, en este contexto se pueden distinguir tres categorías de agentes: proveedores de servicios, solicitantes de servicios y agentes intermediarios. El proceso básico de mediación es el siguiente:

1. Los agentes proveedores de servicios anuncian sus capacidades (servicios capaces de llevar a cabo) a los agentes intermediarios.
2. Los agentes intermediarios almacenan estos anuncios.
3. Un agente solicitante pide a algún intermediario la localización y comunicación con agentes que proporcionen determinado servicio. Esta intermediación puede incluir una transacción completa y otros servicios añadidos.
4. El agente intermediario procesa la petición, analizando las capacidades de los agentes registrados y devuelve el resultado. El resultado puede ser un subconjunto de anuncios junto con los identificadores de los proveedores correspondientes para contactar, o el resultado de la transacción completa utilizando el servicio más adecuado.

La idea en principio es simple, pero es complicado llevarla a cabo en sistemas abiertos en los que coexisten agentes heterogéneos, desarrollados por diferentes entidades y donde los cambios son frecuentes. Por tanto, la funcionalidad esencial de cualquier clase de agente intermediario es facilitar la interoperabilidad de servicios, agentes y sistemas, de una forma flexible, fiable y segura y respetando los marcos legales existentes.

Actualmente el problema de la descripción y descubrimiento de agentes que proporcionan un determinado servicio se suele llevar a cabo utilizando técnicas de los servicios Web semánticos [73, 92, 115], y principalmente se basan en la

búsqueda a través de las entradas y salidas de los servicios, y en algunos casos también de las precondiciones, efectos, y, en menor medida, otros parámetros que describen los servicios. Sin embargo, los mecanismos de descubrimiento de servicios en sistemas basados en agentes no sacan provecho de los modelos organizacionales subyacentes al sistema multi-agente. Este aspecto se trata en esta tesis doctoral.

Cuando no existe ningún servicio que se adecue a los requisitos buscados por el solicitante, es posible construir servicios compuestos combinando varios servicios existentes. Para hacer frente a este problema, un agente intermediario necesita estar dotado de capacidades de planificación que reciban descripciones de servicios como entrada y coordine varios de ellos de la forma adecuada. El problema de composición de servicios tiene algunas diferencias con respecto a los problemas de planificación clásicos de IA, como que los planes de los servicios compuestos no deben ser muy grandes pero, en cambio, pueden ser construidos a partir de un número muy grande de servicios (operadores) que están normalmente registrados en un directorio. Esto es particularmente el caso de los sistemas multi-agentes abiertos orientados a servicios, donde un enfoque de planificación puro basado en técnicas de IA puede ser impracticable. Actualmente todavía existen pocos enfoques sobre planificación de servicios compuestos, en especial basados en OWL-S ([54, 109, 116, 130]). Además, ninguno de ellos utiliza la información organizacional subyacente al modelo de agentes. En esta tesis se aborda este problema.

1.1. Objetivos

El objetivo principal perseguido en esta tesis doctoral es contribuir al campo de la descripción, descubrimiento y composición de servicios en entornos multiagente abiertos, que permitan explotar los modelos organizacionales normalmente presentes en las metodologías de construcción de sistemas basados en agentes, con el objetivo de mejorar tanto la eficiencia como la usabilidad de las arquitecturas orientadas a servicios basadas en agentes.

Este objetivo principal se puede descomponer en los siguientes objetivos específicos:

1. Proponer una extensión a los actuales enfoques para la descripción de servicios, que permita recoger también los conceptos organizacionales comunes, como roles sociales y tipos de interacciones para caracterizar mejor el contexto en el que pueden ser usados ciertos servicios semánticos. En primer lugar se adoptará y/o modificará algún metamodelo basado en roles y centrado en las interacciones. Posteriormente se realizará una propuesta para describir servicios mediante la información proporcionada por dichos modelos. La propuesta será compatible con las actuales tendencias y estándares (OWL-S).

2. Diseñar un algoritmo de equiparación de servicios (matching) para la descripción de servicios propuesta en el objetivo anterior. El algoritmo de equiparación permitirá calcular el grado de similitud entre un anuncio de servicio y una especificación sobre un servicio que se está buscando. El mecanismo se propondrá como una extensión a las técnicas actuales que se enfocan en otros aspectos de la descripción de los servicios (entradas, salidas, precondiciones, efectos, ...). Se estudiará la eficiencia y efectividad del método propuesto, así como su integración en algún matchmaker de propósito general evaluando la mejora aportada por este nuevo enfoque.

3. Proponer un método que reduzca la complejidad del problema de planificación de un servicio compuesto que debe afrontarse en sistemas abiertos. Debido a que la complejidad de las técnicas de planificación dependen en gran medida del conjunto de operadores (en este caso servicios), se investigará la forma de reducir el conjunto de servicios de entrada al planificador estimando la relevancia que un servicio puede tener de formar parte de un hipotético plan para un servicio compuesto deseado. En particular, se utilizará la información organizacional incluida en las descripciones de los servicios para filtrar de forma heurística aquellos servicios que probablemente sean irrelevantes al proceso de planificación.

4. Evaluar la utilidad de los enfoques propuestos mediante su aplicación práctica en problemas del mundo real.

1.2. Terminología utilizada

Esta sección tiene como objetivo aclarar posibles interpretaciones sobre algunos términos utilizados en esta memoria, ya que en ciertos casos son tratados de forma diferente por distintos autores. Además, algunos de ellos no tienen una traducción clara al castellano o simplemente no hay un acuerdo claro sobre su traducción exacta.

Algunos autores distinguen los conceptos de descubrimiento (*discovery*) y equiparación (*matching o matchmaking*). Según esa distinción, el descubrimiento se refiere a la búsqueda de descripciones de servicios en los directorios, usando palabras claves, categorías o algún otro tipo de información muy básica. Tiene sentido sobre todo cuando el directorio está distribuido y hay que buscar posibles servicios candidatos. Cuando se realiza un análisis más exhaustivo (semántico) se denomina equiparación (*matching*). En este trabajo, salvo referencias muy concretas, nos centramos en la equiparación semántica de descripciones de servicios, sin preocuparnos de cómo se han conseguido. Por tanto, consideraremos sinónimos los términos *descubrimiento, búsqueda, equiparación, encaje, ajuste.*

Otro tema de controversia suele ser qué se considera *servicio*, *servicio Web*, *descripción de servicio*, etc. Cuando hablamos de *servicio* nos referimos a un componente computacional. Generalmente tratamos con *descripciones de servicios*, aunque en algunas ocasiones a tales descripciones se refiere simplemente como *servicios*. No obstante el contexto en el que aparecen estos términos deja claro a lo que se refiere.

Algunos otros términos usados ampliamente en la literatura y que suelen tratarse como sinónimos son los siguientes:

- directorios, registros de servicios, páginas amarillas

- servicio compuesto, plan
- solicitud (de servicio), consulta, request, query

1.3. Organización de la Memoria

La presente memoria está organizada de la siguiente manera.

El capítulo 2 revisa el estado actual en los principales campos de investigación en los que se enmarca esta tesis:

- los Sistemas Multi-Agente, donde se revisan las principales metodologías para la construcción de este tipo de sistemas software, así como los estándares y herramientas existentes
- la Computación Orientada a Servicios, donde se recorren los servicios Web, la Web semántica y los servicios Web semánticos, describiendo los principales estándares y técnicas utilizadas para la descripción, descubrimiento y composición de servicios

Los siguientes cuatro capítulos detallan las cuatro aportaciones principales de esta tesis, que se corresponden con los cuatro objetivos propuestos.

En el capítulo 3, de acuerdo con el primer objetivo planteado, se propone la extensión de los enfoques actuales para la descripción de servicios incluyendo conceptos organizacionales subyacentes a los sistemas multi-agente.

El segundo objetivo se trata en el capítulo 4, donde se describe un algoritmo de equiparación de servicios basándose en la información organizacional incluida en las descripciones de los servicios (capítulo 3). Se incluye una evaluación empírica que permite comprobar el rendimiento del método propuesto.

El capítulo 5 aborda el tercero de los objetivos planteados, proponiendo un método genérico para el filtrado de los servicios que se estimen menos relevantes para el proceso de planificación. Además se proponen formas de instanciar

dicho filtro utilizando de nuevo la información organizacional descrita, así como las categorías de los servicios.

El capítulo 6 se dedica a la aplicación práctica de las propuestas desarrolladas en los capítulos anteriores (objetivo cuarto). Por un lado, se muestra un ejemplo de aplicación de la propuesta de descripción de servicios en un sistema de ayuda a la decisión para la gestión de transporte. Por otra parte, se describe la utilización de los componentes software desarrollados como parte de una plataforma para la coordinación de servicios Web semánticos que ha sido validada en una aplicación real para la asistencia en casos de emergencias médicas.

Finalmente, en el capítulo 7, se describen las principales conclusiones obtenidas como resultado de este trabajo y las publicaciones que han surgido durante su desarrollo, así como algunas de las líneas de investigación futuras que quedan abiertas como consecuencia de este trabajo.

El apéndice A incluye una lista de abreviaturas utilizadas en la memoria. Además, una versión en inglés de las conclusiones se encuentra en el apéndice B.

Capítulo 2

Antecedentes

En este capítulo se analiza el estado actual de los dos principales campos de investigación con los que se relaciona esta tesis.

En primer lugar se revisa el campo de los sistemas multi-agente, describiendo en qué consisten, las principales metodologías para la construcción de sistemas software siguiendo este paradigma, así como los estándares y herramientas existentes.

En la segunda parte se analiza la Computación Orientada a Servicios. Se describen los servicios Web, aplicación más representativa actualmente de este paradigma, con sus principales estándares. A continuación se presenta cómo la Web Semántica pretende describir el contenido de la Web actual y, mediante la utilización de esas técnicas, la evolución hacia los servicios Web semánticos. Se describen las técnicas actuales más importantes para la descripción y descubrimiento de servicios, así como la composición de servicios.

2.1. Sistemas Multi-Agente

La evolución de las tecnologías de la información y comunicaciones está provocando la proliferación de sistemas distribuidos conectados a través de una red

(por ejemplo, Internet). Sistemas que se comunican con otros a través de una red de comunicaciones de forma que pueden intercambiar información. Para que sea posible tal comunicación es necesario que los diferentes sistemas traten la información intercambiada con un formato conocido. De ahí que hayan surgido diferentes estándares de comunicación, sobre todo de bajo nivel (hasta el nivel de transporte) y objetos distribuidos.

Pero muchas aplicaciones deben realizar razonamientos cada vez más complejos, se construyen distribuyendo sus componentes o deben negociar con otros sistemas (por ejemplo para conseguir un producto al mejor precio del mercado). La tecnología de agentes permite afrontar estos sistemas de forma más sencilla que con otras técnicas tradicionales de ingeniería del software.

Un agente es un sistema que está situado en un entorno y que es capaz de actuar de forma flexible y autónoma en ese entorno para conseguir unos objetivos [126]. Las principales características del concepto agente son [128]:

- *autonomía*: los agentes operan sin la intervención directa de humanos o de otros agentes, y tienen alguna clase de control sobre sus acciones y estado interno
- *reactividad*: los agentes son capaces de percibir estímulos de su entorno (que puede ser un mundo físico, un usuario a través de una interfaz gráfica, internet,...) y reaccionar en un instante pequeño de tiempo a los cambios que se produzcan en él
- *proactividad*: los agentes no son sólo entidades que reaccionan a su entorno, sino que son capaces de exhibir un comportamiento dirigido por objetivos tomando iniciativa
- *sociabilidad*: los agentes son capaces de comunicarse con otros agentes en algún lenguaje de comunicación de agentes.

Podríamos decir que un agente es una evolución de un objeto a un nivel de abstracción mayor y con autonomía (por ejemplo, puede decidir si llevar a cabo una acción o no).

En los Sistemas Multi-Agentes (SMA) [123], el sistema está compuesto de varios agentes que se comunican usando un Lenguaje de Comunicación de Agentes (ACL) que se basa en la teoría de actos de habla [104]. Este lenguaje ofrece un nivel de abstracción mayor que las primitivas proporcionadas por los sistemas de objetos distribuidos tradicionales.

Desde hace dos décadas, los agentes software y sistemas multi-agente han crecido como un área de investigación muy activa, y se han ido proponiendo distintas metodologías para la construcción de sistemas multiagente. En la siguiente sección se revisan algunas de las más relevantes.

2.1.1. Metodologías

En esta sección se resumen algunas de las metodologías más prestigiosas para la construcción de sistemas multiagente. Existen a su vez varios artículos que resumen y/o comparan diversas metodologías, también desde diversos puntos de vista, como [27, 28, 61, 117, 127].

2.1.1.1. Extensiones de metodologías provenientes de la Ingeniería del Conocimiento

Varias de las primeras metodologías para la construcción de sistemas multiagente se basaron en metodologías existentes en otros campos, como la ingeniería del conocimiento, adaptándolas para incluir las características específicas de los agentes. En particular, varias de ellas se basaron en CommonKADS [102], como CoMoMAS y MAS-CommonKADS. Ambas metodologías se componen de varios modelos, la mayoría equivalentes a los definidos en CommonKADS aunque adaptados al contexto de los sistemas basados en agentes.

En CoMoMAS [53] se definen modelos de *Experiencia*, *Tareas* y *Diseño* de forma muy similar a como se hace en CommonKADS, además de los modelos de *Agentes*, que es el modelo principal y define la arquitectura y conocimiento de agente, el modelo de *Cooperación* entre agentes y el modelo del *Sistema*,

que define los aspectos organizacionales de la sociedad de agentes junto con aspectos arquitecturales de los agentes.

MAS-CommonKADS [62] comienza con una fase de conceptualización informal para recoger los requisitos del usuario y obtener una primera descripción del sistema desde el punto de vista del usuario. Igual que CoMoMAS, toma los modelos de *Experiencia*, *Tareas* y *Diseño* análogos a CommonKADS, y los complementa con el modelo de *Agentes*, un modelo de *Coordinación* que describe las conversaciones entre agentes, es decir, sus interacciones, protocolos y capacidades requeridas, un modelo de *Organización* que describe la organización en la que se va a introducir el sistema y la organización de la sociedad de agentes (jerarquía de agentes, relaciones entre agentes y el entorno, y la estructura de la sociedad de agentes), y un modelo de *Comunicación* que detalla las interacciones usuario-software, y los factores humanos para desarrollar estas interfaces.

2.1.1.2. AAII

Esta metodología [65] adapta y extiende metodologías orientadas a objetos (OO). Aunque se basa en arquitectura BDI (Belief-Desire-Intention), es suficientemente general. Emplea un conjunto de modelos a dos niveles de abstracción. Desde un punto de vista *externo*, se centra en los agentes y relaciones entre ellos. Desde un punto de vista *interno*, se emplea un conjunto de modelos dependientes de la arquitectura utilizada, en este caso, creencias, objetivos y planes.

La descripción de un sistema desde el punto de vista *externo* se realiza mediante dos modelos, independientes de la arquitectura BDI:

- *Modelo de Agentes*: describe la relación jerárquica entre diferentes clases abstractas y concretas de agentes, e identifica las instancias que pueden existir de esas clases.
- *Modelo de Interacción*: describe las responsabilidades de una clase de

agentes, los servicios que proporciona, las interacciones asociadas y las relaciones de control entre clases de agentes. Incluye la sintáctica y semántica de los mensajes intercambiados.

Desde el punto de vista *interno*, cada clase de agentes es especificada por tres modelos, específicos para la arquitectura BDI:

- *Modelo de Creencias*: describe la información sobre el entorno y estado interno que una clase de agentes puede tener, y las acciones que puede realizar.
- *Modelo de Objetivos*: describe los objetivos que un agente puede adoptar y los eventos a los que puede responder.
- *Modelo de Planes*: describe los planes que un agente podría emplear para alcanzar sus objetivos o responder a eventos que percibe.

La metodología sigue una descomposición del sistema basada en los roles clave en una aplicación. La identificación de roles y sus relaciones guía la especificación de la jerarquía de clases de agentes. El análisis de las responsabilidades de cada clase de agente lleva a la identificación de los servicios proporcionados y usados por un agente, y de ahí sus interacciones externas. Aspectos como la creación y duración de roles y sus interacciones determinan las relaciones de control entre clases de agentes.

La construcción de los modelos externos se realiza en cuatro pasos principales:

1. Identificar los roles del dominio de aplicación y el tiempo de vida de cada rol. Elaborar una jerarquía de clases de agentes.
2. Para cada rol, identificar sus responsabilidades y servicios proporcionados y usados. Descomponer las clases de agentes al nivel de servicio.
3. Para cada servicio, identificar las interacciones, actos de habla y contenido. Identificar eventos y condiciones a tener en cuenta, acciones a ser

ejecutadas. Determinar las relaciones de control entre agentes. En este punto se puede llevar a cabo el modelado interno de cada clase de agente.

4. Refinar la jerarquía de agentes.

En cuanto a los modelos internos el proceso se resume en dos pasos:

1. Analizar los medios de alcanzar los objetivos. Para cada objetivo, analizar los diferentes contextos en los que el objetivo tiene que ser alcanzado y para cada contexto, descomponer cada objetivo en actividades (subobjetivos).

2. Construir las creencias del sistema. Analizar los diferentes contextos y las condiciones que controlan la ejecución de actividades y acciones, y descomponerlas en componentes de creencias.

2.1.1.3. GAIA

GAIA [129] es quizá la metodología más conocida. Se aplica a sistemas donde la estructura organizativa es estática, el número de agentes y las relaciones entre ellos no cambian en tiempo de ejecución, al igual que las habilidades y servicios que proporcionan los agentes.

El *análisis* y *diseño* se puede ver como un proceso de construcción de *modelos* del sistema cada vez más detallados. GAIA anima al desarrollador a pensar en construir sistemas basados en agentes como un proceso de *diseño organizacional*

Los principales conceptos que se utilizan en GAIA se dividen en dos categorías: *abstractos* (utilizados durante el análisis) y *concretos* (utilizados en el diseño).

El objetivo de la fase de *análisis* es el entendimiento del sistema y su estructura por medio de la *organización* del sistema (colección de roles). La entidad más abstracta en la jerarquía de conceptos es el *sistema*, en el sentido de sociedad u organización. El siguiente nivel en la jerarquía es el *rol*. Un rol se define

por cuatro atributos: las *responsabilidades* determinan la funcionalidad, los *permisos* son los recursos disponibles por el rol, las *actividades* son las tareas que un rol puede realizar sin interactuar con otros agentes y, por último, los *protocolos* de interacción con otros roles.

El modelo de organización se compone de dos modelos: *modelo de roles* y *modelo de interacción*. El *Modelo de Roles* identifica los roles importantes en el sistema. Se caracterizan por sus permisos y responsabilidades. El *Modelo de Interacción* consiste en un conjunto de definiciones de protocolos, uno por cada tipo de interacción entre roles. Por cada protocolo se define el propósito, iniciador, participante, entradas, salidas y procesamiento.

El proceso de análisis se resume en las siguientes fases:

1. Identificar informalmente los *roles* del sistema
2. Por cada rol, definir los protocolos asociados (modelo de interacción)
3. Utilizando el modelo de interacción, construir el modelo de roles
4. Iterar las fases 1-3

El objetivo de la fase de *diseño* es transformar los modelos generados en el análisis en abstracciones de suficiente bajo nivel que permita aplicar técnicas de diseño tradicionales para implementar los agentes. En GAIA el diseño consiste en la generación de tres modelos.

El *Modelo de Agente* identifica los tipos de agentes que compondrán el sistema y las instancias de esos tipos. Un tipo de agente se compone de uno o más roles. El modelo se define mediante un árbol de tipos de agentes cuyas hojas son roles.

El *Modelo de Servicios* identifica los principales servicios (funciones) necesarios para llevar a cabo el rol del agente. Para cada servicio se identifican las entradas, salidas, precondiciones y poscondiciones. Se basa en la lista de protocolos, actividades y responsabilidades de un rol.

El *Modelo de Conocidos* define las líneas de comunicación (no los mensajes) entre tipos de agentes mediante un grafo dirigido, con el objetivo de identificar cuellos de botella potenciales.

El proceso de diseño consiste en:

1. Crear un modelo de agente: agregar roles en tipos de agentes y refinarlos para formar una jerarquía de tipos de agentes
2. Construir un modelo de servicios
3. Construir el modelo de conocidos a partir del modelo de interacción y modelo de agente

Más recientemente, con la ayuda de Zambonelli [134], GAIA fue extendido enfatizando las abstracciones organizacionales que consideran necesarias para el análisis y diseño de MASs. Estos conceptos no sólo incluyen los *roles* e *interacciones* incluidos en la mayoría de las metodologías, sino también el *entorno* (también bastante común), las *reglas organizacionales* (que define las restricciones que deben cumplirse en una organización) y las *estructuras organizacionales* (definen explícitamente la arquitectura de la organización, frente a la definida implícitamente por el modelo de roles).

La fase de *análisis* consiste en la identificación de los *objetivos de las organizaciones* que constituyen el sistema y su comportamiento global esperado (incluye descomponer la organización global en suborganizaciones poco acopladas), el *modelo de entorno*, el *modelo preliminar de roles* (sin la imposición de ninguna estructura organizacional), el *modelo preliminar de interacción*, y las *reglas* que deberían respetar y hacer respetar la organización en su comportamiento global.

La fase de *diseño* se descompone en una fase de diseño arquitectural y una fase de diseño detallado. La fase de diseño arquitectural incluye la definición de la *estructura organizacional* del sistema en términos de su topología y régimen de control (explotando la existencia de catálogos de patrones organizacionales) y la *finalización de los modelos preliminares de roles e interacción*. La fase de

diseño detallado cubre la definición del *modelo de agente* y la definición del *modelo de servicios* (el modelo de conocidos deja de tener relevancia).

2.1.1.4. ROADMAP

ROADMAP [64] extiende la versión original de GAIA [129] mejorando algunos puntos débiles de esta metodología. En la fase de especificación y análisis se crean los siguientes modelos secuencialmente, donde cada modelo toma como entrada todos los modelos previos. Estos modelos se refinan iterativamente hasta que se dispone de suficiente información. Son los siguientes

- *Modelo de Casos de Uso*. Se adapta a partir de las metodologías OO. En este caso la semántica es diferente, en lugar de imaginar al usuario interactuando con el sistema software, se imagina al usuario interactuando con un equipo de agentes abstractos ideales.
- *Modelo de Entorno*. Se construye una jerarquía de zonas en el entorno y un conjunto de esquemas de zonas que describen cada una de ellas. Un esquema incluye una descripción textual de la zona así como los atributos objetos estáticos (con los que los agentes no interactúan explícitamente), objetos (con los que sí se interactúa), restricciones, fuentes de información y asunciones sobre la zona. Para construir este modelo se basa en los escenarios del modelo de casos de uso.
- *Modelo de Conocimiento*. Representa el conocimiento global del sistema. El modelo consiste en una jerarquía de componentes de conocimiento, junto con la descripción de cada componente. A partir de los modelos de casos de uso y de entorno se identifica el conocimiento necesario en cada zona por caso de uso. Cuando se crean componentes de conocimiento se asignan a roles. Cuando cambia el comportamiento esperado del sistema, el conocimiento de los roles puede ser revisado.
- *Modelo (revisado) de Roles*. Extiende el modelo de roles de GAIA. Ahora se compone de una jerarquía de roles y un conjunto de esquemas de ro-

les que describen cada rol. La jerarquía se representa mediante un árbol, donde las hojas son roles atómicos, que mantienen la semántica original de GAIA. El resto de roles son compuestos y se definen en términos de otros roles (a su vez compuestos o atómicos) mediante relación de agregación. La semántica de agregación es diferente del enfoque tradicional OO. Los sub-roles individuales se dice que participan en el super-rol. La jerarquía de roles modela las sociedades a diferentes niveles de abstracción, en lugar de descomponer el control. Además se permite a los roles modificar las definiciones de otros roles.

- *Modelo de Protocolos*. Es el mismo que modelo de interacción de GAIA

- *Modelo de Interacción*. Se basa en diagramas de interacción AUML[1], con roles y zonas representados gráficamente.

La fase de diseño se compone de los mismos tres modelos originales de GAIA sin sufrir apenas variaciones.

2.1.1.5. MESSAGE/INGENIAS

MESSAGE [20] extiende UML aportando conceptos a nivel de conocimiento de agente y diagramas para visualizarlos que extienden los diagramas de clases y actividad de UML.

Un modelo de análisis es una red de clases e instancias interrelacionadas derivadas de los conceptos definidos en el meta-modelo. MESSAGE define un número de vistas que se enfocan en partes del modelo.

- *Vista de Organización*: muestra entidades concretas (agentes, organizaciones, roles, recursos) del sistema y su entorno y relaciones principales entre ellos.

[1] http://www.auml.org/

- Vista de Objetivo/Tarea: muestra Objetivos, Tareas, Situaciones y las dependencias entre ellos. Se construyen grafos de descomposición de objetivos en sub-objetivos y de dependencias temporales.

- *Vista de Agente/Rol*: se enfoca en agentes y roles. Para cada agente/rol se utiliza un esquema con diagramas para describir sus características (objetivos, eventos, recursos, tareas, ...)

- *Vista de Interacción*: para cada interacción entre agentes/roles, muestra el iniciador, colaboradores, motivador (normalmente un objetivo), información relevante proporcionada/alcanzada por cada participante, eventos que disparan la interacción y otros efectos.

- *Vista de Dominio*: muestra conceptos específicos del dominio y relaciones relevantes para el sistema en desarrollo.

Los modelos de análisis se producen mediante pasos de refinamiento. El nivel mas alto de descomposición es el nivel 0. Consiste en definir el sistema a construir. El sistema se ve como un conjunto de organizaciones que interactúan con recursos, actores u otras organizaciones. Los actores pueden ser usuarios humanos u otros agentes. Las siguientes etapas de refino consisten en la creación de modelos a nivel 1, nivel 2, etc. A nivel 0 el proceso de modelado comienza construyendo las vistas de Organización y de Objetivos/Tareas. Estas vistas actúan como entradas para crear las vistas de Agente/Rol y Dominio. Finalmente la vista de Interacción se construye partiendo de los demás modelos. El nivel 0 da una vista general del sistema, su entorno y su funcionalidad global. En el nivel 1 se define la estructura y comportamiento de entidades como organización, agentes, tareas, objetivos del dominio. Se pueden definir niveles adicionales, aunque MESSAGE sólo considera los niveles 0 y 1. Para refinar los modelos de nivel 0 se pueden seguir varias estrategias (centrado en organizaciones, agentes, descomposición de objetivos/tareas, ...).

A partir de los resultados obtenidos en MESSAGE surge INGENIAS [94], que mejora MESSAGE en tres aspectos:

- Integración de las vistas de diseño del sistema
- Integración de resultados de investigación
- Integración con el ciclo de vida de desarrollo de software

Esta metodología propone un lenguaje de especificación de Sistemas Multi-Agente así como su integración en el ciclo de vida. Siguiendo el ejemplo de MESSAGE, el lenguaje se especifica con meta-modelos y lenguaje natural. La integración en el ciclo de vida se consigue definiendo un conjunto de entregas y actividades involucradas en el desarrollo. Existe también un conjunto de herramientas de soporte para la metodología.

2.1.1.6. MaSE

MaSE [32] describe completamente el proceso que guía al desarrollador desde la especificación inicial del sistema hasta su implementación. Este proceso se divide en siete pasos, en las que los modelos generados en cada una son entradas de algunas de las siguientes. Los pasos se dividen en dos fases.

Los tres primeros pasos corresponden a la fase de Análisis:

- *Captura de objetivos*: transforma la especificación inicial del sistema en una jerarquía estructurada de objetivos del sistema.
- *Aplicar casos de uso*: crea diagramas de casos de uso y de secuencia a partir de la especificación inicial del sistema. Los casos de uso representan las interacciones lógicas entre varios roles.
- *Refinar roles*: crea los roles que son responsables de los objetivos definidos en el primer paso. Además se crea un conjunto de tareas, que definen cómo resolver los objetivos relativos al rol.

Los siguientes pasos forman la fase de Diseño:

- *Crear clases de agentes*: especifica un diagrama de clases de agentes. Las clases de agentes se definen en términos de los roles que los agentes juegan y las conversaciones en las que deben participar.
- *Construir conversaciones*: define los protocolos de coordinación entre pares de agentes mediante autómatas de estados finitos (diagramas de estado).
- *Ensamblar clases de agentes*: se crea la funcionalidad interna de las clases de agentes.
- *Diseño del sistema*: en este último paso se crean las instancias de agentes y se crea la estructura final del sistema.

Aunque la organización de las fases anteriores es secuencial, el analista o diseñador puede moverse libremente entre ellas para completar los diferentes modelos.

2.1.1.7. Prometheus

Prometheus [91] es una metodología dirigida a no expertos. Se compone de tres fases.

- *Especificación del sistema*: incluye dos actividades, (i) determinación del entorno del sistema, y (ii) determinación de los objetivos y funcionalidad del sistema. El entorno se define en términos de percepciones y acciones. La funcionalidad del sistema se realiza identificando objetivos, funcionalidades para alcanzarlos, y definiendo escenarios de casos de uso.
- *Diseño de la arquitectura*: incluye tres actividades, definición de tipos de agentes, definición de la estructura global del sistema, y la definición de las interacciones entre agentes. Los tipos de agentes se obtienen agrupando funcionalidades. Cada tipo de agente se describe usando un descriptor que describe el ciclo de vida de ese tipo, sus funcionalidades,

los datos que usa y produce, sus objetivos, los eventos a los que debería responder, sus acciones y los otros tipos de agentes con los que interactúa. La estructura general del sistema es capturada en un diagrama que es el artefacto de diseño más importante. Proporciona una visión general del sistema, mostrando los tipos de agentes, las comunicaciones entre ellos, y los datos. También muestra los límites del sistema y su entorno. La dinámica del sistema se captura mediante los protocolos de interacción, desarrollados mediante diagramas de interacción obtenidos a partir de los escenarios.

- *Diseño detallado*: describe la estructura interna de cada agente y cómo llevará a cabo las sus tareas en el sistema. Para cada tipo de agente define sus capacidades, eventos internos, planes y estructura de datos detallada.

Prometheus se soporta mediante dos herramientas, el entorno de desarrollo JACK (JDE)[2] y el entorno de diseño de Prometheus (PDT).

2.1.1.8. Tropos

Una de las principales diferencias de Tropos [16] con el resto de metodologías es su fuerte enfoque en el análisis de requisitos. Se compone de cinco fases:

1. *Requisitos Iniciales (Early)*: usa el concepto de actor y objetivos. Divide los objetivos en dos tipos, fuertes (funcionales) y débiles (no funcionales). En esta fase hay dos modelos para representar objetivos y actores. El *diagrama de actores* describe los actores y sus relaciones (dependencias sociales) en el dominio. El *diagrama de objetivos* muestra el análisis de objetivos y planes con respecto a un actor específico que tiene la responsabilidad de conseguirlos. Se utilizan tres técnicas básicas de razonamiento: análisis de medios fines, descomposición Y/O y análisis de contribución.

[2] www.agent-software.com

2. *Requisitos Finales (Late)*: en esta fase se extienden los modelos creados en la fase anterior. El objetivo de esta fase es modelar el sistema en su entorno. El sistema se modela como uno o más actores. Las interdependencias con otros actores contribuyen a alcanzar los objetivos. Por tanto, estas dependencias definen los requisitos funcionales y no funcionales del sistema.

3. *Diseño de la Arquitectura*: se definen tres pasos que se pueden aplicar en esta fase. En el primero, se incluyen y describen nuevos actores extendiendo el *diagrama de actores*, basándose en la elección del estilo arquitectural. En el segundo paso se identifican las capacidades, y finalmente se agrupan para formar tipos de agentes.

4. *Diseño Detallado*: consiste en la especificación de agentes a nivel micro. Hay tres tipos de diagramas para detallar las capacidades, planes e interacciones. *Diagramas de actividad* UML para representar capacidades y planes, *diagramas de planes* y *diagramas de interacción entre agentes*.

5. *Implementación*: elige una plataforma BDI para la implementación (JACK). Se proporcionan guías y heurísticas para adaptar conceptos de Tropos a BDI y de BDI a JACK.

2.1.1.9. RICA

RICA (Role/Interaction/Communicative Action) [105, 106, 108] usa conceptos comunes en la literatura sobre modelos organizacionales como *roles e interacciones sociales*, y relaciona estas nociones con conceptos comunicativos que son particularmente relevantes en sistemas de agentes, como *protocolos de interacción* y *acciones comunicativas*. En particular, los *roles comunicativos* y las *interacciones comunicativas* se conciben como las clases especiales de roles e interacciones sociales que encapsulan la competencia pragmática que los roles sociales deben poseer para tomar parte en interacciones sociales de acuerdo a las reglas de los protocolos. El metamodelo RICA se encuentra descrito visualmente mediante el diagrama de clases UML representado en la figura

2.1.

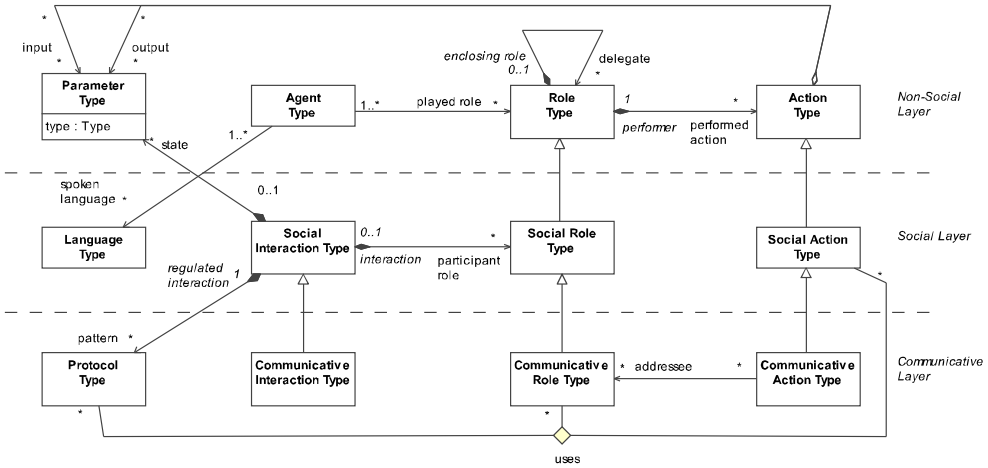


Figura 2.1: Metamodelo RICA

Serrano [105] realiza un análisis detallado del lenguaje de comunicación de agentes definido por el estándar FIPA [47, 48, 50] en base al modelo RICA, obteniendo una taxonomía de roles, mostrada en la figura 2.2, y que se usa como base y puede ser extendido mediante el análisis de otras aplicaciones, como se describe a continuación.

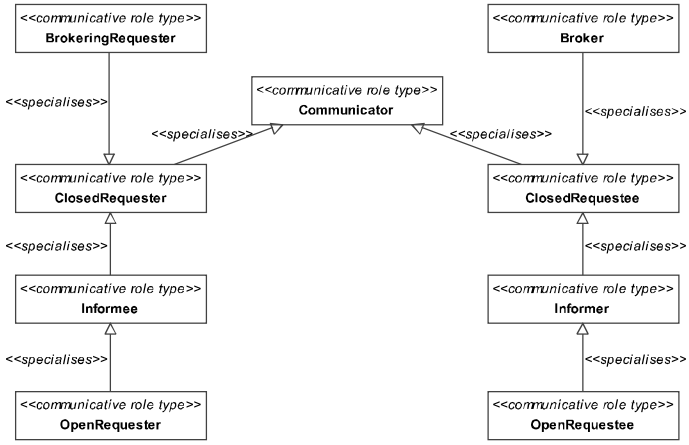


Figura 2.2: Taxonomía de roles para el estándar FIPA

De momento obsérvese que en este modelo organizacional subyacente a FIPA ACL, existe una relación de especialización/generalización entre algunas interacciones que permite organizarlas como una jerarquía de especialización de roles. Una interacción de intercambio de información, por ejemplo, se puede ver como una clase específica de interacción de solicitud de realización de una acción, donde la acción deseada es una acción comunicativa de tipo *inform*.

Además del metamodelo RICA, también se acompaña de un método de diseño de lenguajes de comunicación y protocolos de interacción basado en el metamodelo RICA. Este método parte de un modelo organizacional inicial ("a priori") que especifica los roles dependientes del dominio y los tipos de interacción jugados por las diferentes clases de agentes en el sistema multiagente. Además también toma como entrada la biblioteca de protocolos de interacción existente de FIPA (FIPA IPL [51]). El método de diseño consta de cuatro pasos:

1. *Identificación de diálogos*. Se identifica una muestra de diálogos en lenguaje natural para los diferentes tipos de interacción dependientes del dominio.

2. *Análisis pragmático*. Tiene como objetivo identificar los tipos de acciones comunicativas que mejor describan el comportamiento comunicativo de los agentes en los diálogos, mediante expresiones ilocutivas en lenguaje natural. Además, también se consideran los patrones particulares de mensajes intercambiados para cada diálogo. Como resultado se obtiene una primera versión del lenguaje de comunicación (MAS ACL), constituido por los performativos asignados por los diseñadores del sistema, que mejor expresan los tipos de actos de habla realizados, y por un conjunto de diagramas de secuencia UML que resumen la información necesaria.

3. *Análisis de reusabilidad*. En este paso se analiza el conjunto de verbos de acto de habla preliminar con el fin de identificar las acciones comunicativas y protocolos estándar definidos por FIPA. Como consecuencia, se produce un primer refinamiento del modelo inicial, ya que ciertos roles

que realizan ciertos tipos de acciones comunicativas en el modelo inicial se modelan como especializaciones de los roles implícitos en FIPA ACL.

4. *Diseño de componentes no estándar*. Se realiza en el caso de acciones comunicativas y/o protocolos que carecen de una contrapartida adecuada en el estándar FIPA. En primer lugar se realiza un refinamiento del modelo organizativo, que consiste en abstraer los roles genéricos (y sus tipos de interacción particulares) a partir de los dependientes del dominio, analizando sus expresiones ilocutivas características incluidas e el catálogo preliminar. Este mecanismo de abstracción permite fomentar la reusabilidad de las extensiones realizadas al catálogo FIPA CAL [50]. A continuación se diseñan los tipos de acciones comunicativas y los protocolos de interacción.

2.1.2. Coordinación mediante agentes intermediarios

Igual que ocurre en muchos ámbitos de las sociedades humanas, los agentes intermediarios juegan un papel importante en las sociedades de agentes artificiales. Estos agentes proporcionan medios para facilitar la coordinación entre agentes en entornos abiertos, mediante servicios de localización y comunicación entre agentes que proporcionan servicios y agentes que los solicitan. En general, en este contexto se pueden distinguir tres categorías de agentes: proveedores de servicios, solicitantes de servicios y agentes intermediarios. El proceso básico de mediación es el siguiente:

1. Los agentes proveedores de servicios anuncian sus *capacidades* (servicios capaces de llevar a cabo) a los agentes intermediarios.

2. Los agentes intermediarios almacenan estos anuncios.

3. Un agente solicitante pide a algún intermediario la localización y comunicación con agentes que proporcionen determinado servicio. Esta intermediación puede incluir una transacción completa y otros servicios añadidos.

4. El agente intermediario procesa la petición, analizando las capacidades de los agentes registrados y devuelve el resultado. El resultado puede ser un subconjunto de anuncios junto con los nombres de los proveedores correspondientes para contactar, o el resultado de la transacción completa utilizando el servicio más adecuado.

La idea en principio es simple pero complicada de llevar a cabo en sistemas abiertos en los que coexisten agentes heterogéneos, desarrollados por diferentes entidades y donde los cambios son frecuentes. Por tanto, la funcionalidad esencial de cualquier clase de agente intermediario es facilitar la interoperabilidad de servicios, agentes y sistemas, de una forma flexible, fiable y segura, y respetando los marcos legales existentes.

En general, los agentes intermediarios son agentes que ayudan a otros en la localización y su conexión con proveedores de servicios [31]. Además, cualquier agente intermediario se puede caracterizar por sus habilidades sobre:

- Prestación de servicios básicos de mediación a la sociedad de agentes.

  Los agentes intermediarios deben gestionar las capacidades y descripciones de servicios de los agentes. Esto significa que tienen que ser capaces entender y procesar de forma automática peticiones y descripciones de servicios. Para ello se debe utilizar algún lenguaje común de descripción de servicios, de forma que sea posible comparar las solicitudes de servicios con los servicios registrados y determinar cuáles de los servicios registrados se adapta mejor al solicitado. La elección del mecanismo de equiparación entre peticiones y servicios registrados depende de la estructura y semántica de las descripciones de servicios, así como de la clase y calidad deseada del resultado del proceso de equiparación. Por ejemplo, se puede utilizar comparación simple de palabras claves y valores, usar inferencia sobre estructuras y tipos de datos, y/o utilizar mecanismos más complejos como subsunción de conceptos.

  Una de las principales dificultades que los agentes intermediarios encuentran en la mediación de servicios es la heterogeneidad tanto sintáctica

como semántica de los datos y conocimiento que tienen que procesar. La mayoría de los métodos para resolver el problema de heterogeneidad semántica se basan en la utilización parcial o global de conocimiento ontológico que puede ser compartido entre los agentes. Esto implica que el agente intermediario debe proporcionar servicios que permitan gestionar ontologías generales y de dominios específicos, así como relaciones entre ontologías. El uso de lenguajes comunes de representación de ontologías reduce el problema.

Los agentes intermediarios disponen de mecanismos internos para almacenamiento y gestión eficiente de datos y conocimiento de sí mismo y de otros agentes. La información que almacenan no sólo es la descripción de las capacidades y servicios de los agentes registrados sino que también pueden guardar peticiones anteriores de agentes, conocimiento ontológico compartido entre agentes, etc. Esta información podría ser utilizada para mejorar la eficiencia y calidad de los servicios del agente intermediario.

En algunos casos el agente intermediario tiene que realizar consultas a múltiples bases de datos u otras fuentes de información externas para recopilar información para los solicitantes de algún servicio. Esto requiere que el intermediario tenga el conocimiento para realizar planificación de consultas y ejecución de transacciones distribuidas.

- Coordinación de los servicios de mediación.

  La coordinación de la mediación requiere por parte del agente intermediario capacidad de comunicación con agentes, sistemas y usuarios implicados en el proceso de mediación de acuerdo a protocolos, convenciones y políticas existentes.

  La localización de agentes relevantes por el agente intermediario presupone que sus clientes están registrados y son accesibles en cualquier momento, lo que implica la existencia de servicios básicos de registro y nombrado. Además, un agente intermediario debe ser capaz de acceder a otras sociedades de agentes, permitiendo así que un agente solicitante de un servicio se comunique con un proveedor que pertenece a otro sistema

multiagente, y sin que esos agentes sean conscientes de que pertenecen a sociedades distintas. Esto se consigue mediante la colaboración entre los agentes intermediarios [52].

Cualquier interacción de coordinación entre cualquier tipo de agente se realiza mediante el uso más o menos estandarizado de algún lenguaje de comunicación entre agentes (como FIPA ACL [48]), protocolos de comunicación y políticas de comunicación.

Además de la capacidad de comunicación con sus clientes un agente intermediario también puede proporcionar servicios de interfaz a usuarios humanos para permitirles ver la información sobre los servicios disponibles en la sociedad de agentes.

- Fiabilidad de la mediación.

  Los agentes intermediarios deben proporcionar confianza y calidad de servicio a los clientes que requieren sus servicios. Factores de calidad son la accesibilidad, utilidad y disponibilidad de los servicios; corrección, completitud, consistencia y redundancia de datos; así como funcionalidad, eficiencia, mantenibilidad y portabilidad de la implementación de los servicios de mediación. Además de la calidad del servicio es necesario que el agente intermediario inspire confianza a sus clientes garantizando privacidad de datos, anonimato, y verificación de las supuestas capacidades de los agentes registrados [125].

  Los agentes intermediarios deben ser robustos frente a manipulación externa o ataques de agentes o entes externos. Por otro lado, los clientes de un agente intermediario no deben hacer un mal uso de los datos revelados por el agente intermediario durante la mediación. Hay, por tanto, una necesidad de que todas las partes implicadas en la mediación utilicen un modelo de confianza adecuado que incluya mecanismos como el uso de certificados de seguridad, cálculo e intercambio de matrices de confianza, reputación, etc.

Podemos distinguir principalmente tres **clases de agentes intermediarios** [69]: mediador, broker y matchmaker.

Un agente **Mediador** [124] realiza una función de interfaz a los usuarios (solicitantes de servicios) frente a datos y recursos de conocimiento. El escenario más común de actuación de un mediador es en un sistema de información en el que actúa como unidad principal con un conjunto de agentes (wrappers) que proporcionan acceso a algún tipo de fuente de información. El mediador proporciona un modelo de información global, integrado semánticamente y consistente. Además ofrece servicios añadidos del dominio de la aplicación. Recibe las peticiones de los agentes clientes y las responde utilizando el modelo de información global o realizando consultas (distribuidas) a los agentes wrappers.

Un agente **Broker** (o Facilitador) realiza una función de interfaz entre agentes que proporcionan servicios y los que utilizan esos servicios haciendo de intermediario en sus transacciones. Todas las comunicaciones entre pares de agentes pasan a través del broker. Típicamente, contacta con los agentes proveedores más relevantes, negocia, ejecuta y controla las transacciones correspondientes y devuelve el resultado al agente que requirió el servicio. A diferencia de los mediadores, un broker no proporciona un modelo de información global. Tampoco tiene tantos servicios añadidos de mediación como un mediador sino que se basa en la colaboración y negociación con otros agentes. Puesto que no está restringido a comunicarse con un conjunto fijo de wrappers definidos a priori, como sí ocurre con los mediadores, los brokers suelen actuar en entornos dinámicos en los que los recursos y agentes pueden entrar y salir continuamente de la sociedad.

Un **Matchmaker** es un agente que empareja solicitantes con proveedores de servicios mediante la equiparación de solicitudes con servicios anunciados por agentes. A diferencia de mediadores y brokers, un matchmaker simplemente devuelve una lista (valorada) ordenada de agentes que proporcionan el servicio solicitado. Cada elemento de la lista lleva asociado un valor que indica el grado de adaptación al servicio solicitado. El agente que solicita el servicio tiene que elegir el agente proveedor y negociar con él. Toda la interacción, por tanto, es independiente al matchmaker, evitando además cuellos de botella.

2.1.3. Estándares y herramientas

Para una correcta evolución del paradigma basado en agentes de construcción de sistemas es necesario la existencia de estándares que faciliten la interacción entre agentes y herramientas que ayuden en el desarrollo de aplicaciones.

FIPA[3] (Foundation for Intelligent Physical Agents) es una organización internacional de la IEEE Computer Society dedicada a promover la tecnología basada en agentes produciendo estándares para proporcionar interoperabilidad entre agentes heterogéneos, así como de sus estándares con otras tecnologías. FIPA ha jugado un rol fundamental en el desarrollo de estándares para agentes y ha promocionado un número de iniciativas y eventos que han contribuido al desarrollo de esta tecnología. Además, muchas de las ideas originalmente desarrolladas en FIPA están siendo utilizadas en la nueva generación de tecnologías de la Web.

Las especificaciones de FIPA representan una colección de estándares dirigidos fundamentalmente a promover la interoperación de agentes heterogéneos y los servicios que representan. El conjunto de especificaciones se clasifican en diferentes categorías: comunicación entre agentes, transporte, gestión de agentes, arquitectura abstracta y aplicaciones. La categoría más importante es la comunicación entre agentes.

FIPA especifica una arquitectura abstracta con el objetivo de fomentar la interoperabilidad y la reutilización. Para ello, identifica elementos que tienen que ser codificados de forma que queden reflejadas las características comunes de los diferentes sistemas que usan distintas tecnologías. El objetivo principal es conseguir el intercambio de mensajes con contenido semántico entre agentes, que pueden utilizar diferentes mecanismos de transporte, lenguajes de comunicación (ACLs) o lenguajes de contenido. Por eso se tienen en cuenta los siguientes aspectos:

- Interoperabilidad en transporte de mensajes

[3] http://www.fipa.org/

- Varias formas de representación de lenguajes de comunicación
- Varios lenguajes de contenidos de mensajes
- Múltiples representaciones de servicios de directorio

La arquitectura abstracta no puede ser implementada directamente sino que es la base para especificaciones concretas, que describen en detalle cómo construir sistemas de agentes en términos de elementos software (lenguajes, interfaces, protocolos de red, sistemas operativos, etc.).

Por otra parte, la especificación no prohibe que una implementación concreta añada nuevos elementos en que sean útiles para construir un buen sistema sino que establece el conjunto mínimo de elementos necesarios. La arquitectura abstracta también describe elementos opcionales que, aunque son opcionales a nivel abstracto, podrían ser obligatorios en una implementación concreta.

Existen ya gran cantidad de herramientas para construir sistemas basados en agentes, algunas asociadas a metodologías concretas otras no, siguiendo o no el estándar FIPA, proporcionando mayor o menor nivel de abstracción, comerciales o libres. Básicamente estas plataformas abstraen los detalles de implementación relacionados con la comunicación de mensajes. Además, unas más que otras, proporcionan herramientas para las diferentes fases de construcción de sistemas basados en agentes. Puede encontrarse una lista bastante amplia en http://www.multiagent.com/Software/Tools_for_building_MASs/index.html.

Entre todas las plataformas existentes, seguramente la más utilizada sea JADE (Java Agent DEvelopment Framework) [7]. Implementado en Java, la plataforma JADE puede estar distribuida en varias máquinas. Los agentes pueden migrar o clonar a otras máquinas de la plataforma independientemente del sistema operativo. El ciclo de vida de los agentes se puede controlar remotamente a través de una interfaz gráfica, que permite arrancar herramientas de depuración. La arquitectura de comunicaciones intenta ofrecer (transparente al agente) envío de mensajes de forma flexible y eficiente eligiendo el mejor protocolo de transporte compatible con FIPA. Permite la construcción de agentes

con mecanismo de razonamiento basado en Jess[4] (Java Expert System Shell) [57], un sistema de reglas de producción.

El éxito de JADE seguramente se debe a que fue una de las primeras plataformas no comerciales que implementaron el estándar FIPA, lo que hizo que una gran comunidad lo utilizase y permitió el desarrollo de sucesivas versiones. Pese a que no proporciona herramientas de alto nivel que apoyen la construcción de modelos en las primeras fases de desarrollo, su éxito es innegable.

2.2. Computación Orientada a Servicios

En esta sección se revisan los conceptos fundamentales de la computación orientada a servicios, centrándose principalmente en la tecnología de los servicios Web y servicios Web semánticos, así como los métodos y técnicas existentes para la descripción, descubrimiento y composición de servicios.

2.2.1. Servicios Web (Semánticos)

En los últimos años, el gran desarrollo de los ordenadores personales, redes de ordenadores y la creación de la World Wide Web ha cambiado en gran medida la vida, dando lugar a la denominada sociedad de la información. Mientras que las capacidades y ámbito de la actual Web son impresionantes, su continua evolución hacia una recurso con características inteligentes presenta muchos desafíos. Actualmente, el problema de ejecutar tareas inteligentes, como los Servicios Web, es que deben utilizar lenguajes propietarios. La solución de la World Wide Web Consortium (W3C)[5] es ilustrar el camino para desarrollar una arquitectura Web construida sobre diversas capas de lenguajes de marcado estándar que pueden utilizarse mediante servicios Web semánticos. En este apartado se comienza repasando los orígenes de la Web y los servicios Web, para continuar profundizando en las técnicas existentes en el campo de los

[4] http://herzberg.ca.sandia.gov/
[5] http://www.w3.org/

servicios Web, la Web Semántica y los servicios Web semánticos.

2.2.1.1. Orígenes de la Web y los servicios Web

Se podría decir que la historia de la Web se inicia en el año 1962 cuando RAND Corporation comenzó una investigación financiada por las fuerzas armadas de Estados Unidos para determinar cómo desarrollar redes distribuidas robustas para control militar que sobrevivieran a un ataque nuclear, de forma que aunque parte de la red fuera destruida todavía se tuviera el control sobre las armas militares para un contraataque. Este estudio inicial dio lugar al desarrollo de la red ARPANET, que ademas de su robustez frente ataques promocionara la compartición de supercomputadores entre investigadores científicos y militares en EEUU. ARPANET se creó en 1969 con cuatro nodos y en los siguientes años se fueron añadiendo nodos y características. En los 70 se introdujeron software y protocolos para facilitar el correo electrónico y la transferencia de ficheros. Se convirtió en una forma muy rápida, fiable y sencilla de comunicación, y que la gente utilizaba para colaborar en proyectos de investigación. El sistema era distribuido pero su tamaño limitado.

Aunque al principio el énfasis de ARPANET era en el hardware de la red, pronto se comenzó con el desarrollo de software de apoyo. En 1969 se desarrolló el GML (Generalized Markup Language), como un sistema para construir documentos legales portables. En 1986 desarrollaron el SGML (Standard Generalized Markup Language), cuyo objetivo era proporcionar una potente herramienta para definir otros lenguajes de marcado específicos. Pronto SGML dio lugar a HTML (Hipertext Markup Language), XHTML (eXtensible Hipertext Markup Language) y XML (eXtensible Markup Language).

Mientras se desarrollaban los lenguajes de marcado, ARPANET continuó creciendo. En 1973 se añadieron los dos primeros nodos fuera de EEUU. En 1981 ya eran 213 nodos, en 1987 eran más de 10000.

En 1978, Bob Kahn y Vint Cerf y otros miembros de su equipo crearon TCP/IP, el lenguaje común de comunicación de todos los ordenadores de In-

ternet desde 1980. TCP e IP eran protocolos que conectaban redes separadas para formar una red de redes (la "Internet"). Estos protocolos especificaban el entorno de unos servicios básicos que cualquiera necesitaría (transferencia de ficheros, correo electrónico, acceso remoto, ...).

El boom de mediados de los 80 en el mercado de los ordenadores personales y minicomputadores y la disponibilidad de servidores potentes y equipos de red permitió que muchas empresas se unieran a Internet, permitiéndolas comunicarse entre ellas y con sus clientes.

En 1990 Internet contaba ya con más de 300000 nodos y el crecimiento es exponencial hasta nuestros días. Debido a este crecimiento, la habilidad de buscar contenidos a través de Internet se había convertido en una tarea extremadamente difícil.

En 1991 se produjo un hecho que aceleró el desarrolló de la tecnología de la información. Tim Berners-Lee, que trabajaba en el CERN, introdujo el concepto de World Wide Web, que proporcionaba la habilidad de combinar palabras, figuras y sonidos (contenido multimedia) en páginas de Internet. Berners-Lee y sus colaboradores idearon el protocolo HTTP para enlazar documentos Web, el lenguaje HTML para formatear documentos Web, y el sistema URL para localizar documentos Web.

En 1994 se funda el World Wide Web Consortium (W3C), liderado por Tim Berners-Lee. El W3C se compone de individuos y compañías involucradas en Internet y la Web. El propósito del W3C es desarrollar estándares abiertos para que la Web evolucione de forma coherente, en una única dirección.

Mientras el W3C desarrolla estándares abiertos para la Web, las compañías han estado adaptando sus aplicaciones y servidores. Desde 1994, muchas empresas han desarrollado Objetos Distribuidos, que derivó en la creación del estándar CORBA (Common Object Request Broker Architecture), aunque Microsoft se desmarcó de esta propuesta con su modelo DCOM (Distributed Common Object Model). CORBA permitía desarrollar objetos que podían ser invocados remotamente independientemente del lenguaje de programación y

sistema operativo utilizado. Sin embargo, la interoperabilidad conseguida no fue total, quizá debido a la no unión de Microsoft.

Para conseguir una tecnología estándar global sobre componentes distribuidos era necesario encontrar una mínima tecnología común. XML y HTTP ofrecían ese mínimo, y SOAP fue desarrollado para definir el uso de XML y HTTP para acceder a servicios, objetos y servidores de forma independiente de la plataforma utilizada.

En 2001, los objetos interoperables, soportados por IBM y Microsoft, fueron definidos como *Servicios Web* y se podían utilizar junto a una tecnología complementaria llamada UDDI (Universal Discovery Description and Integration). Los Servicios Web permiten publicar componentes y servicios en un directorio que otras aplicaciones Web pueden buscar e implementar mediante la llamada al servicio.

2.2.1.2. Servicios Web

Los servicios Web es la tecnología actual basada en el concepto de SOC más prometedora [122]. Los Servicios Web son aplicaciones autocontenidas y modulares que pueden ser descritas, publicadas, localizadas e invocadas en una red, generalmente la Web. Estas aplicaciones hacen uso de los protocolos del Web y se basan en el estándar XML. Los servicios Web proporcionan la base para el desarrollo de procesos de negocio que están distribuidos por la red y disponibles vía interfaces y protocolos estándar. Pueden usar Internet como medio de comunicación (así como otros protocolos de transporte) y estándares abiertos basados en Internet.

Los tres estándares principales para los servicios Web son SOAP (Simple Object Access Protocol) [121] como medio de comunicación entre servicios, WSDL (Web Services Description Language) [25] para la descripción de servicios y UDDI (Universal Description, Discovery and Integration) [8] como directorio para registro de descripciones de servicios.

A continuación se describen brevemente estos estándares, en especial WSDL por su utilidad para la descripción de servicios.

SOAP

SOAP (Simple Object Access Protocol) [121] es un protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido. Está basado en XML y consta de tres partes: un *envelope* que define un marco para describir qué hay en el mensaje y cómo procesarlo, un conjunto de *reglas de codificación* para expresar instancias de tipos definidos por la aplicación, y una convención para representar llamadas a procedimientos remotos (RPCs) y respuestas. SOAP puede usarse potencialmente en combinación con una variedad de otros protocolos, aunque lo más común es mediante HTTP.

Además de la estructura básica del mensaje, la especificación SOAP define un modelo que impone cómo los receptores deberían procesar los mensajes SOAP. El modelo de mensaje también incluye actores, que indican quién debería procesar el mensaje. Un mensaje puede identificar actores que indican una serie de intermediarios que procesan las partes del mensaje para ellos y pasan el resto. Los mensajes SOAP se asumen que se dirigen de un receptor al siguiente hasta que llegan al receptor final.

SOAP define un *envelope* para transmitir mensajes y reglas para representar llamadas a procedimientos remotos.

Cada mensaje SOAP debe incluir un *cuerpo* y opcionalmente una *cabecera*. La cabecera proporciona información adicional que podría influir en el tratamiento del cuerpo por parte de un receptor, bien el final o intermedio. Tanto la semántica de la cabecera como del cuerpo se deja a la aplicación.

Para gestionar posibles errores que pudieran producirse SOAP proporciona un elemento denominado *fault* que se puede incluir en el cuerpo de un mensaje.

WSDL

WSDL (Web Services Description Language) [25] es un lenguaje basado en XML desarrollado por IBM y Microsoft para describir servicios Web. WSDL intenta separar los servicios, definidos en términos abstractos, de los formatos de datos y protocolos concretos utilizados por la implementación, y define asociaciones entre las descripciones abstractas y sus realizaciones específicas.

Un documento WSDL define los *servicios* como colecciones de *puntos finales (endpoints)* o *puertos*. La definición abstracta de puertos y mensajes está separada de su asociación a una implementación concreta o formato de datos. Esto permite la reutilización de definiciones abstractas: *mensajes*, que son descripciones abstractas de los datos intercambiados, y *tipo de puertos* que son colecciones abstractas de *operaciones*. Las especificaciones concretas de los protocolos y formato de datos para un tipo de puerto particular constituyen *enlaces (binding)* reutilizables. Un puerto se define asociando una dirección de red a un enlace reutilizable, y una colección de puertos definen un servicio. Un documento WSDL usa los siguientes elementos en la definición de servicios:

- *Tipos*. Proporciona definiciones de tipos de datos utilizados para describir los mensajes intercambiados. WSDL reconoce la necesidad de sistemas de tipos para describir los formatos de mensajes, y soporta la especificación de Esquemas XML (XSD) como su sistema de tipos canónico, aunque se podrían utilizar otros.
- *Mensaje*. Representa una definición abstracta de los datos comunicados. Un mensaje se compone de partes lógicas, cada una de ellas asociada con una definición dentro de algún sistema de tipos.
- *Operación*. Una descripción abstracta de una acción soportada por el servicio.
- *Tipo de Puerto*. Un conjunto de operaciones abstractas soportadas por uno o más puntos finales. Cada operación tiene mensajes de entrada y salida. Existen cuatro primitivas de transmisión que puede admitir

un punto final: *One-way* (el punto final recibe un mensaje), *Request-response* (el punto final recibe un mensaje, y responde con otro), *Solicit-Response* (el punto final envía un mensaje, y recibe otro), y *Notification* (el punto final envía un mensaje).

- *Enlaces (Binding).* Definen los detalles sobre protocolos y formato de datos concretos para los mensajes de un tipo de puerto particular.
- *Puerto.* Define un punto final de red individual especificando una dirección única para un enlace.
- *Servicio.* Una colección de puertos relacionados.

WSDL no permite descripción semántica de servicios. Se centra en el mapping de servicios y aunque tiene el concepto de tipos de entrada y salida, no soporta la definición de restricciones entre sus parámetros de entrada y salida. Esta falta de expresividad reduce la capacidad de uso como lenguaje para describir la funcionalidad de los servicios, siendo más apropiado para describir el acceso a los mismos.

UDDI

UDDI (Universal Description, Discovery and Integration) [8] define un estándar para un registro online que permita la publicación y descubrimiento dinámico de servicios Web comerciales. UDDI permite a los programadores y otros representantes de un negocio localizar socios potenciales y formar relaciones en base a los servicios que provean. Por tanto facilita la creación de nuevas relaciones comerciales.

Cada descripción de un negocio en UDDI consta de un elemento *businessEntity*, semejante a un elemento de páginas blancas que describe la información de contacto de un negocio. Un *businessEntity* describe un negocio mediante un nombre, una clave, categorización, servicios ofrecidos (elementos *businessService*) e información de contacto. Un elemento *businessService* describe un

servicio usando un nombre, clave, categorización y múltiples elementos *bindingTemplate*. Esto puede ser considerado como análogo a un elemento de páginas amarillas que categoriza un negocio. Un elemento *bindingTemplate* a su vez describe la clase de acceso que el servicio requiere (teléfono, correo, http, ftp, fax, etc.), valores claves y *tModelInstances*. *tModelInstances* se usan para describir los protocolos, formatos de intercambio que el servicio entiende, es decir, la información técnica requerida para acceder al servicio. También se usa para describir los espacios de nombres para las clasificaciones usadas. Muchos de los elementos son opcionales, incluidos la mayoría de los que serían necesarios para los propósitos de equiparación o composición de servicios.

Usando UDDI, un proveedor de un servicio Web registra sus anuncios junto con palabras claves para su categorización. Un usuario de un servicio Web recupera anuncios del registro basándose en búsqueda por palabras clave. UDDI en sí mismo no soporta descripciones semánticas de los servicios. Por tanto, dependiendo de la funcionalidad ofrecida por el lenguaje de contenido, aunque un agente puede buscar y recuperar las descripciones incluidas en el registro, debe realizarse algún procesamiento adicional externo.

2.2.1.3. Web Semántica

Tim Berners-Lee inventó la Web y constituyó el W3C para desarrollar extender y estandarizar la Web. Pero continuó investigando y comenzó el desarrollo conceptual de la Web Semántica [10]. La Web Semántica pretende ser un paradigma con un impacto similar al de la Web original.

La Web actual está construida sobre HTML, que describe cómo la información debe ser mostrada en una página Web para ser leída por humanos. La Web se ha enfocado hacia los humanos, sin centrarse en que los datos pudieran ser procesados automáticamente.

El objetivo de la Web Semántica es proporcionar una inteligencia interpretable automáticamente que provenga de vocabularios hiperenlazados que los autores usarían para definir sus palabras y conceptos. La idea permite a los agen-

tes software analizar la Web, realizar inferencias más allá de simples análisis lingüísticos realizados por los buscadores actuales (básicamente búsqueda por palabras clave). Hoy en día, los datos disponibles en las páginas Web (HTML) son difíciles de usar a gran escala debido a la ausencia de un esquema global, por lo que no hay un sistema para publicar datos de forma sencilla de procesar por las máquinas. La Web Semántica traerá estructura y contenido definido a la Web, creando un entorno en el que los agentes software puedan llevar a cabo tareas sofisticadas para sus usuarios. Esto presupone que los desarrolladores anotarán sus datos Web mediante lenguajes de marcado avanzados.

Estos lenguajes son conceptualmente más ricos que HTML y permiten la representación del significado y estructura del contenido (interrelaciones entre conceptos), haciendo la Web entendible por los agentes software, abriendo la puerta a una nueva generación completa de tecnologías para el proceso, recuperación y análisis de la información.

Una tecnología clave para el desarrollo de la Web Semántica ya existe hace tiempo: XML. XML permite a cualquiera crear sus propias marcas. Los programas pueden usar esas marcas de forma sofisticada pero el programador tiene que saber cómo el autor usa cada marca. XML básicamente permite a los usuarios añadir una estructura arbitraria a sus documentos, pero no dice nada sobre el significado de esa estructura.

El desarrollo de la Web Semántica permite dotar a los servicios Web de información semántica lo cual facilita la automatización de su descubrimiento, composición e invocación. La Web Semántica va a dirigida a la mejora del entendimiento automático del contenido Web, en donde las ontologías se han identificado como un aspecto clave. Varios lenguajes para representar ontologías se construyen sobre XML (ver figura 2.3), como RDF [97], RDF Schema [98] y OWL [90]. La Web Semántica y los Servicios Web apuntan a ser las tecnologías claves para la siguiente generación de las aplicaciones Web.

A continuación se describen los principales lenguajes utilizados para describir semánticamente contenido Web (RDF y OWL).

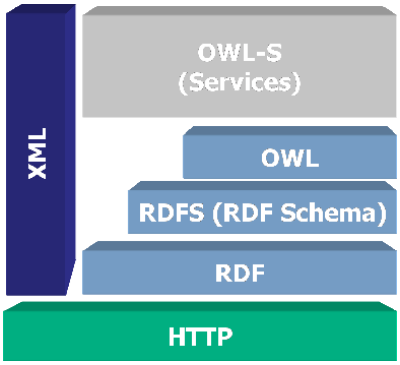


Figura 2.3: Pila de lenguajes en la Web Semántica

RDF/RDFS

RDF (Resource Description Framework) [97] proporciona un medio para añadir semántica a un documento sin hacer ninguna asunción sobre su estructura. El modelo de RDF es equivalente a una red semántica, donde los nodos representan conceptos, instancias de conceptos y valores de propiedades, y los arcos representan propiedades de los conceptos o relaciones entre conceptos. La sintaxis de RDF está definida en XML adaptada para añadir meta-información a los documentos Web.

El modelo de datos de RDF proporciona tres tipos de objetos:

- Un *sujeto* es una entidad a la que se puede referir mediante una URI. Los recursos son los elementos que se describen mediante sentencias RDF.
- Un *predicado* define una relación binaria entre recursos y/o valores atómicos proporcionados por definiciones de tipos de datos primitivos en XML
- Un *objeto* especifica un valor para el sujeto de un predicado. Esto es, los objetos proporcionan las caracterizaciones reales de los documentos Web.

Por ejemplo, en la sentencia "Cervantes escribió El Quijote", "Cervantes" es el sujeto, "escribió" es el predicado y "El Quijote" es el objeto. Y una posible representación en RDF sería:

```
<rdf:Description rdf:about="El Quijote">
       <midominio:autor>Cervantes</midominio:autor>
</rdf:Description>
```

RDF no proporciona primitivas para definir relaciones entre propiedades y recursos (por ejemplo para definir tipos de datos de las propiedades). Esto se resuelve con RDF Scheme (RDFS) [98], cuya combinación con RDF se conoce como RDF(S). Usando RDFS se puede expresar que, por ejemplo, un "Ordenador Personal" es un tipo de "Ordenador", y que "Ordenador" es una subclase de "Máquina". También se pueden crear propiedades y clases, así como rangos y dominios para las propiedades. Los tres conceptos más importantes de RDF(S) son *recurso* ($rdfs : Resource$), *clase* ($rdfs : Class$), y *propiedad* ($rdf : Property$).

OWL

OWL (Web Ontology Language) [90] es un lenguaje para escribir ontologías basado en representación de conocimiento mediante lógicas descriptivas (DL) [5]. OWL se deriva de DAML+OIL y está construido sobre XML, RDF y RDFS proporcionando vocabulario adicional junto con una semántica formal. Proporciona una forma natural de describir clases y relaciones de subclase entre conceptos, así como restricciones en las relaciones entre clases y entre instancias. OWL representa parte de la iniciativa de la Web Semántica para proporcionar semántica a la Web, es decir, hacer los contenidos Web interpretables automáticamente sin ambigüedad.

OWL se compone de tres sub-lenguajes que en orden creciente de poder expresivo son: *OWL Lite*, *OWL DL*, y *OWL Full*.

- *OWL Lite* permite una clasificación jerárquica y restricciones simples. Por ejemplo, aunque incluye restricciones de cardinalidad, sólo permite valores de cardinalidad 0 o 1. Es más sencillo el soporte de una herramienta para OWL Lite que para los otros sublenguajes, y proporciona un camino rápido de migración para otras taxonomías.
- *OWL DL* (Descriptive Logics) proporciona la máxima expresividad pero manteniendo la competitud y decidibilidad computacional. OWL DL incluye todos los constructores del lenguaje OWL, pero sólo pueden usarse bajo ciertas restricciones (por ejemplo, si bien una clase puede ser subclase de muchas clases, una clase no puede ser instancia de otra clase).
- *OWL Full* proporciona la máxima expresividad y libertad sintáctica de RDF pero sin garantías computacionales. Por ejemplo, en OWL Full una clase puede ser tratada simultáneamente como una coleccion de individuos y como un individuo en su propio derecho. OWL Full permite a una ontología aumentar el significado del vocabulario predefinido (RDF o OWL). Es improbable que cualquier software de razonamiento sea capaz de realizar un razonamiento completo para cada característica de OWL Full.

A la hora de adoptar OWL se debe considerar cuál de los sublenguajes se adapta mejor a las necesidades de cada aplicación. La elección entre OWL Lite y OWL DL depende de la medida en la que se requieran los constructores más expresivos de OWL DL. La elección entre OWL DL y OWL Full depende principalmente de en qué medida se necesitan las facilidades de meta-modelado de RDF Schema (por ejemplo, definir clases de clases o asociar propiedades a clases).

OWL Full se puede ver como una extensión de RDF, mientras OWL Lite y OWL DL pueden verse como extensiones de una vista restringida de RDF. Cada documento OWL (Lite, DL, Full) es un documento RDF, y cada documento RDF es un documento OWL Full, pero sólo algunos documentos RDF serán documentos legales OWL Lite u OWL DL. Por esto, hay que tener cuidado cuando se quiere migrar un documento RDF a OWL. Cuando

la expresividad de OWL DL u OWL Lite se estima apropiada, se tienen que tomar algunas precauciones para asegurar que el documento RDF original es compatible con las restricciones adicionales impuestas por OWL DL y OWL Lite. Entre otras, cada URI que es utilizada como nombre de clase debe ser definida explícitamente de tipo $owl : Class$ (igual para las propiedades), cada individuo tiene que pertenecer al menos a una clase (aunque sólo sea a $owl : Thing$), las URIs usadas para clases, propiedades e individuos deben ser mutuamente excluyentes.

2.2.1.4. Servicios Web Semánticos

Los servicios Web extienden la funcionalidad a la actual Web hacia una integración de componentes software distribuidos utilizando estándares de la Web. Sin embargo, las tecnologías utilizadas en los servicios Web (SOAP, WSDL, UDDI) operan a nivel sintáctico y requieren en gran medida de la intervención humana para la programación de aplicaciones que utilicen los servicios Web.

El desarrollo de la Web Semántica está permitiendo dotar a los servicios Web de información semántica, lo cual facilita la automatización de su descubrimiento, composición e invocación.

La Web Semántica y los Servicios Web apuntan a ser las tecnologías claves para la siguiente generación de las aplicaciones Web. La primera se va a dirigida a la mejora del entendimiento automático del contenido Web, en donde las ontologías se han identificado como el componente clave. El objetivo de los servicios Web es permitir la computación distribuida sobre Internet mediante el descubrimiento, composición y ejecución de servicios de forma automatizada y dinámica, proporcionando por lo tanto una nueva tecnología para la ingeniería basada en sistemas Web. La tecnología actual de servicios Web permite el intercambio de mensajes entre servicios Web (SOAP), la descripción de la interfaz para consumir un servicio Web (WSDL), y soporta el anuncio de servicios Web en registros (UDDI). Sin embargo, estas tecnologías no incluyen descripciones explícitas de la funcionalidad de un servicio Web. Además, las descripciones existentes se representan a un nivel sintáctico y por tanto no

representan el significado de la información a intercambiar. Los Servicios Web Semánticos (SWS) aplican tecnologías de la Web Semántica a los servicios Web mediante el uso de descripciones semánticas que permitan la provisión de mecanismos inteligentes para el descubrimiento, composición, contratación, y ejecución de servicios Web.

Actualmente existen dos dos tendencias principales para describir servicios Web semánticos, OWL-S y WSMO, además de alguna propuesta de extensión de WSDL con anotaciones semánticas. A continuación se describen éstas.

OWL-S

OWL-S [77] (anteriormente DAML-S) es una ontología escrita en OWL (anteriormente DAML+OIL) para describir servicios Web. El objetivo de OWL-S es hacer que los servicios Web sean interpretables por un programa, es decir, descritos con suficiente información para permitir, de forma automatizada, el descubrimiento, invocación, composición y monitorización de la ejecución de servicios.

Como ontología OWL, OWL-S tiene todos los beneficios de los contenidos Web descritos en OWL. En concreto, tiene una semántica bien definida, permite la definición de un vocabulario de servicios Web en términos de objetos y las relaciones complejas entre ellos, incluyendo clase, relación de subclase, restricciones de cardinalidad, etc. También incluye todo el formato de tipos de XML.

La ontología, cuyo elemento principal es la clase *Service*, se compone de tres partes o subontologías:

1. *ServiceProfile*: describe qué es lo que hace el servicio. Es similar a una entrada de páginas amarillas para un servicio. Describe las propiedades de un servicio necesarias para el descubrimiento automático, como la funcionalidad que proporciona el servicio, y sus entradas, salidas, precondiciones y efectos.

2. *ServiceModel*: describe el modelo de procesos del servicio (flujo de control y de datos involucrados en el uso del servicio), es decir, cómo se ejecuta el servicio. Está diseñado para permitir composición, ejecución y monitorización automática de servicios.

3. *ServiceGrounding*: especifica los detalles de cómo se accede al servicio. Típicamente se especificará un protocolo de comunicación, formatos de los mensajes, y otros detalles específicos del servicio como el número de puerto para contactar con el servicio, etc.

Un mismo servicio puede tener varios *profiles* y *groundings* pero un solo *service model*.

Service Profiles. El *service profile* describe un servicio como función de tres tipos básicos de información: información sobre el proveedor del servicio, qué función realiza el servicio, y un conjunto de propiedades (perfil) que describen características del servicio.

La información sobre el proveedor consiste en información de contacto sobre la entidad que proporciona el servicio. Esta información se describe con las propiedades *serviceName* (nombre identificador del servicio), *textDescription* (breve descripción textual del servicio) y *contactInformation* (información sobre el mecanismo de contacto con los responsables del servicio). Sólo puede haber un nombre y descripción textual por servicio pero sí puede haber más de una información de contacto.

La descripción de la funcionalidad del servicio se expresa en términos de la transformación producida por el servicio. Se especifican las entradas y salidas, además de las condiciones externas necesarias y efectos esperados de la ejecución del servicio. Por ejemplo, en un servicio de venta de productos, las entradas serían un número de tarjeta de crédito y fecha de caducidad, la precondición que el número de tarjeta es válido, la salida generada puede ser un recibo y los efectos son que la tarjeta es cargada con el precio de la compra y el producto es entregado al comprador. Para expresar la funcionalidad, en la

ontología existen las siguientes propiedades de la clase *Profile*: *hasParameter*, *hasInput*, *hasOutput*, *hasPrecondition*, *hasResult*.

Por último, el perfil del servicio permite la descripción de un conjunto de propiedades que se utilizan para describir características del servicio. Esta información puede incluir su posible clasificación en alguna categoría dentro de alguna taxonomía de servicios y, en general, cualquier parámetro que el servicio pudiera querer especificar como calidad del servicio, máximo tiempo de respuesta, disponibilidad geográfica del servicio, etc. Las propiedades disponibles en OWL-S son: *serviceParameter* (lista de propiedades que describen el servicio mediante las propiedades *serviceParameterName* y *sParameter* de la clase *ServiceParameter*) y *serviceCategory* (cuyo valor es una instancia de la clase *ServiceCategory* con propiedades categoryName, *taxonomy*, *value*, *code*).

Service Model. Para dar una perspectiva detallada de cómo funciona el servicio, éste se puede ver como un proceso (clase *Process*). Este modelo se basa en el trabajo existente en varios campos como los relacionados con planificación y modelado de procesos. Un proceso puede tener entradas, salidas, precondiciones y efectos. Las salidas y efectos pueden tener condiciones. La clase *Process* tiene las propiedades *hasParameter*, *hasInput*, *hasOutput*, *hasPrecondition* y *hasResult*, cuyos valores son de las clases *Parameter*, *Input*, *Output*, *Condition* y *Result*, respectivamente. En la actualidad OWL-S deja libertad en el lenguaje utilizado para representar condiciones.

En el modelo de procesos se identifican tres tipos de procesos: atómicos, simples y compuestos. Los procesos *atómicos* son invocables directamente (pasando los mensajes correspondientes). No tienen subprocesos y se ejecutan en un único paso, desde la perspectiva del usuario.

Los procesos *simples* no son invocables directamente pero, como los atómicos, se conciben como que se ejecutan en un único paso. Los procesos simples se usan como medio de abstracción pudiendo proporcionar una vista abstracta de un proceso atómico, o una representación simplificada de un proceso compuesto.

Los procesos *compuestos* se componen de subprocesos, que a su vez pueden ser atómicos, simples o compuestos. La descomposición de los procesos se puede especificar con los constructores de control especificados en la tabla 2.1. Tal descomposición normalmente muestra, entre otras cosas, cómo se conectan las entradas y salidas entre subprocesos.

Costructor	Descripción
Sequence	Ejecuta una lista de procesos en orden
Split	Ejecuta un grupo de procesos de forma concurrente
Split+Join	Ejecuta un grupo de procesos de forma concurrente con sincronización
Any-Order	Permite la ejecución de un grupo de procesos en un orden no especificado o concurrentemente
Choice	Elige entre alternativas
If-Then-Else	Si se cumple cierta condición, ejecutar Then, si no ejecutar Else
Iterate	Repite la ejecución de un grupo de procesos mientras se cumple cierta condición
Repeat-Until	Repite la ejecución de un grupo de procesos hasta que se cumple cierta condición
Repeat-While	Repite la ejecución de un grupo de procesos mientras se cumple cierta condición

Tabla 2.1: Constructores de procesos compuestos en OWL-S

Service Grounding. Especifica los detalles de cómo un programa o agente puede acceder al servicio, relacionados con protocolos y formatos de mensajes, serialización, transporte y direccionamiento. Es un mapping de una especificación de servicio abstracta a una concreta. OWL-S propone utilizar WSDL para este propósito (figura 2.4).

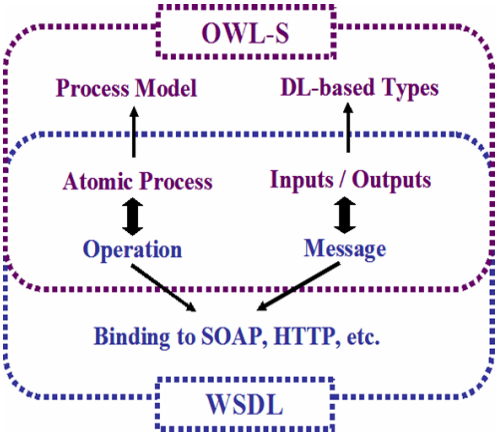


Figura 2.4: Uso de WSDL para especificar el grounding en OWL-S

WSMO

WSMO (Web Service Modeling Ontology) [29, 38, 100] es una ontología para describir los aspectos relevantes de los servicios Web con el fin de facilitar la automatización de las tareas (p.e. descubrimiento, selección, composición, mediación, ejecución, monitorización, etc.) involucradas en la integración de servicios Web. WSMO tiene su base conceptual en WSMF (Web Service Modeling Framework) [37], el cual refina y extiende, y desarrolla una ontología y un conjunto de lenguajes formales.

WSMO trata de seguir las tecnologías Web actuales. Cada recurso se especifica de forma independiente sin considerar su posible uso o interacciones con otros recursos, siguiendo la naturaleza abierta y distribuida de la Web. Utiliza la mediación como principio para manejar las heterogeneidades que surgen en entornos abiertos. WSMO diferencia claramente entre los deseos de los usuarios (o clientes) y los recursos disponibles. WSMO busca proporcionar un modelo descriptivo apropiado, y ser compatible con las tecnologías emergentes.

WSMO identifica cuatro conceptos principales para definir servicios Web semánticos: *Ontologías*, *Servicios Web*, *Objetivos* y *Mediadores*. Utiliza MOF [86] para especificar el metamodelo.

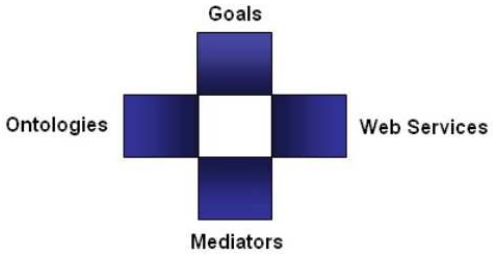


Figura 2.5: Elementos principales de WSMO

Ontologías. Proporcionan la semántica formal para la terminología utilizada por otros elementos de WSMO. WSMO recomienda el uso de WSML [30], un lenguaje formal específicamente diseñado para expresar descripciones semánticas de acuerdo al metamodelo de WSMO, aunque se podrían utilizar otros lenguajes.

Se pueden definir *propiedades no funcionales* para caracterizar las ontologías. Se usan principalmente para describir aspectos como el creador, fecha, descripciones en lenguaje natural, versión, etc. Las *ontologías importadas* permiten un enfoque modular para el diseño de la ontología y se pueden usar siempre que no sea necesario resolver conflictos entre las ontologías. Cuando se importan ontologías en escenarios reales, son necesarias algunas operaciones de alineamiento, mezcla y transformación de las ontologías importadas para resolver desajustes. Por esta razón se usan *mediadores* (ooMediators). Los *conceptos* constituyen los elementos básicos de la terminología acordada para un problema concreto. Para modelar interdependencias entre conceptos (respectivamente instancias de estos conceptos) se usan las *relaciones*. Casos especiales de relaciones son las *funciones*, que tienen rango unario y dominio n-ario (parámetros heredados de la relación), donde el valor del rango depende funcionalmente de valores del dominio. Por último, las *instancias* se definen explícitamente o mediante un enlace a un almacén de instancias externo.

Servicios Web. Representan entidades computacionales capaces (mediante invocación) de proveer acceso a servicios que, a su vez, proporcionan algún

valor en un dominio. Las descripciones de servicios Web constan de las capacidades, interfaces y funcionamiento interno del servicio. De este modo un servicio Web podría proporcionar diferentes servicios, como por ejemplo Amazon[6], que puede usarse para comprar libros o para obtener el ISBN de un libro.

Un servicio Web se define mediante los siguientes elementos: propiedades no funcionales, ontologías importadas, mediadores, capacidad e interfaces.

Las *propiedades no funcionales* y las *ontologías importadas* juegan un rol similar al de la descripción de las ontologías. En cuanto a los *mediadores*, se incluye un tipo adicional de mediador para tratar adaptaciones necesarias relativas a protocolos y procesos.

Los últimos dos atributos son los que propiamente definen semánticamente los servicios Web. Una *capacidad* describe funcionalmente un servicio, describiendo restricciones sobre la entrada y salida de un servicio mediante las nociones de pre-condiciones, asunciones, post-condiciones y efectos. Por su parte las *interfaces* especifican cómo se comporta el servicio para alcanzar su funcionalidad. Una interfaz se compone de una coreografía (choreograpy) que describe la interfaz para la interacción requerida para invocar el servicio, y una instrumentación (orchestration) que describe cómo se alcanza la funcionalidad mediante la agregación de otros servicios Web.

Objetivos. Los objetivos describen aspectos relativos a los deseos del usuario respecto a la funcionalidad y comportamiento requerido. Se pueden entender como descripciones de servicios Web que potencialmente satisfacerían los deseos del usuario. Se usan ontologías para definir la terminología usada en la especificación de los objetivos.

Los elementos que forman una descripción de un objetivo son los mismos que los de la descripción de un servicio Web.

[6] http://www.amazon.com/

Mediadores. Describen elementos para resolver incompatibilidades entre elementos de WSMO a nivel de datos (terminologías), procesos (combinación de servicios) y protocolos (comunicación entre servicios). Un mediador WSMO conecta elementos y proporciona facilidades de mediación para resolver incompatibilidades. Los elementos de un mediador son:

- *propiedades no funcionales*
- *ontologías importadas*, especialmente las que definen lenguajes para mediación
- *origen*: denota los recursos heterogéneos que son conectados mediante el mediador
- *destino*: elemento que recibe los componentes mediados.
- *servicio de mediación*: define la facilidad de mediación. El enlace con el servicio de mediación utilizado por un mediador específico puede definirse de tres maneras: directamente, mediante un objetivo que especifique la mediación deseada o a través de otro mediador.

Existen diferentes tipos de mediadores para conectar los distintos elementos WSMO: *OO Mediators* conectan y median ontologías heterogéneas, *GG Mediators* conectan objetivos, *WG Mediators* enlazan servicios Web a objetivos, y *WW Mediators* resuelven incompatibilidades entre servicios Web.

Extensiones semánticas de WSDL

WSDL-S [2] es una propuesta para añadir semántica a los servicios Web proporcionando una extensión sencilla de WSDL. La extensión proporciona un mecanismo para especificar precondiciones y efectos sobre las operaciones de las interfaces WSDL. WSDL-S describe un mecanismo para ligar un modelo semántico del servicio Web con las descripciones funcionales sintácticas capturadas por WSDL. Usando los elementos de extensibilidad de WSDL, se pueden

crear un conjunto de anotaciones para describir semánticamente las entradas, salidas y operaciones de un servicio Web. Este método mantiene el modelo semántico fuera de WSDL, haciendo el enfoque independiente de cualquier lenguaje de representación de la ontología como se muestra en la figura 2.6.

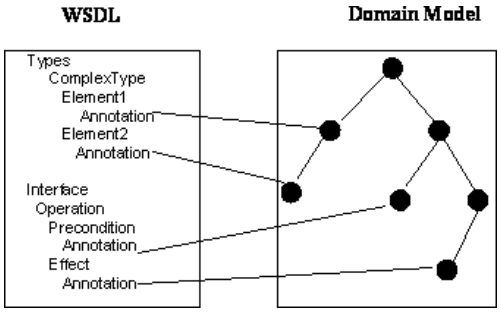


Figura 2.6: Asociación de semántica a los elementos de WSDL

Las principales ventajas de este enfoque son que (i) se construye sobre WSDL usando su mecanismo de extensión, (ii) es agnóstico al lenguaje ontológico usado para las anotaciones semánticas y (iii) permite anotaciones de tipos de datos XML Schema.

Desarrollado en el proyecto METEOR-S[7] y tomando parte de las ideas de WSDL-S, surgió la recomendación del W3C SAWSDL (Semantic Annotations for WSDL and XML Schema) [36], también como propuesta de extensión de WSDL.

SAWSDL define mecanismos usando anotaciones semánticas que pueden añadirse a los componentes WSDL. SAWSDL no especifica un lenguaje para representar modelos semánticos (ontologías). En su lugar, proporciona mecanismos para que conceptos de los modelos semánticos que se definen dentro o fuera del documento WSDL puedan ser referenciados dentro de los componentes WSDL como anotaciones.

SAWSDL tiene dos tipos básicos de anotación, el modelo de referencia y el esquema de mapping. El modelo de referencia, el mismo que el de WSDL-S,

[7] http://lsdis.cs.uga.edu/projects/meteor-s/

se usa para asociar tipos de puertos/interfaces, operaciones, entradas, salidas, y elementos y atributos XML schema con conceptos semánticos. Las anotaciones del modelo de referencia se usan en METEOR-S para el descubrimiento de servicios Web y la configuración dinámica de procesos Web. Las anotaciones del esquema de mapping, de nuevo similar en principio a la versión de WSDL-S, se usan en METEOR-S para tratar con la heterogeneidad de datos, particularmente transformando datos de una representación en otra, de forma que se puedan usar en otro servicio Web.

2.2.2. Similitud entre conceptos

En esta sección se repasan algunos de los trabajos más representativos existentes en el campo de la similitud entre conceptos. Esto es, dados dos conceptos de una ontología determinar mediante una medida numérica cómo de parecidos son dichos conceptos. Esta medida puede ser útil a la hora de determinar si un servicio (descrito posiblemente mediante un lenguaje que hace uso de conceptos de una ontología) es similar y en qué grado a otro buscado.

Algunos autores definen la *similitud* mientras que otros prefieren usar la *distancia* entre conceptos. Ambas medidas son inversamente proporcionales entre ellas y normalmente se pueden calcular como la inversa de la otra (a mayor distancia menor similitud y viceversa).

A continuación se comenzará presentando propuestas sobre la similitud de conceptos simples, para a continuación presentar algunas propuestas sobre conceptos compuestos.

2.2.2.1. Conceptos Simples (o Atómicos)

En este apartado se describen algunos de los enfoques propuestos en la literatura para medir la similitud entre dos conceptos simples.

Una de las medidas de *distancia* entre conceptos más conocida es la *longitud del camino más corto* entre ellos en la taxonomía, propuesto por Rada et

al. [96]. Puesto que ambos conceptos podrían no estar en la misma rama del árbol de la taxonomía, se puede calcular como la suma de la longitud desde cada concepto a su subsumidor menos común (lcs)[8].

$$dist(c_1, c_2) = depth(c_1) + depth(c_2) - 2 \times depth(lcs(c_1, c_2)) \tag{2.1}$$

Leacock y Chodorow [70] definen la similitud entre dos términos como la *relatedness*, que definen como una función inversa de la distancia semántica.

$$relatedness(t_1, t_2) = -\log \frac{dist(t_1, t_2)}{2D} \tag{2.2}$$

donde $dist(t_1, t_2)$ es la misma que en (2.1), y D es la máxima profundidad de la taxonomía.

Otras propuestas más refinadas también consideran la *profundidad de los conceptos en la taxonomía*, lo que tiene sentido bajo la asunción de que los conceptos en niveles superiores del árbol tienen semánticas más generales y menos similitud entre ellos, mientras que los conceptos de los niveles inferiores tienen semánticas más concretas y por tanto mayor similitud.

En esta línea, Wu y Palmer [131] usan la terminología *score* para definir la similitud entre dos términos de la siguiente manera:

$$score(t_1, t2) = \frac{2N_3}{N_1 + N_2 + 2N_3} \tag{2.3}$$

Donde N_1 y N_2 son las longitudes del camino más corto desde t_1 and t_2 (respectivamente) al su lcs, y N_3 es la longitud del camino más corto desde el lcs a la raíz.

Li et al. [74] definieron la similitud entre dos conceptos como:

[8] *least common subsumer (lcs)*, también conocido como el *antecesor más específico* (*most specific ancestor (msa)*)

$$sim(c_1, c_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & si\ c_1 \neq c_2 \\ 1 & otherwise \end{cases} \tag{2.4}$$

donde $\alpha \geq 0$ y $\beta \geq 0$ son parámetros para ponderar la contribución de la longitud del camino más corto (l) entre los dos conceptos y la profundidad del *lcs* (h) en la jerarquía de conceptos, respectivamente.

Aunque las medidas basadas en la distancia entre conceptos en una taxonomía son las más comunes existen otros enfoques que no se basan en esa medida.

Tversky [118] propone un enfoque en el que un concepto C es caracterizado mediante un conjunto de características, $ftrs(C)$. Introdujo dos clases de medidas:

1. *contrast model*

   $$contrast(C, D) = \theta f(ftrs(C) \cap ftrs(D)) - \alpha f(ftrs(C) \setminus ftrs(D)) - \beta f(ftrs(D) \setminus ftrs(C)) \tag{2.5}$$

   donde $\setminus$ es la diferencia de conjuntos, θ, α y β son constantes no negativas, y $f(\cdot)$ es normalmente el número de características en el conjunto. En definitiva, esta medida es una generalización del número de características comunes menos el número de las no comunes.

2. *ratio model*

   $$sim(C, D) = \frac{f(ftrs(C) \cap ftrs(D))}{f(ftrs(C) \cap ftrs(D)) + \alpha f(ftrs(C) \setminus ftrs(D)) + \beta f(ftrs(D) \setminus ftrs(C))} \tag{2.6}$$

   Cuando no es deseada la asimetría de la medida de similitud, se puede tomar $\alpha = \beta = 0.5$, y bajo la asunción de que f sea distributiva sobre conjuntos disjuntos ($f(V \cup W) = f(V) + f(W)$), la similitud suele tomarse como:

   $$dist(C, D) = \frac{2 \times f(ftrs(C) \cap ftrs(D))}{f(ftrs(C)) + f(ftrs(D))} \tag{2.7}$$

Resnick [99] propone un modelo basado en el *contenido de información (IC)* en el que existe información sobre la probabilidad de que un individuo esté descrito por un concepto C. Usa el $lcs(c_1, c_2)$ como representante de la similitud entre c_1 y c_2, y propone usar el *contenido de información* como medida de similitud:

$$sim(c_1, c_2) = IC(lcs(c_1, c_2)) = -\log pr(lcs(c_1, c_2)) \tag{2.8}$$

Este enfoque tiene la ventaja de que no se ve afectado por cambios en la jerarquía.

Jiang y Conrath [63] refinan la medida de Resnick:

$$sim(c_1, c_2) = IC(c_1) + IC(c_2) - 2 \times IC(lcs(c_1, c_2)) \tag{2.9}$$

Lin [75] propuso:

$$sim(c_1, c_2) = \frac{2 \times IC(lcs(c_1, c_2))}{IC(c_1) + IC(c_2)} \tag{2.10}$$

Borgida et al. [13] aplican algunos de los enfoques anteriores a una lógica descriptiva muy simple (DL A), que sólo permite conjunciones en la definición de conceptos. En particular:

- Enfoque de características: toman los conceptos atómicos como características.
- Enfoque basado en distancias: se utiliza el camino entre los dos conceptos
- Enfoque de contenido de información: aquí no necesitan ninguna adaptación ya que se usan los conceptos tal cual.

Di Noia et al.[85] también se enfocan en lógicas descriptivas y proponen una función para lo que denominan *potential match* entre dos servicios (algunos

requisitos en la demanda D no están especificados en el servicio S). La función de ranking cuenta (0 es lo mejor):

- el número de nombres de conceptos en D que no están en S
- el número de restricciones de D que no se infieren por las de S
- añade recursivamente el resultado de la función para cada rol cuantificado universalmente en D

Fanizzi y d'Amato [35] definen una medida de similitud entre conceptos en DL ALN. Descomponen la forma normal de las descripciones de conceptos y miden la similitud de los subconceptos:

- Conceptos primitivos: proporción de individuos comunes respecto al número de individuos de ambos conjuntos.
- Restricciones de valor: se calculan recursivamente y se toma la media.
- Restricciones numéricas: proporción de solapamiento entre los dos intervalos y el intervalo mayor (cuyos extremos son el mínimo y máximo de ellos), se toma la media.

En el enfoque de equiparación de servicios Web semánticos OWLS-MX [67], se utiliza un razonamiento lógico que se complementa con técnicas de cálculo de similitud provenientes del campo de Recuperación de Información (IR). En particular, permiten cuatro métricas diferentes basadas en cadenas de tokens: la del *coseno*, la *pérdida de información*, la *extended Jacquard* y la *Jensen-Shannon information divergence*. Estas métricas se aplican a conceptos desdoblados, es decir la expresión desdoblada *(and C (and B (and A)))* corresponde al concepto C ($C \sqsubseteq B \sqsubseteq A$).

La tabla 2.2 resume los diferentes enfoques sobre similitud de conceptos descritos en esta sección.

	Enfoque	Foco	Sim	Dis	Prof.	Medida	rango
Rada et al.	estructural	conc.	sí	sí	no	distancia	0..2H
Leacock-Chodorow	estructural	conc.	sí	sí	no	similitud	0..∞
Wu-Palmer	estructural	conc.	sí	sí	sí	similitud	0..1
Li et al.	estructural	conc.	sí	sí	sí	similitud	0..1
OWLS-MX	estructural	conc.	sí	sí*	sí*	similitud	0..1
Tversky-contrast	caract.	conc.	sí	no	no	similitud	0..N
Tversky-ratio	caract.	conc.	sí/no	no	no	similitud	0..1
Resnick	estructural	inst.	sí	no	no	similitud	0..∞
Jiang-Conrad	estructural	inst.	sí	no	no	distancia	0..∞
Lin	estructural	inst.	sí	no	no	similitud	0..1
Borgida et al.-car.	DL A	conc.	sí	no	no	similitud	0..1(N)
Borgida at al.-estr.	DL A	conc.	sí	sí	no	distancia	0..2H
Borgida et al.-IC	DL A	instancia	sí	no	no	dist/sim	0..∞(1)
Di Noia et al.	DL	conc.	no	no	no	distancia	≥ 0
Fanizzi-d'Amato	DL ALN	conc.	sí	no	no	similitud	0..1

Tabla 2.2: Resumen de enfoques sobre similitud entre conceptos

La primera característica destacada es si se usa el árbol de la taxonomía (*estructural*), un modelo basado en características (*caract.*) o si se aplica a lógicas descriptivas (*DL*) (y qué lenguaje DL).

La mayoría de los enfoques descritos aquí se aplican a definiciones de *conceptos* pero otros basan sus medidas de similitud en el número de *instancias* de conceptos, que se ve menos afectado por cambios en la taxonomía.

Aunque la *simetría (Sim)* ha sido definida por varios autores como una propiedad de las funciones de similitud no todos los enfoques la cumplen. Obsérvese que, dependiendo de la aplicación, podría desearse una medida asimétrica. Este es el caso de la equiparación de servicios semánticos donde, por ejemplo, es importante distinguir si un concepto de entrada en una consulta subsume o es subsumido por un concepto de entrada en un anuncio de servicio (esto determina si el servicio puede, al menos, ser invocado). En este sentido, obsérvese que el enfoque Tversky-ratio permite ambas opciones dependiendo de los valores de algunos parámetros en la función de similitud.

La *distancia (Dis)* entre conceptos en la taxonomía es el principal parámetro usado por los enfoques estructurales (incluyendo el de Borgida et al. DL basado en la función de Rada). Esta medida tiene sentido bajo la asunción de que las instancias están equidistribuidas entre los conceptos; de lo contrario pares de conceptos en una parte de la ontología especificada con mucho detalle serían puntuados con una similitud más baja que conceptos a la misma distancia en otra parte de la taxonomía. En el caso de DL, la distancia es difícil de ser usada a menos que se adopte algún tipo de "representación canónica" (lo cual es difícil de encontrar para DL expresivas).

Como se mencionó anteriormente, frecuentemente se asume que en una taxonomía jerárquica, los nodos en niveles superiores representan conceptos más generales (menos similares semánticamente), mientras que los niveles inferiores contienen conceptos más específicos. Por esta razón, algunos de los enfoques también tienen en cuenta la profundidad de los conceptos en la taxonomía. Sin embargo, esto asume que, en general, las instancias están equidistribuidas entre los conceptos.

Obsérvese que, la forma en la que OWLS-MX aplica las técnicas de IR para desdoblar nombres de conceptos, indirectamente usa la distancia y profundidad de los conceptos, pero también podría verse como una clase de similitud basada en características.

También se detalla en la tabla si las diferentes propuestas definen una función de *similitud* o una función de distancia. Aunque ambas medidas se pueden obtener fácilmente a partir de la otra (por ej. $sim = \frac{1}{dist}$) se ha preferido mantener su definición original, ya que varían en el *rango* del valor devuelto (última columna). Es conveniente que el rango de valores sea unificado para facilitar la agregación de valores de similitud en caso de expresiones complejas que involucren varios conceptos.

2.2.2.2. Conceptos Compuestos

Rada et al. también extendieron la definición de *distancia* para manejar conceptos compuestos representados por un conjunto de conceptos. Los conceptos en esos conjuntos se pueden interpretar como conjunciones o disyunciones. En el caso de una disyunción de conceptos la distancia se define como:

$$dist(C_1 \sqcup \ldots \sqcup C_k, C) = min_i dist(C_i, C) \tag{2.11}$$

donde C_i y C representan conceptos (elementales o compuestos). Cuando C es a su vez un concepto disyuntivo, se aplica la misma función ($dist(C_i, C)$).

En el caso de conjunción entre conceptos se aplica la siguiente función de distancia:

$$dist(V_1, V_2) = \begin{cases} 0 & si\ V_1 = V_2 \\ \\ \frac{1}{|V_1||V_2|} \sum_{u \in V_1} \sum_{v \in V_2} dist(u, v) & \text{en otro caso} \end{cases} \tag{2.12}$$

donde V_1 y V_2 son conjuntos que representan conceptos compuestos que con-

sisten en una conjunción ($\sqcap$) de sus elementos, y $dist(u, v)$ es la longitud del camino más corto entre los nodos u y v.

Ehrig et al. [34] analizan tres niveles sobre los que se puede medir la similitud entre conceptos: datos, ontología y contexto. Aquí nos interesa el nivel de la ontología. Usan la función propuesta por Li et al. [74] en caso de similitud entre conceptos. También proponen la siguiente fórmula (coseno) para calcular la similitud entre dos conjuntos de conceptos:

$$sim(E, F) = \frac{\sum_{e \in E} \vec{e} \cdot \sum_{f \in F} \vec{f}}{\left| \sum_{e \in E} \vec{e} \right| \cdot \left| \sum_{f \in F} \vec{f} \right|} \tag{2.13}$$

con $E = \{e_1, e_2, \ldots\}$, $\vec{e} = (sim(e, e_1), sim(e, e_2), \ldots, sim(e, f_1), sim(e, f_2), \ldots)$, y análogo para F y $\vec{f}$, respectivamente.

Sierra y Debenham [110] definen la similitud semántica entre dos fórmulas lógicas φ y ψ como la similitud maxmin ente los conjuntos de conceptos ($O(\cdot)$) que aparecen en las fórmulas:

$$sim(\varphi, \psi) = \max_{c_i \in O(\varphi)} \min_{c_j \in O(\psi)} \{sim(c_i, c_j)\} \tag{2.14}$$

2.2.3. Descripción y equiparación de servicios

En este apartado se presentan algunos de los sistemas más representativos para la descripción y equiparación de servicios. Los tres primeros (InfoSleuth, Retsina/Larks e IMPACT) son sistemas basados en agentes que incluyen técnicas para la descripción y equiparación de servicios. Los siguientes, más recientes, se centran en equiparación de servicios Web semánticos.

2.2.3.1. InfoSleuth

Es un sistema de recuperación de información de forma distribuida en entornos abiertos y cambiantes. En InfoSleuth [84] los agentes cooperan para conseguir información. Existen varios roles que pueden jugar los agentes en el sistema. Especialmente interesantes son los llamados *broker agents*. Estos agentes se encargan de la realización de la equiparación (matchmaking), tanto a nivel sintáctico como semántico, entre agentes que proporcionan servicios y los que los solicitan. Los anuncios y peticiones de servicios se realizan en términos de ontologías compartidas entre los agentes. Las ontologías se refieren a diversos aspectos a tener en cuenta en la descripción de los servicios: (1) las conversaciones utilizadas en la comunicación, (2) el lenguaje de interfaz para el servicio (p. ej. sql), (3) la descripción semántica del servicio y (4) el dominio de la información que trata (p. ej. medicina). La información sobre las capacidades de los agentes se define mediante un conjunto de fragmentos de ontologías: una parte de la ontología suficiente para describir algún aspecto de la funcionalidad de un agente. Un anuncio contiene una o más descripciones de servicios, mientras que una consulta se refiere solamente a un servicio. Cada descripción de servicio se compone de la conjunción de varios fragmentos de ontologías. La figura 2.7 muestra un ejemplo de descripción de servicio en InfoSleuth.

El mecanismo de equiparación se realiza en un proceso descendente (top-down):

- Equiparación de Consultas con Anuncios. Al nivel más alto, el conjunto de fragmentos de ontologías de una consulta se interpreta como una conjunción. Por tanto, un servicio anunciado equipara con esta consulta si incluye en su descripción al menos todos los fragmentos de ontología de la consulta y, además, cada fragmento individual equipara con el correspondiente fragmento del anuncio.

- Equiparación de Fragmentos de ontología. En primer lugar, los nombres de los fragmentos deben ser idénticos. Además, las clases anunciadas deben equiparar con las de la consulta, pero aquí la consulta tiene la opción

```
Advertisement
  capability env-subscription
    ontology conversation
      class conversation
        slotconversation-type in {subscribe}
        slotlanguage in {kqml}
    ontology sql
    ontology services
      class data-response
        slot language in {tuple-format}
        slot delivery in {http,inline}
      class subscription
        slot computation in {direct}
    ontology environment
      class site
        slot contaminant
        slot remedial-response
        slot city
        slot state in {Texas}
```

```
Query
  capability subscribe-to-capability
    ontology conversation
      class conversation
        slot conversation-type
             in {subscribe}
        slot language in {kqml}
    ontology sql
    ontology services
      class data-response
        slot language in {tuple-format}
        slot delivery in {inline}
      class subscription
        slot computation in {direct}
    ontology environment
      class site
        slot contaminant
        slot city in {Austin}
        slot state in {Texas}
```

Figura 2.7: Ejemplo de anuncio y consulta en InfoSleuth [23]

de especificar si su semántica es mediante conjunción o disyunción. Si es una conjunción entonces el fragmento de la ontología en el servicio anunciado debe contener por lo menos todas las clases del fragmento de la consulta y, además, esas clases deben equiparar. En el caso de una disyunción, es suficiente con que una de las clases de la consulta equipare con alguna de las anunciadas.

- Equiparación de Clases de la ontología. Una clase puede tener uno o más atributos. A este nivel, la consulta también puede especificar si estos atributos se interpretan como conjunción (todos los atributos de la consulta deben equiparar con algún atributo del anuncio) o disyunción (al menos uno). El primer criterio a seguir es que las clases tengan el mismo nombre. Sin embargo, puesto que las ontologías suelen tener jerarquías de clases, los nombres también equiparan si la clase del anuncio es una subclase de la clase de la consulta. Cada clase especificada en una consulta tiene una profundidad de inferencia que indica el máximo número de relaciones clase-subclase a atravesar (0 para nombres idénticos, infinito para indicar que no hay límite).

- Equiparación de Atributos. Debe tener el mismo nombre y una intersección no vacía entre los posibles valores del atributo en el anuncio y en la consulta.

En InfoSleuth la mediación no siempre se realiza mediante un único broker, sino que permite distribuir el proceso de equiparación entre múltiples brokers colaboradores, cada uno de ellos representando a un conjunto de agentes. Los agentes deben anunciar su servicios por lo menos a un broker. Cuando un agente solicita un servicio a un broker, éste no sólo consulta su base de conocimiento de anuncios de servicios sino que también puede consultar a otros brokers.

2.2.3.2. RETSINA/LARKS

En la plataforma RETSINA [114], el lenguaje utilizado para la descripción de servicios se llama LARKS [115]. El conocimiento del dominio se especifica mediante ontologías escritas en un lenguaje de conceptos (ITL). Tanto los anuncios como las peticiones de servicios se expresan con el mismo lenguaje. Una especificación de servicio en LARKS es un marco con los siguientes atributos:

- *Context*: palabras clave que describen el dominio de la descripción.
- *Types*: definición opcional de tipos de datos usados en la especificación.
- *Input y Output*: declaración de parámetros de entrada y salida
- *InConstraints y OutConstraints*: restricciones lógicas (cláusulas de Horn) sobre los parámetros de entrada/salida (precondiciones y postcondiciones).
- *ConcDescriptions*: descripción opcional de conceptos que aparecen en la especificación.
- *TextDescription*: descripción textual opcional.

La figura 2.8 muestra un ejemplo de descripción de servicio en LARKS.

LARKS es bastante expresivo y permite realizar inferencia automática. El proceso de equiparación de servicios utiliza diferentes técnicas provenientes de campos como recuperación de información, IA e ingeniería del software para calcular la similitud sintáctica y semántica entre descripciones y solicitudes de servicios.

La equiparación se realiza mediante cinco filtros, cada uno de los cuales es independiente de los demás y que van reduciendo el conjunto de candidatos con respecto a un criterio. Los filtros van creciendo en restrictivos y en complejidad computacional. A continuación se describen brevemente los cinco filtros.

```
Context          Computer*Computer;
Types            InfoList: ListOf (model: Model*ComputerModel,
                 brand: Brand*Brand,
                 price: Price*Money, color: Color*Colors);
Input            brands: SetOf Brand*Brand;
                 areas: SetOf State;
                 processor: SetOf CPU*CPU;
                 priceLow*LowPrice: Integer;
                 priceHigh*HighPrice: Integer;
Output           Info: InfoList;
InConstraints
OutConstraints   sorted(Info).
ConcDescriptions Computer = (and Product (exists has-processor CPU)
                 (all has-memory Memory) (all is-model ComputerModel));
                 LowPrice=(and Price (ge 1800)
                              (exists in-currency aset(USD)));
                 HighPrice=(and Price (le 50000)
                              (exists in-currency aset(USD)));
                 ComputerModel=aset(HP-Vectra,PowerPC-G3,Thinkpad770,Sat315);
                 CPU = aset(Pentium,K6,PentiumII,G3,Merced)
                 [Product, Colors, Brand, Money]
```

Figura 2.8: Ejemplo de servicio para búsqueda de información sobre ordenadores

1. *Equiparación de Contexto*. Se comprueba que las dos especificaciones tienen el mismo o similar contexto, es decir, si las distancias de las raíces de las palabras no superan un umbral, y las distancias entre los conceptos asociados no superan un umbral.

2. *Equiparación de Perfiles*. En esta fase la especificación se considera un documento y se aplica una técnica estándar de recuperación de información, llamada TF-IDF [101]. La similitud se calcula como:

$$dps(Req, Ad) = \frac{Req \bullet Ad}{|Req| \cdot |Ad|}$$

   donde $\bullet$ representa el producto escalar de los vectores de términos ponderados. Si el valor supera un umbral pasa el filtro.

3. *Equiparación de Similitud*. Se basa en una combinación de las distancias calculadas para pares de declaraciones de entrada y salida, y restricciones de entrada y salida. La similitud total es la media de la similitud de todos los pares de declaraciones y restricciones.

4. *Equiparación de Signatura*. Este filtro considera las partes declarativas de la solicitud y el anuncio, y determina si los tipos de los pares de parámetros corresponden.

5. *Equiparación Semántica*. Comprueba si ambas especificaciones encajan a nivel semántico en términos tomados del área de la ingeniería del software. La descripción de un componente software D_2 encaja (plug-in) semánticamente con otra descripción D_1 si (1) sus signaturas encajan, (2) el conjunto de restricciones de D_1 lógicamente implica las de D_2, y (3) el conjunto de restricciones de D_2 lógicamente implica las de D_1.

El usuario puede seleccionar cuáles de los filtros desea que se apliquen, según el compromiso calidad-eficiencia que desee, eligiendo entre los siguientes modos:

1. Encaje Exacto o Completo. Es el más preciso. Ambas descripciones de servicio son equivalentes, bien por ser iguales o mediante renombrado de

variables o lógicamente iguales obtenido mediante inferencia lógica. Se aplican todos los filtros.

2. Encaje Plug-in. Menos exacto que el anterior, pero quizá más útil. Se obtiene cuando algunos criterios de equiparación se cumplen y otros no, porque alguno de los campos a comparar está en blanco. Sólo aplica los filtros de signatura y semántico.

3. Encaje Relajado. Menos exacto que los anteriores. Devuelve un valor numérico de distancia entre dos especificaciones aunque no equiparen semánticamente. Sólo aplica los filtros de contexto, perfil y similitud.

4. Encaje de Perfil. Sólo aplica los filtros de perfil y contexto.

2.2.3.3. IMPACT

IMPACT (Interactive Maryland Platform for Agents Collaborating Together) [4] es una plataforma para la construcción de agentes autónomos que puedan interactuar con otros de forma dinámica. Existe un *servidor de páginas amarillas* que es un matchmaker centralizado que conecta proveedores con solicitantes de servicios. El servidor de páginas amarillas se apoya en un *servidor de tipos* que mantiene una jerarquía de tipos y conceptos estándar, un *servidor de registros* encargado de registrar y mantener los servicios anunciados por los agentes proveedores, y un *tesauro* con sinónimos de términos. El lenguaje de descripción de servicios es un lenguaje de marcado tipo HTML. Una especificación de servicio se compone de:

- Nombre del servicio. De la forma *verbo:nombre(nombre)*. Por ejemplo, *vender:coche(japonés)* para un servicio de venta de coches japoneses.

- Entradas. El servicio asume que el usuario proporcionará cero o más entradas. Para cada entrada se especifica un nombre y tipo de la forma *nombre:tipo*. Por ejemplo, *destino:ciudad* especifica una entrada llamada destino del tipo ciudad (un enumerado, p. ej.).

- Salidas. Se especifica de forma análoga a las entradas.
- Atributos. Puede haber atributos adicionales para especificar el servicio como coste, tiempo medio respuesta, etc.

La figura 2.9 muestra un ejemplo de servicio para encontrar una ruta entre dos puntos de un mapa.

```
<S>
    find:route
    <MI>from_streetnumber:integer<\MI>
    <MI>from_street:string<\MI>
    <MI>from_city:city<\MI>
    <MI>from_state:us_states<\MI>
    <MI>to_streetnumber:integer<\MI>
    <MI>to_street:string<\MI>
    <MI>to_city:city<\MI>
    <MI>to_state:us_states<\MI>
    <O>map:file<\O>
<\S>
```

Figura 2.9: Ejemplo de descripción de servicio en IMPACT

Se utilizan dos enfoques para realizar la equiparación que pueden ser seleccionados por el solicitante. El primer método consiste en encontrar los k servicios más parecidos. El segundo método devuelve todos los servicios en un rango r de similitud. En ambos casos es necesaria una función de distancia entre dos servicios. Esta función se aplica solamente sobre el nombre de los servicios. Para ello el servidor de páginas amarillas dispone de una tabla de agentes como la de la tabla 2.3 con la descripción del nombre de cada agente y un identificador.

La búsqueda de servicios se realiza en base a dos jerarquías de verbos y nombres (figura 2.10). Estas jerarquías son grafos valorados dirigidos acíclicos. La distancia entre dos nodos es la suma de los pesos de los arcos del camino más corto entre esos nodos. Si no hay camino, la distancia es infinita. Si dos nom-

VERBO	TÉRMINO	AGENTE
rent	car(Japanese)	agent1
rent	car(American)	agent3
rent	vehicle()	agent6
hire	Honda()	agent4
rent	car(Japanese)	agent2
sell	Mazda()	agent2
rent	vehicle()	agent3

PALABRA	SINONIMOS
explore	inquire, examine, probe, investigate
vehicle	carrier, carriage, van, wagon, car, truck
car(Japanese)	Honda, Nissan, Mazda, Toyota
scan	browse, glance, skim
provide	supply, hand over
seek	search, look up
rent	hire, lease
car(American)	Chevy, Neon, Dodge

Tabla 2.3: Ejemplo de tablas de agentes y tesauro

bres o verbos están en el mismo nodo, su distancia es 0. Para comparar pares (verbo, nombre) es necesaria una función de distancia compuesta. Los autores describen las propiedades que debe cumplir la función de distancia. La suma es un ejemplo de función de distancia válida.

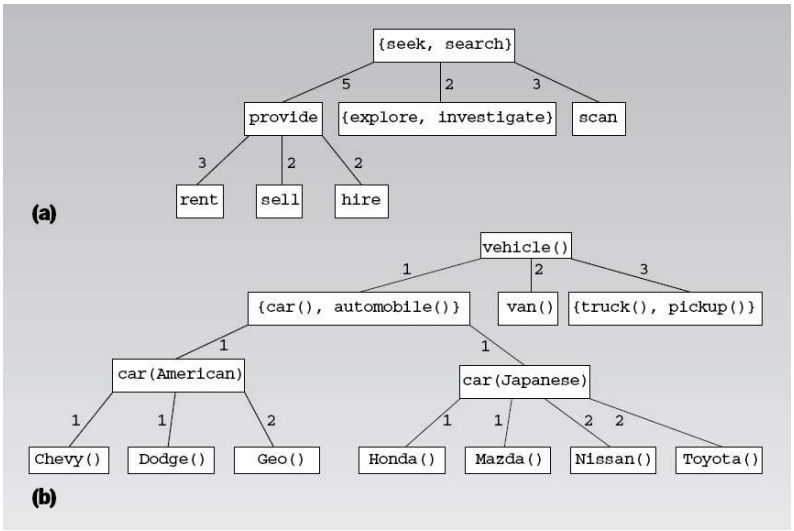


Figura 2.10: Ejemplos de jerarquías de a) verbos y b) nombres

2.2.3.4. OWL-S Matchmaker

Muchos de los actuales enfoques de equiparación de servicios Web semánticos, particularmente los basados en OWL-S, parten del trabajo de Paolucci et al. [92], que propone un algoritmo de equiparación para la búsqueda de servicios utilizando OWL-S como lenguaje de descripción (en realidad lo proponen para DAML-S, precursor de OWL-S). El algoritmo realiza equiparación semántica basándose en subsunción de conceptos de la ontología.

Un anuncio equipara con una solicitud de servicio cuando todas las salidas de la solicitud son equiparadas por las salidas del anuncio, y todas las entradas del servicio anunciado son equiparadas por las entradas de la solicitud. Con esto se asegura que el servicio proporciona todas las salidas deseadas por el

cliente y que éste proporciona todas las entradas necesarias.

La equiparación de las salidas se produce si y sólo si para cada salida de la solicitud existe una salida del anuncio que equipara con ella. El grado de equiparación se determina mediante la distancia mínima entre conceptos en el árbol de la ontología. Existen cuatro grados de equiparación (OUT_S y OUT_R corresponden a las salidas del anuncio y solicitud de servicio, respectivamente)[9]:

- *exact*: se $\text{OUT}_R \doteq \text{OUT}_S$; o OUT_R es una subclase directa de OUT_S bajo la asunción de que anunciando OUT_S el proveedor se compromete a proporcionar salidas consistentes con cada subtipo inmediato de OUT_S.
- *plug-in*: si $\text{OUT}_S \succeq \text{OUT}_R$, esto es, OUT_S podría ser utilizado en lugar de OUT_R.
- *subsumes*: si $\text{OUT}_R \succeq \text{OUT}_S$, entonces el proveedor no cumple exactamente con las necesidades del solicitante, pero éste quizá pueda usar el servicio para realizar parte de sus tareas.
- *fail*: no hay ninguna relación de subsunción entre OUT_S y OUT_R.

El resultado de la equiparación global de las salidas es el menor de los grados de equiparación de cada una de ellas. La equiparación de las entradas se realiza de forma análoga, pero con el orden de solicitud y anuncio invertido.

Por último, los servicios que equiparan en algún grado con la solicitud se ordenan teniendo como primer criterio el grado de equiparación de las salidas y en caso de igualdad, se tiene en cuenta el de las entradas.

2.2.3.5. OWLS-MX

OWLS-MX [67] realiza una equiparación semántica híbrida que complementa razonamiento lógico con medidas sintácticas de similitud utilizando técnicas

[9] donde $\doteq$ y $\succeq$ representan la equivalencia y subsunción de conceptos, respectivamente

de recuperación de información (IR). Establece también varios grados de equiparación, los primeros tres son basados en razonamiento lógico y, aunque los llama igual que Paolucci, se definen de forma diferente (por ejemplo, en OWLS-MX las entradas de los anuncios siempre deben por lo menos subsumir las de la solicitud, así el servicio puede ser invocado). En concreto (IN_S y IN_R corresponden a las entradas del anuncio y solicitud de servicio, respectivamente)[10].

- *exact*: si y sólo si $\forall\ \text{IN}_S\ \exists\ \text{IN}_R$: $\text{IN}_S \doteq \text{IN}_R \wedge \forall\ \text{OUT}_R\ \exists\ \text{OUT}_S$: $\text{OUT}_R \doteq \text{OUT}_S$.

- *plug-in*: $\forall\ \text{IN}_S\ \exists\ \text{IN}_R$: $\text{IN}_S \succeq \text{IN}_R \wedge \forall\ \text{OUT}_R\ \exists\ \text{OUT}_S$: $\text{OUT}_S \in \text{LSC}(\text{OUT}_R)$. Se espera que S devuelva un dato de salida más específico cuya semántica definida es muy cercana a la requerida por el usuario.

- *subsumes*: $\forall\ \text{IN}_S\ \exists\ \text{IN}_R$: $\text{IN}_S \succeq \text{IN}_R \wedge \forall\ \text{OUT}_R\ \exists\ \text{OUT}_S$: $\text{OUT}_R \succeq \text{OUT}_S$. Ésta relaja la restricción de subsunción directa.

- *subsumed-by*: $\forall\ \text{IN}_S\ \exists\ \text{IN}_R$: $\text{IN}_S \succeq \text{IN}_R \wedge \forall\ \text{OUT}_R\ \exists\ \text{OUT}_S$: $(\text{OUT}_S \doteq \text{OUT}_R \vee \text{OUT}_S \in \text{LGC}(\text{OUT}_R)) \wedge \text{SIM}_{IR}(\text{S, R}) \geq \alpha$. El dato de salida es más general que el solicitado. Se enfoca en conceptos de salida que sean padres directos para evitar seleccionar servicios que devuelvan datos demasiado generales. Se combina con la similaridad sintáctica.

- *logic-based fail*: la equiparación falla de acuerdo a los criterios de equiparación lógica anteriores.

- *nearest-neighbor* $\forall\ \text{IN}_S\ \exists\ \text{IN}_R$: $\text{IN}_S \succeq \text{IN}_R \wedge \forall\ \text{OUT}_R\ \exists\ \text{OUT}_S$: $\text{OUT}_R \succeq \text{OUT}_S \vee \text{SIM}_{IR}(\text{S, R}) \geq \alpha$.

- *fail*: el anuncio y la solicitud de servicio no encajan de acuerdo a los criterios anteriores.

[10] $LSC(C)$ (conjunto de conceptos menos específicos (hijos directos) de C), $LGC(C)$ (conjunto de conceptos menos genéricos (padres directos) de C), $Sim_{IR}(A, B) \in [0, 1]$ grado numérico de similitud sintáctica entre dos strings A y B de acuerdo con la métrica IR elegida

2.2.3.6. Li/Horrocks

Li y Horrocks [73] conciben la descripción de las entradas y salidas como un concepto en lógica descriptiva. Extienden los grados de equiparación propuestos por Paolucci añadiendo un nuevo grado, *intersection*. Formalmente,

- *exact*: si $A \equiv R$.
- *plug-in*: si $R \sqsubseteq A$.
- *subsume*: si $A \sqsubseteq R$.
- *intersection*: si $\neg(A \sqcap R \sqsubseteq \bot)$
- *disjoint*: si $A \sqcap R \sqsubseteq \bot$.

2.2.4. Composición de servicios

En el contexto de la computación orientada a servicios, cuando no existe ningún servicio que satisfaga las funcionalidades requeridas por el solicitante, debe existir la posibilidad de combinar varios servicios existentes que en conjunto cumplan con la necesidad deseada.

Esta necesidad ha desencadenado un gran esfuerzo en la actualidad en la búsqueda de técnicas para la composición de servicios. La composición de servicios Web se puede clasificar en tres categorías principales [132]:

- Composición fija: requiere que sus servicios constituyentes sean sintetizados de forma pre-especificada fija. La estructura de la composición y los servicios reales son ligados estáticamente.
- Composición semi-fija: requiere que el servicio compuesto sea especificado estáticamente pero las ligaduras reales sean decididas en tiempo de ejecución. Cuando un servicio compuesto es invocado, la especificación real es generada en base al encaje entre los servicios especificados en la composición abstracta y los posibles servicios disponibles.

- Composición explorativa: esta categoría requiere que los servicios compuestos sean generados "al vuelo" en base a una solicitud de un cliente. El proceso de búsqueda puede generar una serie de servicios compuestos factibles alternativos, que pueden ser puntuados o ser elegidos por los clientes en base a criterios no funcionales como disponibilidad, coste o rendimiento.

Al igual que lenguajes como UDDI, WSDL, SOAP y parte de OWL-S (ServiceProfile y ServiceGrounding) definen formas estándar para la descripción, descubrimiento e invocación de servicios, otras iniciativas como BPEL4WS y el ServiceModel de OWL-S se centran en representar composiciones de servicios donde el flujo de un proceso y las ligaduras (bindings) entre procesos se conocen a priori.

BPEL4WS (Business Process Execution Language for Web Services) [15] permite la especificación de procesos de negocio ejecutables (incluidos servicios Web) y protocolos de procesos en términos de sus lógicas de ejecución o flujo de control. Los *procesos ejecutables* especifican el comportamiento de participantes individuales en las interacciones internas del servicio Web y pueden ser invocados directamente, mientras los *protocolos* de negocio (también llamados procesos abstractos) abstraen el comportamiento interno para describir los mensajes intercambiados entre varios servicios Web en una interacción. Los procesos abstractos sólo consideran datos relevantes al protocolo e ignoran los datos y computación internos al proceso. Los procesos ejecutables, por otra parte, se describen usando un lenguaje rico de descripción de procesos que trata tanto con datos relevantes al protocolo como internos al proceso. Los procesos ejecutables también referencian tipos de puertos incluidos en documentos WSDL, que se usan para definir los roles que que pueden ser rellenados por aquellos servicios Web que cumplen el conjunto de restricciones del rol. Estos roles se pueden rellenar dinámicamente y la información específica del puerto puede ligarse en tiempo de ejecución. BPEL4WS también define varios mecanismos para recuperarse frente a fallos, como captura y manejo de fallos, y manejadores de compensación que especifican actividades de compensación

en caso de eventos que no pueden ser deshechos.

A pesar de estos esfuerzos, la composición de servicios Web es todavía una tarea bastante compleja, y está ya más allá de la capacidad humana para tratar con el proceso completo manualmente. La complejidad viene dada por diferentes motivos:

- el número de servicios disponibles en la Web va aumentando y se espera que continúe aumentando dramáticamente en el futuro
- los servicios Web pueden crearse y actualizarse "al vuelo"
- los servicios Web pueden ser creados por diferentes organizaciones o individuos, usando diferentes modelos para describir los servicios

Por tanto, la construcción de servicios Web compuestos mediante una herramienta automática o semiautomática es crítico. Con este fin, se han propuesto varios métodos. En particular, la mayoría de ellos utilizan técnicas de composición de workflows o planificación en Inteligencia Artificial.

Siguiendo el primer enfoque, un servicio compuesto es similar a un workflow: la definición de un servicio compuesto incluye un conjunto de servicios atómicos junto con el flujo de datos y control entre ellos. Las soluciones actuales consisten en el establecimiento manual del workflow pero permitiendo servicios atómicos abstractos que pueden ser ligados dinámmicamente, como ocurre en EFlow [21], donde un servicio compuesto se modela como un grafo que define el orden de ejecución de los procesos (nodos). Aunque el grafo se debe especificar manualmente, EFlow permite la automatización de la ligadura de nodos con servicios concretos. Polymorphic Process Model (PPM) [103] usa un método que combina composición de servicios estática y dinámica. La estática consiste en especificar servicios abstractos que tienen descripción de funcionalidad pero no implementación y que son asociados con servicios concretos en tiempo de ejecución (similar a EFlow). La parte dinámica se soporta mediante el modelado de los servicios por máquinas de estados que especifican los posibles estados de un servicio y sus transiciones. Las transiciones son causadas por

invocaciones de operaciones (o actividades) del servicio o transiciones internas al servicio. La composición dinámica se lleva a cabo mediante el razonamiento basado en la máquina de estados.

Por otro lado, se requieren métodos de composición dinámica para generar el plan automáticamente. La mayoría de éstos utilizan técnicas de planificación en IA y demostración de teoremas. La asunción general en esa clase de métodos es que cada servicio Web se puede especificar mediante sus precondiciones y efectos en el contexto de planificación. En primer lugar, un servicio Web es un componente software que toma los datos de entrada y produce los datos de salida. Por tanto, las precondiciones y efectos son los parámetros de entrada y salida del servicio respectivamente. En segundo lugar, un servicio Web también altera los estados del mundo después de su ejecución. Así que el estado del mundo requerido para la ejecución del servicio es la precondición, y el nuevo estado generado tras la ejecución es el efecto. Si el usuario es capaz de especificar las precondiciones y efectos requeridos para el servicio compuesto, se puede generar un plan automáticamente mediante un demostrador de teoremas lógicos o un planificador IA sin conocimiento de un workflow predefinido.

McIlraith et al. [80] usan un enfoque basado en cálculo de situaciones para representar y razonar sobre servicios (adaptan y extienden el lenguaje Golog). Cada servicio es una acción (primitiva o compleja). Las acciones primitivas se conciben como acciones que cambian el estado del mundo o acciones que cambian el conocimiento del agente. Las acciones complejas son composiciones de acciones individuales. Un servicio compuesto es un conjunto de servicios atómicos conectados mediante constructores de un lenguaje de programación procedural (if-then-else, while, etc.).

Medjahed [81] utiliza reglas de composición para determinar si dos servicios son componibles. Estas reglas consideran las propiedades sintácticas y semánticas de los servicios Web. Las reglas sintácticas incluyen reglas sobre los modos de operación y reglas para ligar los protocolos de los servicios interactivos. Las reglas semánticas incluyen reglas sobre (i) mensaje (dos servicios son componi-

bles solo si el mensaje de salida de uno es compatible con el mensaje de entrada del otro), (ii) operación (compatibilidad de dominios, categorías y propósito de los servicios), (iii) calidad (preferencias del usuario), y (iv) solidez (si una composición es razonable).

Sheshagiri et al. [109] desarrollaron un planificador de servicios DAML-S que usa servicios tipo STRIPS para componer un plan, dado un objetivo y un conjunto de servicios básicos. Está implementado con JESS, y usa un conjunto de reglas JESS para transformar descripciones OWL-S de servicios atómicos en operadores de planificación.

SWORD [95] también sigue un enfoque basado en reglas, en este caso, en forma de cláusulas de Horn, usando Prolog para el razonamiento. Un servicio Web se modela por sus precondiciones y postcondiciones, en forma de una regla (cláusula de Horn) que denota que las postcondiciones se alcanzan si las precondiciones son ciertas. El usuario especifica el estado inicial y final, y la generación del plan se consigue siguiendo la traza de ejecución del intérprete de Prolog.

En [113] se propone usar un planificador jerárgico (HTN), en concreto SHOP2 ([83]), para la composición de servicios Web. Presentan un método para transformar procesos OWL-S en árboles jerárquicos de tareas, muy apropiado debido a la estructura jerárquica del modelo de procesos de OWL-S. Los autores probaron la correspondencia semántica entre ambos modelos.

OWLS-XPlan [68] es un planificador de servicios compuestos que permite la composición rápida y flexible de servicios OWL-S. Para ello convierte la ontología del dominio (OWL) y las descripciones de servicios (OWL-S) en sus equivalentes descripciones del dominio y problema en lenguaje PDDL 2.1 [79]. Luego utiliza el planificador eficiente Xplan para generar el plan que satisface un objetivo dado. Xplan es un planificador heurístico basado en el FF-planner [58], que combina búsqueda local guiada en un grafo con planificación jerárquica y un componente de replanificación.

2.3. Discusión

En este capítulo se ha revisado el estado actual de los dos campos principales en los que se enmarca esta tesis doctoral: los sistemas muti-agente y la computación orientada a servicios.

En primer lugar se ha analizado el campo de los sistemas multiagente, campo en el que cantidad de personas han estado trabajando durante las últimas dos décadas. Durante estos años han ido surgiendo diversas metodologías y herramientas para la construcción de sistemas basados en agentes.

El análisis de las metodologías existentes más representativas muestra que todas ellas reconocen que el proceso de construcción de MASs es diferente de otros sistemas software más tradicionales. Los conceptos organizacionales, como roles e interacciones, juegan un papel fundamental en todas las metodologías, lo que demuestra la consciencia existente de su importancia en el modelado de este tipo de sistemas. Todas (o la mayoría) de ellas, conciben el sistema en términos de una *sociedad organizada* de individuos en la cual cada agente juega *roles* específicos e interactúa con otros agentes.

Recientemente ha surgido con gran fuerza la Computación Orientada a Servicios (SOC), donde las aplicaciones se construyen a partir de componentes básicos autocontenidos denominados servicios. Los servicios son descritos, las descripciones almacenadas en algún tipo de directorio (o páginas amarillas), son invocados y pueden ser compuestos para conseguir funcionalidades no existentes de forma aislada. Esto requiere de estándares que faciliten la interoperabilidad así como el descubrimiento y composición de servicios.

Los servicios Web se presentan como la tecnología dominante del paradigma SOC. En la actualidad, la industria ya está comenzando a adoptar esta tecnología, si bien se opera principalmente a nivel sintáctico (SOAP, WSDL, UDDI) y muchas de las tareas se realizan manualmente.

Sin embargo, el objetivo de la SOC es conseguir automatizar los procesos de búsqueda, invocación y composición de los servicios. Para conseguir ésto,

es fundamental el uso de ontologías para dotar de contenido semántico a las descripciones de los servicios, y razonadores que realicen las tareas de equiparación de servicios y métodos de planificación adaptados al contexto de los servicios Web. Los avances en el campo de la Web semántica están facilitando el desarrollo de las tecnologías necesarias para los servicios Web semánticos (que se pueden ver como la unión de los servicios Web y la Web semántica). Sin embargo estamos todavía ante los primeros pasos en el desarrollo de la Web semántica y, por consiguiente, todavía con mucho trabajo por delante para el pleno desarrollo de la tecnología basada en servicios Web semánticos.

En la Web semántica destacan RDF y OWL como lenguajes para representar ontologías, dependiendo de la capacidad expresiva necesaria. Para las descripciones de servicios OWL-S, una ontología escrita en OWL, es la referencia aunque también hay que considerar WSMO. Las propuestas actuales suelen basarse principalmente en la descripción de las entradas y salidas de los servicios, así como las precondiciones que deben cumplirse y los efectos que produce la ejecución del servicio. También se suelen incluir parámetros adicionales, como la categoría a la que pertenece el servicio.

Los métodos actuales de equiparación entre descripciones de servicios se basan en la similitud entre conceptos de una ontología. Aquí hay varios enfoques. Unos realizan un razonamiento puramente lógico (analizando la relación de subsunción entre conceptos). Otros tienen también en cuenta la distancia entre ellos o técnicas provenientes del campo de la recuperación de información para calcular un valor numérico que se suele combinar con el resultado del razonamiento lógico. Cabe destacar que la mayoría de los métodos existentes de equiparación de servicios sólo consideran las entradas y salidas de los servicios. Esto suele ser debido a que no existe una propusta clara sobre cómo especificar precondiciones y efectos, además de la complejidad computacional para comprobarlos. Por tanto, aún estamos ante un tema que seguramente evolucionará en los próximos años.

Si las técnicas de descripción y búsqueda de servicios están todavía en sus primeras etapas, más aún lo esta el tema de la composición automática de ser-

vicios. En general, la composición de servicios Web podría basarse en técnicas de planificación de IA existentes, aunque estos métodos fueron desarrollados para problemas donde el número de operadores es relativamente pequeño, pero que pueden producir planes complejos. Por contra, la composición de servicios Web para entornos abiertos a gran escala (como Internet) requiere métodos de planificación que puedan tratar con un número muy grande de posibles servicios, pero cuyos planes probablemente no sean muy complejos. Hasta la fecha no hay una técnica acordada como la adecuada para componer servicios cualquiera sea el formalismo utilizado para describirlos.

Las arquitecturas de agentes típicas coinciden en muchas de las características de las arquitecturas orientadas a servicios. Las arquitecturas de agentes proporcionan directorios de páginas amarillas y blancas, donde los agentes anuncian sus funcionalidades y donde otros agentes buscan para localizar agentes para solicitarles esas funcionalidades. Sin embargo, los agentes extienden los servicios Web en varios aspectos importantes:

- Un servicio sólo necesita saber de sí mismo, pero no de sus usuarios o clientes. Los agentes suelen ser auto conscientes, y mediante modelos y aprendizaje pueden ganar consciencia de otros agentes y sus capacidades según se llevan a cabo interacciones entre ellos. Esto es importante puesto que sin esa consciencia el servicio sería incapaz de aprovechar las nuevas capacidades de su entorno, y no podría adaptar su servicio al cliente.
- Los servicios, a diferencia de los agentes, no están diseñados para usar y reconciliar ontologías. Si un cliente y un proveedor usan diferentes ontologías la invocación del servicios produciría resultados imprevistos. Los agentes pueden mediar esas diferencias.
- Los agentes son inherentemente comunicativos, mientras que los servicios son pasivos hasta que se invocan.

Capítulo 3

Información organizacional en la descripción de servicios

Para mejorar tanto al eficiencia como la usabilidad de las arquitecturas orientadas a servicios basados en agentes, sugerimos explotar conceptos organizacionales comunes como *roles sociales* y *tipos de interacciones* para caracterizar mejor el contexto en el que pueden ser usados ciertos servicios semánticos.

En este capítulo, en primer lugar (sección 3.1) se motiva el uso de información organizacional para la descripción de servicios. En la sección 3.2 se describe el enfoque de modelado basado en roles e interacciones que se ha utilizado en este trabajo. Posteriormente (sección 3.3) se presenta una ontología para la representación de roles e interacciones. El detalle del enfoque propuesto para describir servicios mediante la información proporcionada por los modelos organizativos es desarrollado en la sección 3.4. El capítulo termina con una discusión sobre la propuesta realizada.

3.1. Motivación

A lo largo de esta memoria se van a utilizar ejemplos del dominio de asistencia en casos de emergencias médicas. Un escenario típico en este dominio puede ser el siguiente[1].

> Frans, un hombre de negocios finlandés, está de viaje en Austria cuando repentinamente sufre un dolor en el pecho hasta ahora desconocido para él. Frans no sabe qué hacer y activa el servicio instalado en su PDA, la cual contacta automáticamente con un Centro de Asistencia Médica de Urgencias (EMA) de Finlandia donde reside Frans. El operador le pregunta sus síntomas, solicitándole si es necesario que los detalle, y qué medicamentos está tomando actualmente. El operador de este centro le recomienda visitar un centro médico cercano a su lugar actual. El agente instalado en EMA puede informar automáticamente de que Frans está en camino del centro de salud, por si fuera necesario preparar alguna intervención. Frans recibe un mapa con información detallada de cómo llegar al hospital; su agente puede incluso llamar a un taxi si lo necesitara.
>
> Una vez llegado al centro de salud, Frans utiliza su PDA para registrarse y transferir información del tipo número de seguridad social, grupo sanguíneo o posibles alergias a medicamentos. Tras un examen inicial el médico necesitaría acceder a estudios anteriores realizados a Frans sobre esta patología. El agente de su PDA se conecta con el centro médico de Frans, previa autorización, para recabar la información necesaria y transmitírsela al centro que lo está tratando. Dado que el médico no está seguro del diagnóstico prefiere tener una segunda opinión de otro especialista en la materia. Utilizando el agente instalado en el centro médico localiza y contacta con un cardiólogo proporcionándole la información disponible sobre Frans. El cardiólogo pide al médico que proporcione información más precisa sobre un síntoma en particular (por ejemplo si la zona afectada por el dolor ha sido operada en el pasado). Finalmente el cardiólogo proporciona su opinión, pero no es suficientemente clara en cierta parte para el médico, por lo que le pide una explicación adicional. El médico local finalmente decide enviarlo a otro centro con sistemas cardiológicos

[1] Este escenario [22, 55] ha sido implementado en el proyecto CASCOM, como se describe más adelante en la sección 6.2

> avanzados. Las gestiones del nuevo hospital son realizadas utilizando el agente instalado en la PDA de Frans, que posee toda la información de la incidencia, por lo que Frans recibe el cuidado médico apropiado.

En este trabajo concebimos que los servicios son proporcionados esencialmente por agentes. La diferencia entre un servicio Web y un servicio proporcionado por un agente aparentemente podría reducirse a una cuestión de interfaz: un agente puede proporcionar un servicio web ya implementado mediante un proceso de encapsulación del servicio en una interfaz ACL [50] de forma que cualquier agente pueda invocar su ejecución enviando el mensaje (solicitud) adecuado.

Sin embargo, los agentes no sólo son capaces de llevar a cabo un servicio sino que también pueden involucrarse en diferentes tipos de interacciones para la provisión de ese servicio. Por ejemplo, considérese el escenario de asistencia médica como el anterior en el que un paciente o médico desea obtener una segunda opinión sobre cierto problema de salud: un agente que proporcione un servicio de *segunda opinión* debería no sólo ser capaz de proporcionar un diagnóstico, sino que podría ser requerido para que lo explicase, diera más detalles, recomendase un tratamiento, etc. Esto significa que se supone que el proveedor del servicio entabla diferentes interacciones durante la provisión de un servicio. Por tanto, si un médico o paciente necesita uno o más segundas opiniones, debería buscar agentes que incluyeran esas capacidades de interacción adicionales alrededor del servicio de *segunda opinión* "básico". En cierto sentido, este enfoque es similar a la abstracción que un objeto (en programación OO) hace al proporcionar un conjunto de métodos para los datos que encapsula. En este caso, el agente proporciona un conjunto de capacidades de interacción basadas en el servicio.

La mayoría de los matchmakers actuales [67, 73, 92] sólo consideran en los algoritmos de match inferencia lógica sobre las entradas y salidas de los servicios. Algunos pocos utilizan además otras características como las precondiciones, efectos o categoría de los servicios. En este trabajo se propone extender ese enfoque para incluir también información organizacional.

La *precisión* con la que se describen los servicios y, por tanto, la capacidad de encontrar un proveedor que proporcione un servicio, se puede mejorar incluyendo ese tipo de información. Por ejemplo, un servicio de diagnóstico puede requerir los síntomas y registros médicos como entradas y producir un informe como salida. Sin embargo, la funcionalidad del servicio puede alcanzarse de varias formas: (i) generando el informe, (ii) recuperando uno previamente hecho o (iii) mediante un servicio de mediación para contactar con otros expertos en medicina (externos). En los tres casos las entradas y salidas son las mismas, pero el *rol* que el proveedor del servicio juega en las correspondientes interacciones es diferente. Además, la *eficiencia* del proceso de búsqueda también se podría mejorar si previamente se filtran aquellos servicios que son incompatibles en términos de roles e interacciones.

3.2. Modelo organizacional para SMA orientados a servicios

Nuestro modelo organizacional de referencia para sistemas multiagente orientados a servicios (SMAOS), es una extensión de un subconjunto del modelo organizacional RICA (analizado en la sección 2.1.1.9). La figura 3.1 muestra esta extensión.

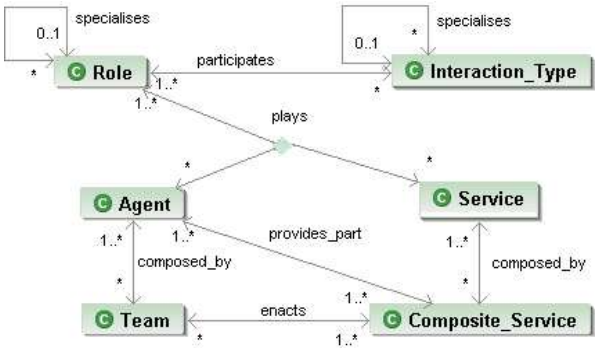


Figura 3.1: Meta-modelo organizacional de SMAOS

Los *servicios* son proporcionados por *agentes*, que son capaces de involucrarse

en diferentes *tipos de interacciones* para proporcionar o acceder a un servicio, jugando los *roles* que participan en ellas. Esto está representado mediante la relación *plays* en la figura 3.1. Los servicios pueden ser simples (en cuyo caso son proporcionados por un único agente) o pueden ser combinados para formar un *servicio compuesto* que proporcione una funcionalidad más compleja que ningún servicio simple es capaz de proporcionar. En este caso, un *equipo* de agentes colaboran proporcionando cada uno de ellos parte del servicio compuesto total. Un mismo agente puede proporcionar varios servicios y, por tanto, un servicio compuesto podría ser llevado a cabo por un único agente que proporcione todos los servicios simples que lo componen. Así pues, un equipo de agentes, que es la entidad que proporciona un servicio compuesto, podría estar formado por un único agente. Además, el mismo equipo puede proporcionar varios servicios compuestos. Varios agentes pueden ser capaces de proporcionar el mismo servicio simple, por lo que también varios equipos distintos de agentes puede proveer un servicio compuesto.

Para desarrollar extensiones basadas en conceptos organizativos de los mecanismos de descubrimiento de servicios utilizamos la parte de la información organizacional proporcionada por los roles y tipos de interacción de la figura 3.1. Para un dominio concreto esta información la obtenemos aplicando el método de diseño RICA [105].

Siguiendo este modelo, primero se analizan diferentes casos de uso del escenario del dominio aplicación. Para cada caso de uso, se identifican los tipos de interacciones sociales así como los roles (normalmente dos) que toman parte en esa interacción. El siguiente paso es un proceso de abstracción en el que los roles/interacciones sociales (dominio) se generalizan en roles/interacciones comunicativas (esta relación corresponde a la relación reflexiva *specialises* existente para roles e interacciones, mostrada en la figura 3.1).

A continuación se ilustrará el enfoque basado en roles mediante un caso de uso de segunda opinión en el dominio médico. En este escenario, el paciente (o el médico de un centro de emergencias médicas local) pide a un profesional externo una diagnosis en base a los síntomas y datos médicos del paciente,

como exámenes y enfermedades pasadas.

La conversación entre el paciente y el profesional médico se puede modelar mediante una secuencia de acciones (comunicativas) entre los dos agentes involucrados, como se muestra en la figura 3.2. El paciente *pide* al profesional médico una opinión, proporcionando los síntomas y los registros médicos. Si la información es insuficiente, el profesional médico *solicita* información adicional (posiblemente varias veces) y finalmente da su *consejo*. Si el diagnóstico no es lo suficientemente claro, el paciente también puede solicitar una *explicación*.

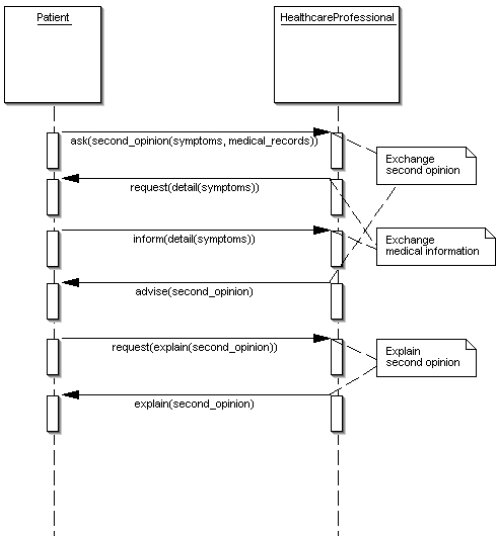


Figura 3.2: Conversación para una segunda opinión

A partir de esta conversación podemos identificar tres diferentes interacciones: (i) el intercambio de una segunda opinión, que puede incluir (ii) un intercambio de información detallada. Cuando el intercambio de la segunda opinión finaliza, puede (iii) llevarse a cabo una explicación.

Una vez identificadas las interacciones básicas, se pasaría a analizar en detalle cada una de ellas de una forma más precisa y además definir los roles involucrados en las interacciones. La figura 3.3 muestra un ejemplo de la interacción de *segunda opinión médica*, donde el rol *SecondOpinionRequestee* se generaliza en un rol *MedicalAdvisor* que, a su vez, se generaliza en un rol *Advisor*.

De forma similar, la interacción *SecondOpinion* se puede generalizar en una interacción *MedicalAdvisement* y luego en una interacción de *Advisement*, en la que el *Avisor* informa al *Advisee* sobre sus creencias con el objetivo de persuadir al *Advisee* de la bondad de sus creencias. A partir de ahí, se reutilizan los roles e interacciones subyacentes al modelo de FIPA (ver descripción de RICA en sección 2.1.1.9).

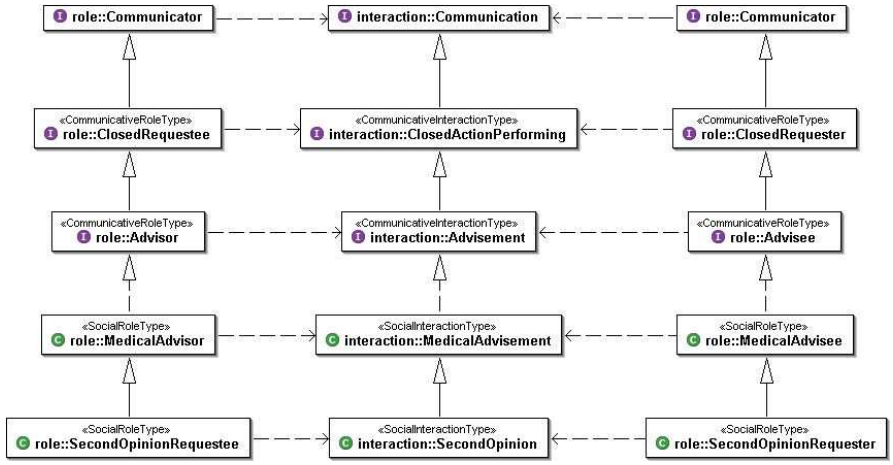


Figura 3.3: Refino de la interacción de segunda opinión

3.3. Ontología de roles e interacciones

Estudiando varios escenarios, hemos derivado una ontología que contiene una taxonomía de tipos de interacciones, y una taxonomía de roles que juegan parte en esas interacciones. La figura 3.4 muestra una representación, mediante un diagrama de clases UML, de la ontología de tipos de interacciones en el dominio de emergencias médicas (los detalles sobre su obtención se encuentran descritos en [24]). Hay dos clases de interacciones: tipos de interacciones sociales y tipos de interacciones comunicativas. Los *tipos de interacciones sociales* son interacciones del dominio (Asistencia en emergencias médicas, recomendaciones médicas, etc.). Los *tipos de interacciones comunicativas* son patrones de interacción genéricos reutilizables. Constituyen interacciones de comuni-

cación abstractas que pueden ser instanciadas en diferentes escenarios. Por ejemplo, *medical advisement* es una especialización, en el dominio médico, del tipo de interacción *advisement*. En el diagrama UML los tipos de interacciones genéricas (comunicativas) se han representado como interfaces, mientras que las interacciones del dominio (sociales) se han representado como clases.

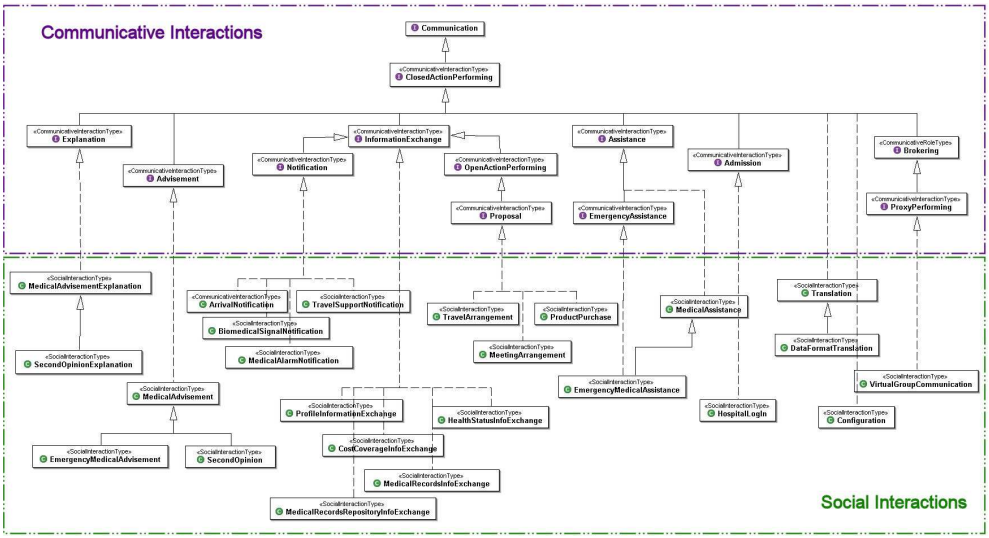


Figura 3.4: Ontología de tipos de interacciones (dominio de emergencias médicas)

El concepto raíz de la ontología es *Communication*, que representa el tipo de interacción más genérico. Cualquier tipo de interacción es una especialización de este concepto. El primer nivel de especialización es el tipo de interacción *ClosedActionPerforming*, que representa cualquier interacción que implique la ejecución de una acción. El matiz "closed" permite diferenciarlo del tipo de interacción *OpenActionPerforming* donde el solicitante deja un margen de alternativas abiertas para que la acción sea realizada (por ejemplo, una convocatoria de *propuestas*). *Explanation*, *Advisement*, *Admission*, *Assistance*, *InformationExchange* y *Brokering* son especializaciones de *ClosedActionPerforming*. Por ejemplo, un intercambio de información (*information exchange*) es un caso particular de ejecución de acción (*action performing*), donde el informado (*informee*) pide al informador (*informer*) que ejecute la

acción de informar sobre cierto hecho. Se puede aplicar un análisis similar a los tipos de interacción del dominio que especializan tipos de interacciones comunicativas en interacciones sociales. Por ejemplo, *ArrivalNotification* es una clase particular de notificación sobre la llegada de un paciente a un hospital. Se pueden añadir nuevos tipos de interacciones (genéricas y del dominio) de forma incremental en esta taxonomía haciéndola más completa y reutilizable.

En línea con la taxonomía de interacciones hay una taxonomía de roles que incluye una jerarquía de los roles que participan en las interacciones. Para cada concepto de la ontología de interacciones habrá al menos dos conceptos participantes en la ontología de roles. Por ejemplo, *informer* e *informee* son conceptos de la ontología de roles que participan en la interacción de intercambio de información (*information exchange*).

Estas ontologías, y especialmente su parte genérica (comunicativa), se usarán en la descripción y equiparación de servicios.

Definición 1 *Una ontología de roles e interacciones es una tupla* $O =< R, I, \sqsubset, \Delta >$*, donde:*

- R es el conjunto de conceptos que representan roles.
- I es el conjunto de conceptos que representan tipos de interacciones.
- $\sqsubset$ es una relación de orden parcial entre dos elementos de R o entre dos elementos de I, que representa una relación de subclase entre roles y entre tipos de interacciones. Así, si $c_1 \sqsubset c_2$ entonces c_1 en una especialización (subclase) directa del concepto c_2. Por ejemplo, $MedicalAdvisement \sqsubset Advisement$.
- $\Delta : R \rightarrow I$ es una función monótona[2] que asocia cada rol con el tipo de interacción en el que participa. Por ejemplo, $\Delta(MedicalAdvisor) = MedicalAdvisement$.

[2] $x \sqsubset y \rightarrow \Delta(x) \sqsubset \Delta(y)$

La relación de subsunción (x subsume a y, $y \prec$ x) es un mecanismo básico de inferencia en ontologías y que usaremos en nuestro caso. Se puede definir de la siguiente manera:

$$\forall x \forall y (y \prec x \leftrightarrow (y \sqsubset x \vee \exists z (y \sqsubset z \wedge z \prec x))) \tag{3.1}$$

En esta ontología deben cumplirse las siguientes propiedades (que vienen a ser un subconjunto del meta modelo RICA):

1. La ontología está formada por dos jerarquías de conceptos con raíz *Communication* en el caso de las interacciones, y *Communicator* para la taxonomía de roles:

$$\forall x \in R (x = Communicator \vee x \prec Communicator)$$

$$\forall x \in I (x = Communication \vee x \prec Communication)$$

2. La relación de especialización se aplica ente dos conceptos del mismo tipo (roles o interacciones)

$$\forall x \forall y (x \sqsubset y \rightarrow ((x \in R \wedge y \in R) \vee (x \in I \wedge y \in I)))$$

3. Para cada rol existe al menos un tipo de interacción en la que participa

$$\forall x \in R \; \exists y \in I \Delta(x) = y$$

4. En toda interacción al menos participa un rol

$$\forall x (x \in I \rightarrow \exists y (y \in R \wedge \Delta(y) = x))$$

5. La misma relación de especialización se cumple tanto en la taxonomía de roles como en la de interacciones.

$$\forall x \forall y \in R (x \sqsubset y \leftrightarrow (\exists x', y' \in I (\Delta(x) = x' \wedge \Delta(y) = y' \wedge x' \sqsubset y')))$$

3.4. Descripción de servicios

En una arquitectura orientada a servicios, suele existir algún tipo de directorio (también llamados páginas amarillas) en donde se registran las descripciones de servicios. Cuando se requiere un servicio, la búsqueda se basa en estas descripciones.

En esta sección se presenta una propuesta sobre cómo describir los servicios usando su información organizacional: los roles jugados en las interacciones en las que son capaces de participar. Primero se describe la información incluida en los anuncios de servicios y en las consultas y posteriormente cómo puede incluirse esta información en OWL-S, el lenguaje de descripción de servicios utilizado como referencia.

Hay que destacar que en este trabajo debe entenderse como una extensión de los enfoques actuales, es decir, que además de la información que se propone en los siguientes apartados una descripción de servicio también debe contener información sobre sus entradas, salidas, precondiciones, efectos, etc.

3.4.1. Descripción de anuncios de servicios

Se utilizará el caso del servicio de segunda opinión introducido en secciones anteriores para ilustrar nuestro enfoque. En este ejemplo, el proveedor del servicio (un profesional médico remoto) y el solicitante (un paciente o médico local) entablan una conversación durante la provisión de un consejo médico (*advisement*). El profesional médico remoto juega el rol *advisor*. Sin embargo, para poder dar el servicio adecuadamente necesita iniciar otro tipo de interacción (*information exchange*) para pedir al paciente que le informe sobre información adicional. Eso significa que el proveedor del servicio requiere que el solicitante sea capaz de jugar el rol *informer*.

Esta situación, presente también en muchos otros tipos de conversaciones, nos lleva a especificar dos clases de información relativa a las interacciones en las que el agente proveedor del servicio puede participar:

- ***Provider Role***: es el rol jugado por el agente proveedor del servicio en la interacción. Por ejemplo, el rol *advisor* en el servicio de segunda opinión.

- ***Depending Roles***: es un conjunto de roles que deben poder ser jugados por el usuario del servicio para su correcta ejecución. Representamos estos roles mediante una fórmula lógica en forma normal disyuntiva (FND), esto es, una disyunción de conjunciones de roles. Esta formulación permite expresar, por ejemplo, que para poder dar el servicio correctamente el usuario del servicio debe ser capaz de explicar (rol *explainer*) o al menos informar (*informer*).

Estos dos campos se repiten por cada rol que el proveedor del servicio es capaz de jugar. Podemos representar gráficamente un anuncio de servicio mediante una tabla con dos columnas, una para el *provider role* y la otra para los *depending roles*. Cada fila representa un tipo de interacción diferente en la que el proveedor es capaz de participar para proveer el servicio. La tabla 3.1 muestra un ejemplo abstracto.

Interacción	*provider role*	*depending roles*
I_1	r_1	$(r_4 \wedge r_5) \vee r_6)$
I_2	r_2	$(r_6 \wedge r_7 \wedge r_8) \vee (r_4 \wedge r_7)$
I_3	r_3	
...	...	...

Tabla 3.1: Ejemplo abstracto de un anuncio basado en roles

En este ejemplo abstracto, el agente proveedor del servicio puede entablar tres tipos diferentes de interacciones para proporcionar el servicio. Puede jugar el rol r_1, que requiere que el solicitante sea capaz de jugar los roles (r_4 y r_5, o r_6). Si esta condición falla, entonces el proveedor no podrá ejecutar el servicio adecuadamente. Sin embargo, cuando el proveedor juega el rol r_2, requiere que el solicitante juegue o bien los roles (r_6, r_7 y r_8); o en otro caso los roles r_4 y r_7. En el caso de r_3, no se exige que el solicitante sepa jugar un determinado rol.

La tabla 3.2 muestra un anuncio basado en roles para el servicio del ejemplo de segunda opinión. El agente anuncia que es capaz de jugar tres tipos diferentes de interacciones para la provisión del servicio, en las cuales puede jugar los roles de *advisor*, *explainer* e *informer*. Además especifica que en la interacción de asesoramiento (*advisor*) puede necesitar que el solicitante sea capaz de jugar el rol de informador (*informer*).

Interacción	*provider role*	*depending roles*
I_1	$Advisor$	$Informer$
I_2	$Explainer$	
I_3	$Informer$	

Tabla 3.2: Ejemplo de anuncio para el servicio de segunda opinión

Teniendo en cuenta las consideraciones anteriores, definimos una descripción de servicio de la siguiente manera.

Definición 2 *Un anuncio de servicio S es un conjunto de pares tal que,*

$$S \subseteq \left\{ < r, \rho > | r \in R, \rho = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} r_{ij}, r_{ij} \in R \right\} \tag{3.2}$$

Siendo R el conjunto de roles de la ontología O descrita en la sección 3.3 ($O =< R, I, \sqsubset, \Delta >$), y donde r es el rol jugado por el proveedor (*Provider Role*) y ρ la expresión en FND que representa los *Depending Roles*.

Para el ejemplo de la tabla 3.2 $S = \{< Advisor, Informer >, < Explainer, - >, < Informer, - >\}$.

3.4.2. Descripción de solicitudes de servicio

En el caso de las solicitudes de servicio, consideramos consultas que con dos elementos:

- ***Searched Provider Roles***: son los roles que el solicitante está buscando. Aunque en la mayoría de los casos sólo un rol será suficiente, para mayor flexibilidad expresiva permitimos unos patrones de búsqueda un poco más complejos en los que el proveedor es capaz de jugar más de un rol. Por ejemplo, un agente podría estar interesado en un servicio de segunda opinión (rol *advisor*) en el cual el proveedor sea también capaz de explicar (*explainer*) su recomendación. Como en el caso de los anuncios de servicios, se requiere aquí una expresión en forma normal disyuntiva.
- ***Requester Capability Roles***: un conjunto de roles que definen las capacidades del solicitante respecto a los roles que es capaz de jugar en interacciones. Esta información es importante si el proveedor requiere capacidades de interacción por parte de los solicitantes. Por ejemplo, el solicitante de una segunda opinión puede incluir en la búsqueda que es capaz de proporcionar información (*informer*) si es necesario.

La tabla 3.3 muestra un ejemplo abstracto de una consulta en la que el solicitante busca un proveedor de servicio que sea capaz de jugar el rol r_2 o tanto el r_4 como el r_6 para proporcionar el servicio. Además, declara que es capaz de jugar los roles r_4, r_7 y r_1 si así fuera requerido. Aunque la equiparación de servicios se describe en detalle en el capítulo 4, intuitivamente puede observarse que el servicio descrito en la tabla 3.1 encaja con esta consulta porque el proveedor es capaz de jugar uno de los roles (r_2) requeridos por el solicitante y, ademas, el éste sabe jugar algunos roles (r_4 y r_7) que hacen cierta la condición de *requester capability roles*.

Searched Provider Roles	$r_2 \vee (r_6 \wedge r_4)$
Requester Capability Roles	r_4, r_7, r_1

Tabla 3.3: Ejemplo abstracto de una consulta basada en roles

La tabla 3.4 muestra un ejemplo en el que un agente está buscando un servicio de segunda opinión (rol *advisor*) que explique sus propuestas (rol *explainer*). La consulta especifica que el solicitante es capaz de jugar los roles *informer* y *explainer* si fuera necesario.

Searched Provider Roles	$Advisor \wedge Explainer$
Requester Capability Roles	$Informer, Explainer$

Tabla 3.4: Ejemplo consulta basada en roles para un servicio de segunda opinión

Formalmente, una solicitud de búsqueda de servicio se define de la siguiente manera.

Definición 3 *Una solicitud de servicio Q es un par tal que,*

$$Q =< \rho, C >, \rho = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} r_{ij}, r_{ij} \in R, C \subseteq R \tag{3.3}$$

De nuevo, R es el conjunto de roles de la ontología O, ρ es una expresión en FND que representa los roles buscados (*Searched Provider Roles*), y C es el conjunto de roles que definen las capacidades del solicitante (*Requester Capability Roles*).

Alternativamente se propone otra descripción del anuncio S y consulta Q, más orientada a su uso para especificar un algoritmo, que será usada en el capítulo 4.1.

```
S = SET OF InteractiveRoleDescription
InteractiveRoleDescription = <ProviderRole, RoleExpression>

RoleExpression = SET OF ConjunctiveRoleList
ConjunctiveRoleList = SET OF Roles

Q = <SearchedRoles, RequesterCapabilities>
SearchedRoles = RoleExpression
RequesterCapabilities = SET OF Roles
```

Este enfoque es compatible con servicios que no utilizan la extensión basada en roles comunicativos en sus descripciones. Si una descripción de servicio no

incluye información basada en roles, asumimos que tiene un *searched provider role* de *Communicator* (el concepto más general de la ontología) y que no se requieren capacidades de interacción adicionales por parte del solicitante. Si en su búsqueda el solicitante no incluye descripción de roles, entonces se asume que no está interesado (o desconoce) en el enfoque basado en roles y, por tanto, se omitirá esa fase en el proceso de equiparación.

3.4.3. Extensión de OWL-S con roles

En esta sección se propone cómo incluir la información descrita en el apartado anterior en el lenguaje de descripción de servicios OWL-S, quizá el lenguaje más extendido en la actualidad, que es el utilizado en este trabajo. Como se describió en la sección 2.2.1.4 una descripción de servicio en OWL-S se compone de tres partes:

- el *service profile*, que describe qué hace el servicio y es utilizado para anunciar y buscar servicios
- el *process model*, que da detalles de cómo opera el servicio
- el *service grounding*, que proporciona detalles de cómo interoperar con el servicio

El *service profile* está diseñado para describir lo que hace el servicio, por lo que es el lugar adecuado para la descripción de los roles. Un *service profile* básicamente incluye información sobre entradas, salidas, precondiciones y efectos del servicio, así como la categoría y un conjunto de parámetros que pueden incluirse ad-hoc. Sin embargo, no incluye un campo específico para describir información organizacional como roles e interacciones, por lo que tenemos que utilizar los campos disponibles para incorporar esta información. Se optó por utilizar los parámetros para incluir esta información. En concreto, se definirá un parámetro denominado *SERVICE_ROLES* en el caso de las descripciones de servicios (anuncios) y *QUERY_ROLES* en el caso de las solicitudes de servicio.

La figura 3.5 muestra parte de una descripción en OWL-S para el anuncio de un servicio de segunda opinión (mostrado en la tabla 3.2).

La tabla 3.5 muestra un ejemplo en el que un agente está buscando un proveedor de servicio de segunda opinión. La consulta especifica que se busca un *advisor* y que el solicitante es capaz de jugar los roles *informer* y *explainer* si fuera necesario. La figura 3.6 muestra parte de su especificación en OWL-S.

Searched Provider Roles	*Advisor*
Requester Capability Roles	$Informer, Explainer$

Tabla 3.5: Ejemplo consulta basada en roles para un servicio de segunda opinión

3.5. Discusión

Analizando las actuales tendencias para la descripción de servicios que proporcionan los agentes (OWL-S, WSMO) se puede observar que nos encontramos en un estado en el que básicamente se realiza a través de la especificación de las entradas y salidas del servicio, descritas como un conjunto de conceptos definidos en cierta ontología compartida. También se suele incluir información textual sobre el servicio, como su nombre u otra descripción, pero ésta va dirigida a los humanos y no suele ser procesada automáticamente. Además de entradas y salidas, también se suelen indicar como relevantes las precondiciones que deben cumplirse para que el servicio pueda ser invocado y los efectos que producirá, si bien, aunque hay varias propuestas, aún no está claro cómo especificar esta información y no suele ser utilizada por las herramientas de equiparación. A veces también se destaca como información relevante la categoría del servicio (según alguna taxonomía) así como características no funcionales.

En este capítulo se ha propuesto utilizar, además de lo anterior, la información organizacional subyacente a los modelos multiagente para tenerla en cuenta en la descripción de la funcionalidad de los servicios, de forma que también pueda

ser tenida en cuenta en pasos de razonamiento relacionados como el descubrimiento o la composición de servicios. Hasta la fecha, al menos que seamos conscientes, no existen propuestas en esta línea. Lo más parecido puede ser la especificación de detalles del protocolo de comunicación utilizado, descrito en las plantillas de FIPA o en los groundings de OWL-S que dan detalles de invocación del servicio. Sin embargo, el nivel de abstracción usado en este trabajo es más alto.

Esta información, de un nivel de abstracción mayor que las entradas/salidas, permite aumentar la expresividad de las descripciones de servicios, de forma que se mejore la precisión y eficiencia del proceso de descubrimiento de servicios (capítulo 4).

Para esto, se ha buscado un método que añada expresividad pero manteniendo una ontología lo suficientemente sencilla para conseguir una buena eficiencia computacional. Esto se ha logrado mediante la definición de la parte genérica (comunicativa) de la ontología de roles. Con ello se mantiene una ontología relativamente estática a lo largo del tiempo lo cual puede ser beneficioso a la hora de diseñar mecanismos eficientes que usen esa taxonomía. Sin embargo, no implica que no se pueda extender la ontología. Según se analicen más aplicaciones la ontología será más completa y reutilizable y cada vez más estable.

Entre las diferentes propuestas para modelar un sistema multiagente desde un punto de vista organizacional, se ha seguido el enfoque RICA. Este modelo permite reflejar el contexto comunicativo en el que participan los agentes, y facilita la reutilización de roles. Además RICA parte de un análisis de la parte comunicativa del estándar FIPA y dispone de un entorno software (RICAJ) que, aunque no es necesario para nuestra aportación, sí puede ser interesante a nivel global del sistema.

Según el enfoque RICA los roles e interacciones sociales que surgen del análisis de los diálogos en los que participan los agentes se van refinando mediante un proceso de abstracción, mediante el cual se obtiene una jerarquía de roles (e interacciones) donde los roles independientes del dominio (denominados también comunicativos) se pueden reutilizar en otras aplicaciones. Es esta

información genérica la que se incluye en la descripción de los servicios. Si bien esta decisión provoca una pérdida de expresividad (en lugar de expresar un servicio de segunda opinión médica solamente se indica un servicio de asesoramiento), la decisión se fundamenta en lo siguiente. En primer lugar, permite mantener una ontología de roles comunicativos compartida de una dimensión reducida y bastante estable en el tiempo, ya que gracias al alto nivel de reutilización, en la mayoría de los casos no habrá que extenderla. Esto tiene la ventaja de que se pueden diseñar mecanismos de razonamiento más eficientes sobre esta ontología (p.ej. se puede tener pre-calculado el grado de similitud entre los diferentes roles). Sin embargo, el inconveniente de la pérdida de la información del dominio no es tan grave si se tiene en cuenta que lo aquí propuesto es una extensión que irá acompañada por el resto de características que describen un servicio, que mayormente son dependientes del dominio (por ejemplo, si una entrada se ha descrito como los resultados de unos análisis de rayos x, se deduce que el servicio de asesoramiento es en el dominio médico). Por el contrario, el mantenimiento de ontologías de roles específicas del dominio puede ser algo más complejo.

En cuanto a los detalles prácticos, se ha tomado como referencia el lenguaje OWL-S (la ontología basada en OWL para describir servicios), ya que es el más extendido en la actualidad, aunque WSMO también está emergiendo con fuerza. Dentro de las tres sub-ontologías que forman parte de OWL-S, la más adecuada es el *service profile* dedicado a describir qué es lo que hace el servicio. En dicha ontología se definen una serie de campos fijos, junto con la posibilidad de añadir otros parámetros según el deseo del usuario que permiten añadir características adicionales a las descripciones de servicios. Ese es el lugar elegido para incorporar la información organizacional, proponiendo unos nombres de parámetros concretos para los roles y expresiones lógicas de roles.

Aunque en este capítulo se han resaltado tanto la taxonomía de roles como la de interacciones y se han relacionado formalmente, finalmente sólo la de roles ha sido utilizada en las descripciones de servicios. Sin embargo se han mantenido ambas en la sección 3.3 en vista a futuras extensiones o utilidad de estas ontologías, además de como documentación. Las ontologías han sido im-

plementadas en OWL para facilitar su integración en OWL-S, aunque debido a su sencillez (básicamente relaciones de subclase), se podría haber elegido un lenguaje más sencillo (por ejemplo, RDF).

Será tema de trabajo futuro la integración de este enfoque en WSMO, en concreto en lo que WSMO denomina *servicios* (anuncios) y *objetivos* (solicitudes de servicios), que se basan en las ontologías descritas en el lenguaje WSML. Por tanto, se explorará la implementación de la ontología en WSML o su uso directamente en OWL.

```
<profile:ServiceParameter rdf:ID="SERVICE_ROLES">
  <profile:sParameter>
   <roles:ServiceRoles rdf:ID="ROLE_DESCRIPTION_LIST">
    <roles:interactiveRole>
     <roles:InteractiveRoleDescription rdf:ID="INTERACTIVE_ROLE_1">
      <roles:providerRole rdf:resource="http://...#AdvisorRole"/>
      <roles:dependingRoles rdf:resource="#DEPENDING_ROLES_1"/>
     </roles:InteractiveRoleDescription>
    </roles:interactiveRole>
    <roles:interactiveRole>
     <roles:InteractiveRoleDescription rdf:ID="INTERACTIVE_ROLE_2">
      <roles:providerRole rdf:resource="http://...#ExplainerRole"/>
     </roles:InteractiveRoleDescription>
    </roles:interactiveRole>
    <roles:interactiveRole>
     <roles:InteractiveRoleDescription rdf:ID="INTERACTIVE_ROLE_3">
      <roles:providerRole rdf:resource="http://...#InformerRole"/>
     </roles:InteractiveRoleDescription>
    </roles:interactiveRole>
   </roles:ServiceRoles>
  </profile:sParameter>
</profile:ServiceParameter>

<roles:RoleExpression rdf:ID="DEPENDING_ROLES_1">
  <roles:item>
    <roles:ConjunctiveRoleList rdf:ID="CONJUNCTIVE_ROLE_LIST_1_2">
      <roles:roleEntry rdf:resource="http://...#InformerRole"/>
    </roles:ConjunctiveRoleList>
  </roles:item>
</roles:RoleExpression>
```

Figura 3.5: Descripción parcial en OWL-S de un anuncio de servicio (ejemplo de tabla 3.2)

```
<profile:ServiceParameter rdf:ID="QUERY_ROLES">
  <profile:sParameter>
    <roles:QueryRoles rdf:ID="ROLE_DESCRIPTION_LIST">
      <roles:searchedProviderRoles>
        <roles:RoleExpression rdf:ID="QUERY_ROLE_EXPRESSION">
          <roles:item rdf:resource="#CONJUNCTIVE_ROLE_LIST_1"/>
        </roles:RoleExpression>
      </roles:searchedProviderRoles>
      <roles:capabilityRoles rdf:resource="#CAPABILITIES_LIST"/>
    </roles:QueryRoles>
  </profile:sParameter>
</profile:ServiceParameter>

<roles:ConjunctiveRoleList rdf:ID="CONJUNCTIVE_ROLE_LIST_1">
    <roles:roleEntry rdf:resource="http://.../roles.owl#AdvisorRole"/>
</roles:ConjunctiveRoleList>

<roles:ConjunctiveRoleList rdf:ID="CAPABILITIES_LIST">
    <roles:roleEntry rdf:resource="http://.../roles.owl#InformerRole"/>
    <roles:roleEntry rdf:resource="http://.../roles.owl#ExplainerRole"/>
</roles:ConjunctiveRoleList>
```

Figura 3.6: Descripción parcial en OWL-S de una solicitud de servicio (ejemplo de tabla 3.4)

Capítulo 4

Búsqueda de servicios basada en roles

El proceso habitual de búsqueda de un servicio consiste en comparar una descripción del servicio deseado con cada una de las descripciones anunciadas o registradas en un directorio. El resultado es una ordenación del conjunto de servicios de acuerdo con el grado de encaje con el servicio buscado. Queda fuera del alcance de este trabajo qué se hace con el conjunto ordenado, por ejemplo si se selecciona el primer servicio o si se utiliza algún mecanismo de selección.

En este capítulo se describe un algoritmo de equiparación (*matching*) para descripciones de servicios basados en roles. El objetivo de la equiparación es conseguir una medida que indique cómo de bien se adapta un servicio (proporcionado por cierto agente) con respecto a la descripción de un servicio que se está buscando (consulta).

Las tendencias actuales en equiparación de servicios, en particular las basadas en descripciones OWL-S, suelen clasificar los servicios anunciados en una de varias "categorías" según un razonamiento basado en lógica. Los servicios son ordenados según su categoría. En este trabajo se propone un algoritmo que

combina las ideas de estos enfoques pero donde el resultado se refina para obtener valores numéricos, siguiendo ideas extraídas del campo de similitud entre conceptos en ontologías (sección 2.2.2).

El resultado que permitirá obtener el algoritmo será, para cada servicio anunciado en el directorio, un valor numérico en la escala real [0..1] que mide la similitud o encaje con el servicio buscado, es decir, cuanto más alto sea el valor más similar se considera. El hecho de asociar un valor numérico a cada servicio permite una ordenación más precisa que si sólo se asignan a una categoría de encaje.

Es importante destacar que el enfoque aquí propuesto se centra en la información organizacional (roles) incluida en las descripciones de los servicios y que fue presentada en el capítulo 3. Esta propuesta debe combinarse con otra de propósito general, como las descritas en la sección 2.2.3, que analice otra información tenida en cuenta típicamente por los matchmakers actuales, como entradas, salidas, precondiciones, efectos, categoría, etc.

La función de comparación se describe en las siguientes dos secciones. En primer lugar (sección 4.1) se describe cómo se calcula el grado de equiparación entre dos roles, uno perteneciente a la consulta y otro perteneciente a un servicio anunciado. Los roles son los elementos básicos que aparecen en una descripción de servicio. Posteriormente (sección 4.2) se describe cómo se obtiene el valor de equiparación entre dos servicios, mediante un proceso de agregación de los resultados obtenidos sobre la similitud de pares de roles. A continuación, en la sección 4.3, se comentan aspectos relativos a la instrumentación software del algoritmo. Los experimentos realizados para la evaluación son detallados en la sección 4.4. Finalmente, el capítulo termina con una discusión sobre los resultados alcanzados.

4.1. Equiparación entre roles

En esta sección se describe cómo se calcula el grado de match entre dos roles. En primer lugar comenzaremos definiendo los requisitos que se desean de la función de equiparación (match) ente dos roles.

1. Debe devolver un valor numérico en el rango real [0..1], siendo un valor más alto cuando más similares se consideren los roles (1 si son iguales). El hecho de ser elegir un valor numérico es para facilitar la comparación, posterior agregación de valores y, finalmente, la ordenación entre los servicios, de una forma fina. Aunque el rango podría ser cualquiera se ha elegido el intervalo [0..1] puesto que es muy intuitivo y ampliamente usado.

2. Debe considerar la distancia entre los conceptos en la ontología, roles a más distancia entre ellos se consideran menos similares. Es decir, debe ser una función decreciente con la distancia.

3. Además, el decremento del grado de equiparación no debe ser el mismo para una misma distancia sino que debe disminuir gradualmente con la distancia entre los conceptos. Es decir, el decremento producido entre encontrarse a distancia de 1 a 2 conceptos debe ser mayor que, por ejemplo, el paso de estar a 5 a estar a 6. Esto es así porque una unidad cuando la distancia es pequeña es más relevante que una unidad cuando la distancia es grande.

4. El cálculo debe ser independiente de la altura total de la taxonomía y la situación de los roles en ella, es decir, la equiparación entre dos roles a la misma distancia debe devolver el mismo valor independientemente del resto de la ontología.

5. Debe tener en cuenta la relación lógica entre los dos roles, esto es, la relación de subsunción según la ontología de roles, que tendrá prioridad respecto a los valores numéricos. Siguiendo los enfoques habituales (como [67, 92]), se definirán varias clases de equiparación con un orden

establecido entre ellas. Así el valor numérico obtenido debe respetar el orden establecido por dichas clases.

Para cumplir el requisito 2 se utilizará la medida de distancia más común, la propuesta por Rada et al. [96] (ecuación (2.1)) y que consiste en la suma de arcos en el camino mínimo entre ambos conceptos en la ontología:

$$dist(r_1, r_2) = depth(r_1) + depth(r_2) - 2 \times depth(lcs(r_1, r_2)) \tag{4.1}$$

Puesto que el requisito 3 exige que el decremento no sea lineal, aquí se puede elegir una función exponencial, que debe ser inversa para asegurar que decrece con el aumento de la distancia entre los roles. Esta función puede ser:

$$e^{-dist(r_1,r_2)} \tag{4.2}$$

Además se puede observar que esta ecuación cumple que mantiene el valor en el rango [0..1], es monótona decreciente con la distancia y vale 1 cuando $r_1 = r_2$ (requisito 1), y es independiente de la altura total de la ontología (requisito 4) ya que solo depende de la distancia entre los roles.

El requisito 5 requiere un análisis más detallado. Utilizamos las cuatro clases de grados de match propuestas por Paolucci et al. [92], de mayor a menor encaje: *exact*, *plug-in*, *subsumes* y *fail*. En particular, para determinar el grado de match entre un rol r_A incluido en un anuncio y un rol r_Q correspondiente a la solicitud de servicio, las clases anteriores se tratan según lo siguiente:

- *Exact*: si ambos roles coinciden ($r_A = r_Q$).
- *Plug-in*: si r_Q subsume a r_A ($r_A \prec r_Q$).
- *Subsumes*: si r_A subsume a r_Q ($r_Q \prec r_A$).
- *Fail*: en otro caso, es decir, no hay relación de subsunción entre r_A y r_Q.

Obsérvese que se prefiere (*plug-in*) un servicio cuyo rol sea más específico que el buscado antes que uno más general (*subsumes*). Por ejemplo, si se busca un intercambio de información (rol *Informer*) se prefiere un notificador (*Notifier*) mejor que un más genérico *ClosedRequester*.

Ahora lo que hay que conseguir es escalar el valor numérico obtenido por la función (4.2) en cada una de esas clases de encaje, ya que se prefiere dar preferencia a encaje lógico y refinarlo con el valor numérico de similitud.

Una forma de realizar esta asociación es dividiendo el rango 0..1 en intervalos excluyentes y escalar el valor obtenido desde [0..1] a ese intervalo (la figura 4.1 ilustra esta idea). En general, cualquier división es aceptable bajo la condición de que mantenga el orden de sus categorías de encaje. Realizamos un análisis general de este método de escalado, para lo que nos apoyamos en la definición de las siguientes funciones (siendo c_1 y c_2 conceptos):

- $com(c_1, c_2)$: devuelve la *clase de match* lógica $\in \{cat_1, cat_2, ..., cat_N\}$, donde $cat_1 > cat_2 > cat_N$, para los conceptos c_1 y c_2. Normalmente $cat_1 = exact$ y $cat_N = fail$ y tendrán asociados los valores 1 y 0 respectivamente. El resto de categorías tendrán un intervalo de valores posibles.

- $inf(cat)$: devuelve el valor inferior del intervalo correspondiente a la categoría *cat*.

- $sup(cat)$: devuelve el valor superior del intervalo correspondiente a la categoría *cat*.

- $sim(c_1, c_2)$: es la función que devuelve la similitud numérica entre los conceptos c_1 y c_2, que es un valor $\in$ [0..1].

- $dom(c_1, c_2)$: devuelve el grado de encaje (*degree of match*) entre c_1 y c_2, que es el valor resultante de escalar el $sim(c_1, c_2)$ en el intervalo correspondiente a la categoría $com(c_1, c_2)$. Se puede definir como

  $$dom(c_1, c_2) = inf(com(c_1, c_2)) + sim(c_1, c_2) \cdot (sup(com(c_1, c_2)) - inf(com(c_1, c_2)))$$

La longitud de cada intervalo no es relevante, aunque lo más intuitivo es que tengan la misma dimensión. Así, si se tienen N categorías, y se reservan los valores 1 y 0 para las categorías 1 y N, a cada una de las demás les corresponde un intervalo de tamaño $\frac{1}{N-2}$. Es decir, para la categoría c_i le corresponde el intervalo $\left[\frac{N-i-1}{N-2}, \frac{N-i}{N-2}\right)$.

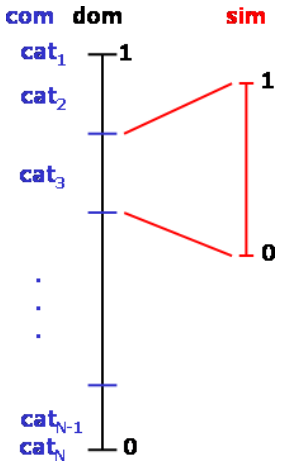


Figura 4.1: Escalado de la medida de similitud en el grado de encaje

En nuestro caso particular, $cat_1 = exact$, $cat_2 = plugin$, $cat_3 = subsumes$ y $cat_4 = fail$. Por lo que el rango de $plug - in$ es $[0.5, 1)$ y el de $subsumes$ es $(0, 0.5)$.

Tras hacer el escalado la función de equiparación queda como se define a continuación.

Definición 4 *El grado de equiparación entre un rol r_A de un anuncio y un rol r_Q de una solicitud de servicio está definido por la ecuación* (4.3)

$$dom(r_Q, r_A) = \begin{cases} 1 & \text{si } r_A = r_Q \\ \frac{1}{2} + \frac{1}{2 \cdot e^{\|r_A, r_Q\|}} & \text{si } r_A \prec r_Q \\ \frac{1}{2} \cdot e^{\|r_A, r_Q\|} & \text{si } r_Q \prec r_A \\ 0 & \text{en otro caso} \end{cases} \tag{4.3}$$

Donde $\| r_A, r_Q \| = depth(r_A) - depth(r_Q)$ representa la distancia (con signo) entre ambos roles. Obsérvese que es suficiente con restar sus profundidades ya que sólo se considera cuando están en la misma rama del árbol (hay relación de subsunción entre ellos). La figura 4.2 muestra gráficamente esta función.

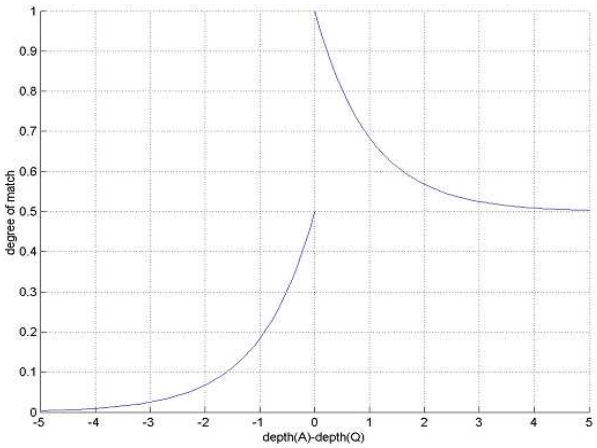


Figura 4.2: Función de match entre dos roles

Por construcción, $dom(r_Q, r_A)$ cumple los cinco requisitos propuestos. Cabe destacar que la figura permite comprobar visualmente cómo la relación de subsunción (requisito 5) determina el signo de $depth(r_A) - depth(r_Q)$ y por tanto si el resultado es mayor o menor de 0.5. También se puede observar que el incremento de una unidad de distancia para pasar de 0 a 1 provoca una reducción del *dom* mucho mayor que el paso de 1 a 2, y así sucesivamente (requisito 3). El resto de requisitos también se cumplen.

4.2. Equiparación entre servicios

El algoritmo desarrollado para la equiparación de servicios toma como entradas (i) una solicitud de servicio (Q), y (ii) un anuncio de servicio (S); y devuelve como resultado el grado de encaje o match (dom) entre ellos, que es un número real en el rango [0..1]. Esencialmente, busca el rol en el anuncio S que mejor encaja con el rol de la consulta Q, teniendo en cuenta, si existen, los

roles dependientes (*depending roles*) y capacidades del solicitante (*requester capability roles*).

En primer lugar se recuerda cuál es la descripción de un servicio anunciado (S) y una solicitud (R) desde el punto de vista de los roles, y en la notación que se facilita la descripción del algoritmo.

```
S = SET OF InteractiveRoleDescription
InteractiveRoleDescription = <ProviderRole, RoleExpression>

RoleExpression = SET OF ConjunctiveRoleList
ConjunctiveRoleList = SET OF Roles

Q = <SearchedRoles, RequesterCapabilities>
SearchedRoles = RoleExpression
RequesterCapabilities = SET OF Roles
```

La equiparación entre dos servicios consiste en un proceso de agregación de resultados de calcular el grado de encaje entre roles pertenecientes a la consulta y los que aparecen en las descripciones de los servicios anunciados.

Los requisitos que se desean de la función de equiparación de servicios son los siguientes:

1. Debe devolver un valor numérico en el rango real [0..1]. El motivo es el mismo que en la equiparación de roles, facilitar la comparación y ordenación en este caso entre los servicios. Aunque el rango podría ser cualquiera se ha elegido el intervalo [0..1] puesto que es muy intuitivo y ampliamente usado.

2. Al agregar valores procedentes del cálculo del grado de equiparación entre dos roles, el valor resultante debe mantenerse dentro del rango correspondiente al valor lógico que intuitivamente deba obtenerse. Por ejemplo, si se busca un servicio que juegue el rol r_1 y un servicio es capaz de jugar tanto r_2 como r_3, el resultado debe ser el mejor del encaje

de r_1 con cada uno de los otros, tanto en su valor lógico (*exact*, *plug-in*, *subsumes*,*fail*) como en el numérico (que debe pertenecer al rango asociado al valor de encaje lógico).

La figura 4.3 muestra el pseudocódigo de la función principal de equiparación (*Match*). Como se describió en el capítulo 3, la solicitud podría incluir no sólo un rol sino también una expresión lógica (una disyunción de conjunción de roles). Los bucles de las líneas 4 y 6 descomponen esa expresión, usando el mínimo como función de combinación para los valores en una conjunción y el máximo para disyunciones.

```
01 Match(Q: service request, S: service advertisement)
02 {
03   dom = 0
04   FOR ALL CRLi IN Q.SearchedRoles {
05     dom' = inf
06     FOR ALL rj IN CRLi {
07       dom' = min(dom',MatchAtomicRequest(rj,Q.RequesterCapabilities,S))
08     }
09     dom = max(dom, dom')
10   }
11   return dom
12 }
```

Figura 4.3: Función principal del algoritmo de match basado en roles. Devuelve el grado de match entre una solicitud Q y un anuncio S.

La función *MatchAtomicRequest* (figura 4.4) devuelve el grado de equiparación entre un rol de la consulta y todo un anuncio de servicio, dado el conjunto de capacidades comunicativas (roles que sabe jugar) del solicitante. Esta función compara el rol buscado con cada rol que puede jugar el proveedor del servicio y devuelve el mayor grado de encaje. Para cada interacción del anuncio, se calcula el encaje entre el rol buscado (*role*) y el rol del proveedor en esa interacción (*ProviderRole*), así como el encaje entre los roles requeridos al solicitante (*depending roles*) y sus capacidades. El mínimo de ambos valores es considerado el grado de match. De nuevo, los *depending roles* vienen dados mediante

una expresión lógica (en FND), que debe ser evaluada descomponiéndola en la función *MatchRoleExpr* (figura 4.5).

```
MatchAtomicRequest(role: Role, RequesterCapabilities: SET OF ROLES,
                   S: service advertisement)
{
  dom = 0
  FOR ALL IRDi IN S {
    dom1 = MatchRole(role,IRDi.ProviderRole)
    dom2 = MatchRoleExpr(IRDi.RoleExpression, RequesterCapabilities)
    dom = max(dom, min(dom1,dom2))
  }
  return dom
}
```

Figura 4.4: Función *MatchAtomicRequest*. Devuelve el grado de match entre un rol (de la solicitud) y los roles de un anuncio de servicio S.

Ejemplo Considérese, por ejemplo, el servicio S mostrado en la tabla 4.1 y una solicitud Q de servicio que especifica el rol buscado $((ClosedRequestee \wedge Explainer) \vee Notifier)$ y que el solicitante puede jugar los roles (*Requester Capability Roles*) *Informer* y *Explainer* .

Interacción	*provider role*	*depending roles*
I_1	$Advisor$	$Informer$
I_2	$Explainer$	
I_3	$Informer$	

Tabla 4.1: Ejemplo de anuncio de servicio

Formalmente, se trata de calcular el encaje entre el anuncio S y la consulta Q, siendo $S = \{< Advisor, Informer >, < Explainer, - >, < Informer, - >\}$ y
$Q =< (ClosedRequestee \wedge Explainer) \vee Notifier, \{Informer, Explainer\} >$
.

```
MatchRoleExpr(RExpression: SET OF ConjunctiveRoleList,
              RequesterCapabilities: SET OF Roles)
{
  dom = 0
  FOR ALL CRLi IN RExpression {
    dom' = inf
    FOR ALL rj IN CRLi {
      dom' = min(dom',MatchRoleInList(rj, RequesterCapabilities))
    }
    dom = max(dom, dom')
  }
  return dom
}
MatchRoleInList(role: Role, RL: SET OF Roles)
{
  dom = 0
  FOR ALL ri IN RL {
    dom = max(dom, MatchRole(role,ri))
  }
  return dom
}
```

Figura 4.5: Función MatchRoleExpr.

En primer lugar, el algoritmo comienza descomponiendo la expresión lógica en la solicitud y calcula el grado de encaje (*MatchAtomicRequest*) de cada uno de los roles de la expresión con el servicio S, teniendo en cuenta la capacidad ({*Informer, Explainer*}) del solicitante. Posteriormente se agregan los resultados usando el mínimo para la conjunción y el máximo para la disyunción.

1. ClosedRequestee: *MatchAtomicRequest(ClosedRequestee,*{*Informer,Explainer*}*,S)*

   Ahora se calcula en encaje entre el rol buscado (*ClosedRequestee*) con cada uno de los roles que podría jugar el servicio S: (i) *Advisor*, (ii) *Explainer*, (iii) *Informer*.

   a) Para el tipo de interacción I_1, el encaje entre el rol del proveedor *Advisor* y el rol buscado *ClosedRequestee* es un caso de *plugin* ($Advisor \prec ClosedRequestee$):

   $$dom(ClosedRequestee, Advisor) = \frac{1}{2} + \frac{1}{2 \cdot e^1} = 0.68 \qquad (4.4)$$

   ya que $||Advisor, ClosedRequestee|| = +1$, basándose en la ontología definida en la figura 3.4.

   Además, el proveedor declara un rol dependiente (*Informer*) y el solicitante incluye ese rol en sus capacidades, por lo que que hay un encaje *exacto* ($dom(Informer, Informer) = 1$) para la sección de *depending roles* (el encaje con su otra capacidad sería $dom(Informer, Explainer) = 0$, *fail*, pero el máximo es elegido).

   Finalmente, el grado de encaje para la interacción I_1 es el mínimo entre el encaje del rol proporcionado y el dependiente: $\text{mín}(0.68, 1) = 0.68$

   b) Para el tipo de interacción I_2, el encaje entre el rol del proveedor *Explainer* y el rol buscado *ClosedRequestee* es:

   $$dom(ClosedRequestee, Explainer) = \frac{1}{2} + \frac{1}{2 \cdot e^1} = 0.68 \qquad (4.5)$$

   En este caso no se declara ningún rol dependiente, por lo que el resultado es 0.68.

c) Para el tipo de interacción I_3, el encaje entre el rol del proveedor *Informer* y el rol buscado *ClosedRequestee* es:

$$dom(ClosedRequestee, Informer) = \frac{1}{2} + \frac{1}{2 \cdot e^1} = 0.68 \qquad (4.6)$$

En este caso tampoco se declara ningún rol dependiente, por lo que el resultado es 0.68.

El resultado final para el primer rol de la expresión buscada es máximo de los tres valores, es decir 0.68. Obsérvese que este resultado se corresponde con la clase *plug-in* (rango [0.5..1]), que corresponde al resultado que se obtendría si se analizase desde el punto de vista puramente lógico solamente.

Se repite ahora el proceso con los otros dos roles de la expresión.

2. Explainer: *MatchAtomicRequest(Explainer,{Informer,Explainer},S)*

   a) I_1: $dom(Explainer, Advisor) = 0$ (*fail*)

   b) I_2: $dom(Explainer, Explainer) = 1$ (*exact*)

   c) I_3: $dom(Explainer, Informer) = 0$ (*fail*)
   máx$(0, 1, 0) = 1$

3. Notifier: *MatchAtomicRequest(Notifier,{Informer,Explainer},S)*

   a) I_1: $dom(Notifier, Advisor) = 0$ (*fail*)

   b) I_2: $dom(Notifier, Explainer) = 0$ (*exact*)

   c) I_3: $dom(Notifier, Informer) = \frac{1}{2} \cdot e^{-1} = 0.18$ (*fail*)
   máx$(0, 0, 0.18) = 0.18$

El grado total de encaje entre los dos servicios se calcula agregando los resultados obtenidos para componer la expresión lógica usando el mínimo para las conjunciones y el máximo para las disyunciones, en este caso

$$Match(Q, S) = \text{máx}(\text{mín}(0.68, 0.68), 0.18) = 0.68 \text{ (\textit{plug-in})} \qquad (4.7)$$

4.3. Implementación

El algoritmo descrito en este capítulo ha sido implementado en un componente software (denominado ROWLS). La implementación se apoya en la biblioteca Mindswap Java Library[1] para el parseo de las descripciones de servicios OWL-S, y en Jena[2] para la gestión de ontologías OWL.

ROWLS toma una descripción de servicio OWL-S como consulta, y devuelve un conjunto ordenado de servicios relevantes que encajan con la consulta, cada uno de ellos con un grado determinado (valor entre 0 y 1).

La clase principal (figura 4.6) es *MatchmakingEngine*, que se encarga de almacenar los servicios anunciados y con los que se calculará en encaje. *MatchmakingEngine* usa un *RolesReader* para leer los ficheros OWL-S y extraer la estructura de roles, tanto de los anuncios como de la consulta, y ejecuta la equiparación semántica entre dos conceptos mediante un *SemanticMatchmaker*. Este diseño admite mediante herencia la implementación de diferentes clases de lectura (implementando la interfaz *RolesReader*) o de algoritmos de matching entre conceptos (heredando de *SemanticMatchmaker*). En concreto, se ha implementado un *RolesReader* que usa la API de Mindswap para OWL-S[3] para manipular los ficheros OWL-S y un *SemanticMatchmaker* que navega en la ontología para calcular la distancia (número de arcos) entre conceptos e implementa la ecuación (4.3).

Puesto que el enfoque propuesto en esta tesis es una extensión a otras propuestas, el componente desarrollado (ROWLS) normalmente no debe concebirse como un elemento aislado sino complementario a otro matchmaker de propósito general. En nuestro caso, ROWLS se ha combinado con OWLS-MX [67] (sección 2.2.3.5), uno de los matchmakers más reconocidos en la actualidad. OWLS-MX es un matchmaker híbrido para servicios Web semánticos que complementa un razonamiento basado en lógica con técnicas aproximadas de cálculo de similaridad sintáctica provenientes del campo de la recuperación de

[1] http://www.mindswap.org/ mhgrove/kowari
[2] http://jena.sourceforge.net
[3] http://www.mindswap.org/2004/owl-s/api

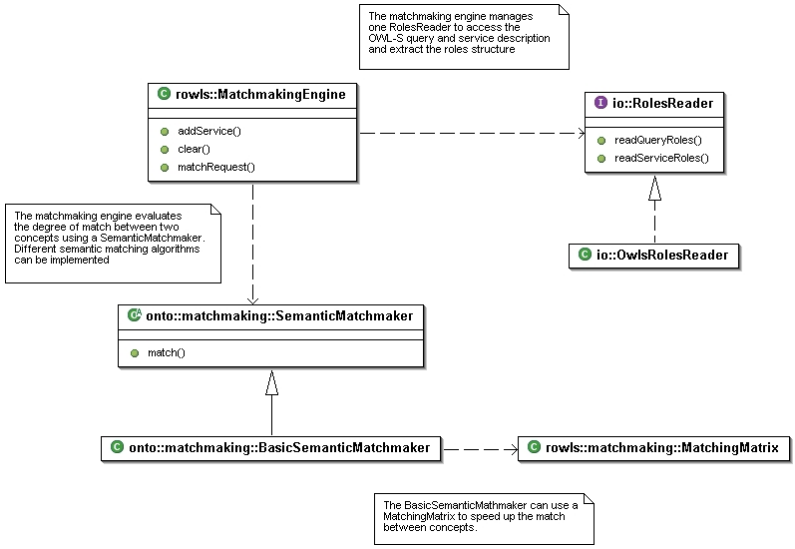


Figura 4.6: Diagrama de clases del matchmaker basado en roles (ROWLS)

información.

La integración se ha llevado a cabo de modo que ROWLS actúa como filtro previo a OWLS-MX, ya que el tiempo de proceso es muy inferior a OWLS-MX. La figura 4.7 muestra la arquitectura. De esta manera, todos los servicios de entrada son analizados por ROWLS y ordenados de mayor a menor nivel de encaje. A continuación, sólo los mejores son pasados a OWLS-MX, por lo que se reduce el tiempo empleado por este último al tener que analizar menos servicios. Finalmente los servicios son ordenados según el criterio de OWLS-MX.

Puesto que ambos matchmakers tienen que cargar las descripciones de servicios a partir de los ficheros, la integración se ha realizado compartiendo las estructuras internas, de forma que la carga sólo tiene que realizarse una vez, lo cual ahorra bastante tiempo (a costa de tener un código más acoplado).

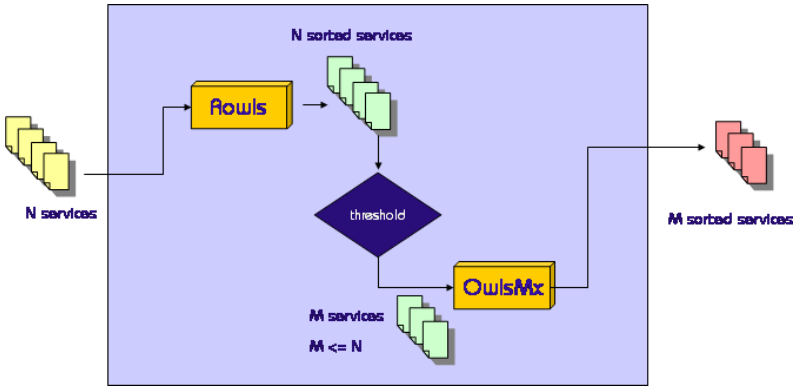


Figura 4.7: Integración de ROWLS con OWLS-MX

4.4. Evaluación

En esta sección se analiza el rendimiento del enfoque de equiparación de servicios propuesto en este trabajo.

En sistemas de clasificación de contenidos suelen evaluarse dos clases de medidas:

- *Eficiencia*: recursos de espacio y/o tiempo consumidos
- *Efectividad*: calidad de la respuesta obtenida

A continuación se realizan algunas consideraciones sobre el conjunto de casos de test utilizados en las pruebas. Posteriormente se analiza en profundidad la eficiencia y la efectividad del componente desarrollado.

4.4.1. Casos de test

La preparación del conjunto de casos de test para realizar los experimentos no ha sido sencilla.

La primera dificultad se encuentra en la no existencia de una colección de prueba estándar de descripciones de servicios en OWL-S (ni en general de

servicios Web semánticos). En realidad, debido a la inmadurez de esta tecnología todavía no se ha generado dicho banco de pruebas con descripciones de servicios Web semánticos y consultas (con sus correspondientes conjuntos de servicios relevantes). Normalmente se utilizan adaptaciones semiautomáticas de servicios WSDL (sintácticos), de los que sí hay algo más de experiencia. Actualmente existe una iniciativa (promovida por Matthias Klusch[4], del DFKI) encaminada a la construcción de esa colección de test que permita comparar los diferentes matchmakers construidos, de forma similar a lo ocurrido hace ya años en las series de conferencias TREC[5] en el campo de la *recuperación de la información*.

Pero además de esa dificultad, que puede ser solventada en mayor o menor medida utilizando servicios y/o algunas de las colecciones disponibles (aunque no ideales), en nuestro caso se añade la necesidad de adaptar las descripciones y consultas (y conjuntos de servicios relevantes) para incluir en ellas la información organizacional (roles) propuesta en este trabajo.

En concreto, se ha decidido utilizar como base un subconjunto de la colección OWLS-TCv2[6], quizá la más completa en el momento de realizar los experimentos[7]. Esta colección contiene 576 servicios que cubren 7 dominios. Los servicios y consultas seleccionados fueron anotados con *roles*. Además, puesto que se observó un desequilibrio en los roles jugados por los servicios de OWLS-TC (debido a su origen y aplicación la mayoría son *informers*), se complementó con nuevos servicios y consultas más ricas en cuanto al enfoque propuesto en este trabajo, así como la replicación de algunos servicios originales (no *informers*). Finalmente, el conjunto de prueba estaba formado por 378 descripciones de servicios OWL-S. La técnica de replicación de servicios fue también utilizada para realizar pruebas de eficiencia.

Como se ha comentado con anterioridad, el componente desarrollado (ROWLS) no debe concebirse como un elemento aislado y se ha combinado con

[4]http://www-ags.dfki.uni-sb.de/∼klusch/
[5]http://trec.nist.gov/
[6]http://projects.semwebcentral.org/projects/owls-tc/
[7]En noviembre de 2007 ha sido publicada la versión 2.2

OWLS-MX [67], un matchmaker de propósito general. Muchos de los resultados mostrados a continuación comparan las características de la combinación de ROWLS y OWLS-MX frente al uso de este último únicamente.

4.4.2. Evaluación de la eficiencia

La evaluación de la eficiencia realizada consiste en analizar el tiempo medio de cómputo en calcular el grado de equiparación entre una consulta y el conjunto de descripciones de servicio disponibles, así como la escalabilidad del método desarrollado. En primer lugar se realiza un análisis teórico para posteriormente analizar empíricamente con un conjunto de casos de prueba el rendimiento del componente software construido a partir del algoritmo desarrollado.

4.4.2.1. Evaluación Teórica

En esta sección se realiza un análisis desde un punto de vista teórico de la eficiencia del método propuesto en este capítulo, tanto de forma individual como en combinación con otros métodos complementarios.

En el siguiente análisis se tomarán las siguientes notaciones:

$S = \{s_1, s_2, ..., s_n\}$: conjunto de descripciones de servicios que forman el directorio, es decir, servicios entre los que se buscan los relevantes para una determinada consulta.

$T_c(s)$: tiempo de carga de la descripción del servicio s. El tiempo de carga viene determinado no sólo por el parseo del fichero sino también por otros factores del entorno como el tiempo de acceso a disco, red, resolución de urls de ontologías, etc.

$T_m(s, q)$: tiempo de equiparación ($match$) del servicio s con la consulta q. Este tiempo no incluye el tiempo de carga.

$T(q)$: tiempo total en procesar la consulta q.

$$T(q) = T_c(q) + \sum_{i=1}^{n}(T_c(s_i) + T_m(s_i, q)) = T_c(q) + \sum_{i=1}^{n} T_c(s_i) + \sum_{i=1}^{n} T_m(s_i, q) \quad (4.8)$$

Podemos aproximar la ecuación anterior suponiendo un tiempo medio de carga de una descripción de servicio (t_c) y un tiempo medio de match (t_m). Entonces la ecuación anterior queda

$$T(q) = t_c + \sum_{i=1}^{n} t_c + \sum_{i=1}^{n} t_m = t_c + n \cdot t_c + n \cdot t_m \approx n \cdot t_c + n \cdot t_m \quad (4.9)$$

La última aproximación se ha realizado ya que n suele ser más o menos grande por lo que considerar un servicio más o uno menos (la consulta) no es relevante.

Dependiendo de la arquitectura del sistema el tiempo de carga (t_c) puede ser o no relevante. Así, si el matchmaker está integrado en el directorio, el tiempo de carga se podría despreciar ya que las descripciones de servicios se cargarían una sola vez al registrar los servicios en el directorio. De esta manera para una nueva consulta q sólo habría que cargar esa descripción. En este caso,

$$T(q) = t_c + n \cdot t_m \quad (4.10)$$

En cambio, si se sigue un enfoque en el que el matchmaker recibe tanto la consulta como la lista de potenciales candidatos (o debe acceder a sus descripciones para cada consulta), entonces sí hay que tener en cuenta el tiempo de carga.

Como ya se comentó anteriormente, el enfoque de matching organizacional propuesto en este trabajo no debe concebirse de forma aislada sino que se supone combinado con otro(s) componente(s) que, a su vez, se enfoquen en otras características de los servicios (p. ej. entradas, salidas, precondiciones, efectos, ...).

Suponiendo que existen varios (l) matchmakers MM_1, MM_2, ..., MM_l, cada uno de los cuales obtiene el grado de equiparación correspondiente a su enfoque

$(dom_i(q,s))$, entonces el resultado es una combinación de esos valores:

$$dom(q,s) = dom_1(q,s) \oplus dom_2(q,s) \oplus ... \oplus dom_l(q,s) \tag{4.11}$$

siendo $\oplus$ una función de combinación.

Se analiza a continuación cuál es el efecto de combinar dos mecanismos de equiparación. En este caso aplicando la ecuación (4.9), considerando que el tiempo de carga es el mismo en ambos casos y sus tiempos de equiparación son t_m^1 y t_m^2:

$$T(q) = n \cdot t_c + n \cdot t_m^1 + n \cdot t_c + n \cdot t_m^2 = 2 \cdot n \cdot t_c + n \cdot t_m^1 + n \cdot t_m^2 \tag{4.12}$$

En cambio, hay situaciones en las que la función de combinación $\oplus$ se puede aplicar de forma secuencial. Ejemplos de funciones pueden ser el operador lógico y, o en general situaciones en las que para considerar que un servicio puede equiparar con una consulta debe cumplirse que supere un umbral para ambos mecanismos. En estas situaciones, es más eficiente combinar los métodos de forma que sólo los (k) servicios que satisfagan uno de los matchmakers son evaluados por el otro, es decir, el primero de ellos actúa como un filtro para el segundo. En este caso, el tiempo de procesamiento de la consulta será:

$$T(q) = n \cdot t_c + n \cdot t_m^1 + k \cdot t_c + k \cdot t_m^2 \tag{4.13}$$

donde n es el número total de servicios y k el número de servicios que pasan el filtro del primer matchmaker.

En este caso, además del valor añadido en cuanto a la eficacia del método por considerar varios criterios, se puede conseguir una mejora en el rendimiento (tiempo de cómputo), lo cual es un aspecto importante (e incluso puede que imprescindible) si el número de servicios a considerar es grande, algo que es de esperar en entornos abiertos como a los que va dirigido este trabajo.

Entonces, desde un punto de vista computacional, nos podemos plantear cuál debería ser el número de servicios k máximo que debería pasar el filtro para que el rendimiento del sistema fuera mejor que sin filtro, esto es, al añadir ese componente adicional no empeora el rendimiento. Esto se calcula mediante la siguiente desigualdad

$$\begin{aligned} n \cdot t_c + n \cdot t_m^1 + k \cdot t_c + k \cdot t_m^2 &\leq n \cdot t_c + n \cdot t_m^2 \\ k \cdot (t_c + t_m^2) &\leq n \cdot (t_m^2 - t_m^1) \\ k &\leq n \cdot \frac{(t_m^2 - t_m^1)}{(t_c + t_m^2)} \end{aligned}$$

O en términos relativos, la proporción de servicios máxima que deben pasar el filtro para que mejore la eficiencia será:

$$p = \frac{k}{n} \leq \frac{(t_m^2 - t_m^1)}{(t_c + t_m^2)} \tag{4.14}$$

Así, por ejemplo, si $p = 0.8$ significa que si el filtro deja pasar el 80 % o menos de los servicios (filtra el 20 % o más), entonces la eficiencia del sistema conjunto es mejor que sin filtro.

La mejora podría aumentarse si no consideramos el tiempo de carga de las descripciones de servicios, o si es posible integrar ambos componentes de forma que la carga se realice una única vez. En este caso tenemos

$$n \cdot t_c + n \cdot t_m^1 + k \cdot t_m^2 \leq n \cdot t_c + n \cdot t_m^2 \tag{4.15}$$

$$n \cdot t_m^1 + k \cdot t_m^2 \leq n \cdot t_m^2 \tag{4.16}$$

$$k \cdot t_m^2 \leq n \cdot (t_m^2 - t_m^1) \tag{4.17}$$

$$k \leq n \cdot \frac{(t_m^2 - t_m^1)}{t_m^2} \tag{4.18}$$

O en proporción:

$$p = \frac{k}{n} \leq \frac{(t_m^2 - t_m^1)}{t_m^2} \tag{4.19}$$

Obsérvese que si no se considera t_c el resultado es el mismo. Por supuesto, en todos los casos se asume que el componente más eficiente actúa como filtro (de lo contrario p sería negativo).

Las tablas 4.2 y 4.3 resumen las ecuaciones obtenidos en el análisis precedente.

Configuración	con carga	sin carga
MM_1	$n \cdot t_c + n \cdot t_m$	$n \cdot t_m$
MM_1 + MM_2	$2 \cdot n \cdot t_c + n \cdot t_m^1 + n \cdot t_m^2$	$n \cdot t_m^1 + n \cdot t_m^2$
Prefiltro (no integrado)	$(n+k) \cdot t_c + n \cdot t_m^1 + k \cdot t_m^2$	$n \cdot t_m^1 + k \cdot t_m^2$
Prefiltro (integrado)	$n \cdot t_c + n \cdot t_m^1 + k \cdot t_m^2$	$n \cdot t_m^1 + k \cdot t_m^2$

Tabla 4.2: Resumen tiempos de proceso de consultas ($T(q)$)

Integración de los componentes	con carga	sin carga
No integrados	$p = \frac{k}{n} \leq \frac{(t_m^2 - t_m^1)}{(t_c + t_m^2)}$	$p = \frac{k}{n} \leq \frac{(t_m^2 - t_m^1)}{t_m^2}$
Integrados	$p = \frac{k}{n} \leq \frac{(t_m^2 - t_m^1)}{t_m^2}$	

Tabla 4.3: Resumen proporción de servicios filtrados para mejorar la eficiencia en secuencia de matchmakers

4.4.2.2. Evaluación Empírica

Los objetivos principales perseguidos mediante la evaluación empírica de la eficiencia son los siguientes:

- Analizar el tiempo medio de proceso de consultas por el componente desarrollado (ROWLS), observando cómo afecta el tiempo de carga, es decir si es despreciable o viceversa.
- Analizar al escalabilidad del componente, es decir, cómo se comporta frente a un número elevado de descripciones de servicios a procesar.

- Comparar la eficiencia con otros matchmakers (en concreto con OWLS-MX).
- Analizar la mejor manera de combinarlo con otro matchmaker de forma que sea más eficiente, si es posible, que sólo uno de ellos.

La figura 4.8 muestra los resultados de los tests de escalabilidad realizados (se utilizó un Intel Pentium 4 a 3.00 GHz con 1GB de RAM). Se observa que el tiempo empleado en el match aumenta linealmente con el número de servicios, dando un tiempo medio de match entre una consulta y un servicio (t_m) de 0.03 ms. Esto es dos órdenes de magnitud inferior al tiempo empleado por OWLS-MX. El motivo es que ROWLS se concentra en menos características de una descripción de servicio, en concreto sólo tiene en cuenta el parámetro que incluye los roles. Además, la ontología usada en ROWLS es, en general, más pequeña en tamaño que las ontologías del dominio que tiene que manejar OWLS-MX.

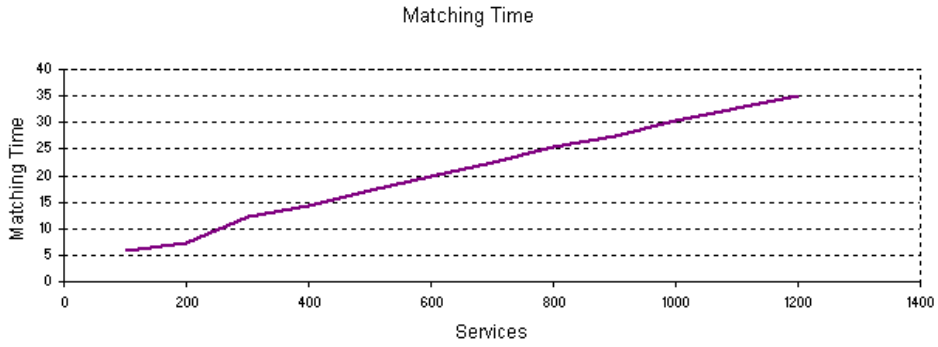


Figura 4.8: Tiempo de match de ROWLS (ms)

El tiempo de carga (t_c) no se ha considerado ya que depende de varios aspectos externos como tiempo de acceso a disco, conexiones de red, servidores web, etc. Téngase en cuenta que la carga de una descripción de servicio (fichero OWL-S) normalmente implica el acceso (vía Internet) y parseo de varias ontologías. En las pruebas realizadas se obtuvo un tiempo de carga medio (t_c) entre 100 y 200 ms. por servicio (difícil de dar un valor más preciso debido a la variabilidad

obtenida por las causas mencionadas), lo cual es mucho más alto que el tiempo de match por servicio.

Esta diferencia entre el tiempo de carga y el tiempo de match ($t_c \gg t_m$) provoca dos importantes decisiones arquitecturales:

1. Para combinar ambos matchmakers debe realizarse una integración fuerte entre ambos, de forma que se comparta la representación interna de los servicios para evitar tener que realizar la carga dos veces.

2. Es recomendable integrar los matchmakers con el directorio de forma que las descripciones de servicios son cargadas sólo una vez al registrar el servicio y no por cada consulta realizada. Otra posibilidad es implementar algún mecanismo de *catching* que mantenga los servicios ya cargados en el pasado.

Si los matchmakers están integrados, la proporción máxima de servicios que debe pasar el filtro para que la eficiencia mejore (ecuación (4.19)) es 0.99 ya que, como se comentó previamente, el tiempo de match de ROWLS es dos órdenes de magnitud inferior al de OWLS-MX. Esto significa que con sólo eliminar el 1 % de los servicios ya el rendimiento mejora.

4.4.3. Evaluación de la efectividad

Para evaluar la *efectividad* se adoptan técnicas de evaluación ampliamente utilizadas en el campo de la *recuperación de información*[8] [6, 120]. Existen varias medidas para evaluar la *efectividad*. Las dos más utilizadas son:

- la *precisión (precision)*: proporción de los documentos obtenidos que son relevantes[9]

[8] *information retrieval (o IR)*

[9] En nuestro caso los documentos relevantes son las descripciones de los servicios que permiten cumplir la consulta

- la *recuperación*[10] *(recall)*: proporción de los documentos relevantes que son obtenidos

Consideremos una consulta $q \in Q$, siendo Q el conjunto de todas las consultas.

Sea A_q el conjunto de documentos recuperados por el clasificador para la consulta q.

Sea R_q el conjunto de documentos relevantes para la consulta q (la respuesta ideal).

Sean $|A_q|$ y $|R_q|$ las cardinalidades de esos conjuntos.

Entonces, la *precisión* π_q para la consulta q se calcula como la proporción de documentos relevantes recuperados respecto a todos los documentos recuperados:

$$\pi_q = \frac{|A_q \cap R_q|}{|A_q|} \tag{4.20}$$

mientras que la *recuperación* ρ_q es

$$\rho_q = \frac{|A_q \cap R_q|}{|R_q|} \tag{4.21}$$

Si el clasificador admite varias clases (consultas) entonces para estimar la precisión (π) y la recuperación (ρ) global se pueden adoptar dos métodos diferentes:

- *Microaveraging*: consiste en sumar los valores de cada categoría y a partir de ellos calcular la precisión (recuperación) global.

$$\pi = \frac{\sum_{q \in Q} |A_q \cap R_q|}{\sum_{q \in Q} |A_q|} \tag{4.22}$$

[10] o exhaustividad

$$\rho = \frac{\sum_{q \in Q} |A_q \cap R_q|}{\sum_{q \in Q} |R_q|} \tag{4.23}$$

- *Macroaveraging*: consiste en calcular los valores de precisión (recuperación) para cada categoría y luego calcular la media.

$$\pi = \frac{\sum_{q \in Q} \pi_q}{|Q|} \tag{4.24}$$

$$\rho = \frac{\sum_{q \in Q} \rho_q}{|Q|} \tag{4.25}$$

La elección de cuál de los anteriores métodos utilizar depende del caso concreto. Por ejemplo, si se desea evaluar el buen comportamiento del clasificador para categorías pequeñas (pocos documentos) entonces es más adecuado usar *macroaveraging*.

Estas dos medidas (*precisión/recuperación*) varían de forma inversa. Para optimizar la precisión se puede reducir el conjunto de documentos devueltos, lo que reduciría la recuperación (el caso extremo sería devolver sólo el primer documento, que se espera que sea relevante). En cambio si aumentamos los documentos devueltos, la recuperación aumentará pero la precisión bajará (el caso extremo sería devolver todos los documentos, logrando una recuperación $\rho = 1$ pero con baja precisión).

Por este motivo, habitualmente ambas medidas no se toman por separado sino en combinación. Muchas veces, los sistemas de recuperación devuelven los documentos recuperados en orden de mayor a menor relevancia para la consulta. En estos casos, para calcular la precisión/recuperación hay que decidir en qué puntos se toman los datos.

Existen varias formas de combinar precisión con la recuperación. Lo más habitual son las curvas precisión/recuperación. Se suele usar una curva en la que

en el eje de abscisas se muestra la *recuperación* y en las ordenadas la *precisión*. Un punto (p, r) de la curva indica un nivel de precisión p para un nivel de recuperación r. Para construir esa curva los datos se pueden obtener de diferente manera:

- Normalmente la curva se basa en *11 niveles estándar de recuperación*: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0. Se calcula la precisión para cada uno de esos niveles de recuperación.
- *niveles de corte* (cutoff levels): se calcula la precisión tras recuperar un número determinado de documentos. Por ejemplo, tras 5 documentos, tras 10, tras 20, tras 50, tras 100, tras 500,... Esta medida es especialmente útil cuando hay muchos documentos relevantes pero realmente se está interesado en obtener unos pocos, como es el caso de la búsqueda en Internet.
- Se mide la precisión cada vez que se recupera un nuevo documento relevante.

También hay medidas que dan un valor combinado de precisión/recuperación. Algunas son:

- *Niveles de corte* (cutoff levels): en lugar de dibujar una curva se calcula la precisión y recuperación medios como una media (ponderada) de esos valores.
- *Precisión promedia*: se mide la precisión cada vez que se recupera un nuevo documento relevante. Se calcula la media de esos valores.
- *Breakeven point*: se calcula de forma similar a las curvas precisión/recuperación. El punto *breakeven* es aquél en el que la precisión iguala al recuperación.
- *Longitud de búsqueda esperada*: es la posición del primer documento relevante recuperado, o cuántos documentos no relevantes se recuperan hasta que se consiguen n documentos relevantes.

- *Precisión-R*: Sea r el número total de documentos relevantes (es decir, la respuesta ideal). Se consideran los r primeros documentos recuperados. La precisión-R es la precisión de este conjunto.

- *Medida-F*(Media Armónica): se define como:

$$F = \frac{2\pi\rho}{\pi + \rho} \tag{4.26}$$

  Esta medida favorece el compromiso entre precisión y recuperación. Vale 0 si alguna es 0, 1 si ambas son 1. También se puede definir para cada documento recuperado.

- *Medida-E*:

$$E = 1 - \frac{b^2\pi\rho + \pi\rho}{b^2\pi + \rho} \tag{4.27}$$

  b es un parámetro que permite controlar la importancia que se da a la precisión y a la recuperación:

  - $b = 1$: la medida-E es simplemente el complemento de la media armónica.
  - $b > 1$: se da más importancia a la precisión
  - $b < 1$: se da más importancia a la recuperación

- *Intersección vs Unión*:

$$\frac{|A \cap R|}{|A \cup R|} = \frac{\pi\rho}{\pi + \rho - \pi\rho} \tag{4.28}$$

Para reunir los resultados de varias categorías (o consultas) hay que adoptar macro- o microaveraging para cada nivel de recuperación.

Según cada caso concreto puede ser interesante optimizar alguna de esas dos medidas. Por ejemplo, si el conjunto de documentos es muy grande y sólo se desea obtener uno o unos pocos entonces es más importante la precisión. Esto es lo que ocurre, por ejemplo, en recuperación de documentos en internet.

El enfoque presentado en este trabajo va dirigido a sistemas abiertos donde existen gran cantidad de servicios proporcionados por agentes. En este caso, cuando un agente busca un proveedor capaz de proporcionar un determinado servicio, el solicitante está interesado en un número reducido (uno o quizá varios) de proveedores. Por este motivo preferiremos dar más importancia a la precisión que a la recuperación.

Además, por las características de nuestro enfoque es posible que la secuencia ordenada de servicios devueltos por el matchmaker esté clasificada por niveles con el mismo grado de match. El motivo es que, aunque se admiten expresiones lógicas para especificar los roles buscados, lo más habitual es que se especifique sólo uno o una expresión con pocos términos. Por ejemplo, si se busca un *notifier* todos los servicios cuyo rol más próximo a la consulta sea *informer* tendrán el mismo grado de match. Entonces, el resultado devuelto por el algoritmo es un orden parcial pues no hay orden entre los elementos con un mismo grado de match. Los valores de precisión/recuperación con estos datos serían diferentes dependiendo del orden en el que se consideren servicios con el mismo grado de match. En este caso se calcula la media, o lo que es lo mismo, se consideran equidistribuidos en cada nivel los servicios relevantes y los no relevantes.

4.4.3.1. Experimentación

Una cuestión relevante es hasta qué punto el uso de ROWLS puede mejorar, en términos de efectividad, los resultados de un matchmaker de propósito general. En particular, debido a que ROWLS es mucho más rápido que OWLS-MX, estábamos interesados en medir el rendimiento de una arquitectura en la que ROWLS es usado como un filtro que elimina un conjunto de servicios que no son pasados como entrada a OWLS-MX. De esta forma los servicios que no superan el umbral de encaje establecido para ROWLS son descartados directamente sin necesidad de comprobar el resto de características. Además ROWLS puede parametrizarse para devolver, entre los que superan el umbral mínimo o un porcentaje de servicios. Recuérdese que los servicios se ordenan

por su grado de encaje, por lo que en caso de no devolver el 100 % de los que superen el umbral sí se pasarán los mejores.

Para este propósito, los tests fueron realizados con diferentes configuraciones del filtro ROWLS que dejan pasar a OWLS-MX sólo un cierto porcentaje de los mejores servicios (de acuerdo con la equiparación basada en roles). En concreto se han realizado pruebas que dejan pasar el 30 %, 60 %, 90 % y 100 % de los servicios que ROWLS considera relevantes en algún grado (no *fail*).

Puesto que se han usado varias consultas distintas para realizar los experimentos es necesario combinar los valores obtenidos para cada una de las consultas. Se ha optado por usar la técnica de *macroaveraging* ya que considera igual de importantes todas las consultas.

La figura 4.9 muestra las *curvas precisión-recuperación* para OWLS-MX y para las cuatro combinaciones de ROWLS y OWLS-MX comentadas anteriormente. Como se puede observar en la figura, el uso de ROWLS como filtro del matchmaker de propósito general OWLS-MX lleva a una mejor *precisión* comparado con sólo OWLS-MX para todos los valores de *recuperación*.

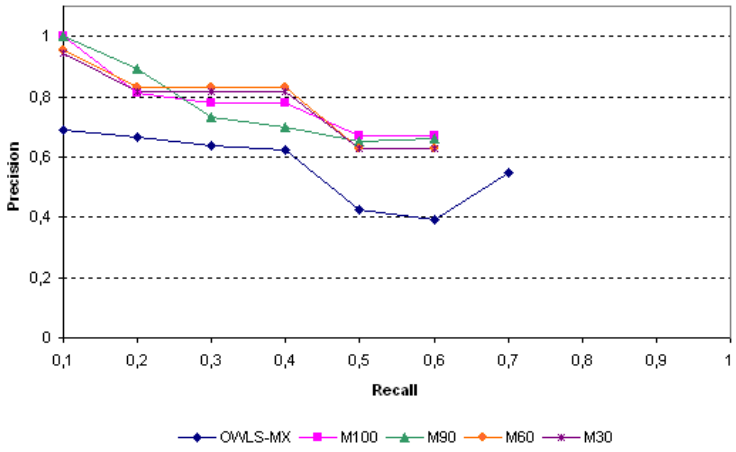


Figura 4.9: Curvas precisión-recuperación

En sistemas multiagente abiertos con miles de servicios, como es el tipo de

dominio de este trabajo, la *precisión* es más relevante que la *recuperación*: estamos interesados en encontrar uno o un conjunto reducido de proveedores de un servicio, pero no es necesario recuperar todos o casi todos. Por esto no nos centramos únicamente en las *curvas precisión-recuperación* sino que también se han tenido en cuenta otras medidas alternativas utilizadas en las series de conferencias TREC como la *precisión promedia* y la *precisión-R*.

La *precisión promedia* se muestra en la figura 4.10. Se puede ver que la combinación de OWLS-MX con ROWLS mejora el rendimiento de OWLS-MX. Además también se observa que cuantos menos servicios pasan el filtro (ROWLS) mayor es la precisión promedia conjunta. Esto es así porque el encaje basado en roles de ROWLS es ortogonal a la estrategia general de match seguida por OWLS-MX, y permite descartar servicios no relevantes a la consulta que (erróneamente) OWLS-MX habría clasificado como relevantes.

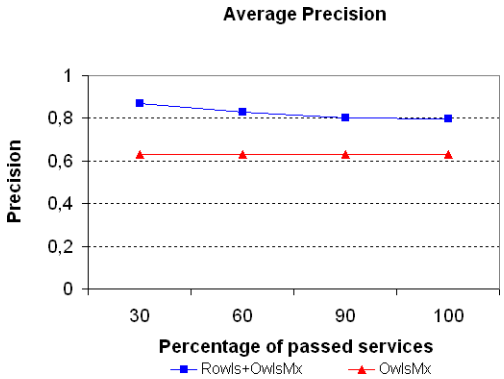


Figura 4.10: Precisión promedia

La figura 4.11 muestra los valores de *precisión-R*[11]. De nuevo, ROWLS mejora el comportamiento de OWLS-MX. En este caso se observa que la *precisión-R* aumenta con el número de servicios que pasan el filtro. El motivo es que sólo los mejores servicios pasan el filtro ROWLS, y cuantos más servicios tiene en cuenta OWLS-MX mayor es la probabilidad de que entre ellos se encuentren servicios irrelevantes.

[11]Precisión tras recuperar R servicios, siendo R el número de servicios relevantes de la colección de prueba

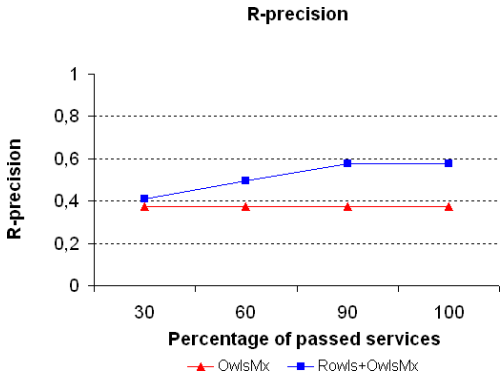


Figura 4.11: Precisión-R

4.5. Discusión

En este capítulo se ha propuesto un algoritmo de equiparación que hace uso de la información organizacional incluida en las descripciones de servicios según la propuesta del capítulo 3. El algoritmo obtiene un valor numérico que representa el grado con el que el servicio en cuestión equipara con la descripción de servicio deseado, todo ello en términos de roles jugados en las interacciones.

El método propuesto combina ideas proveniente de los enfoques actuales sobre equiparación de servicios web semánticos, en especial los descritos en el lenguaje OWL-S, con similitud entre conceptos en ontologías. En primer lugar se considera la relación lógica de subsunción entre conceptos y según esa relación se clasifica el servicio en un grado de equiparación. En este aspecto se ha seguido la propuesta de Paolucci [92]: *exact*, *plugin*, *subsumes* o *fail*. Otros enfoques incluyen algún grado adicional, como [67] y [73], por destacar los más relevantes.

Sin embargo, a diferencia de Paolucci y la mayoría de propuestas actuales, a cada servicio se le asocia un valor numérico, para lo que se usan ideas provenientes del campo de similitud entre conceptos, en concreto se tiene en cuenta principalmente la distancia entre los conceptos (propuesta por Rada et al. [96]). Para ello se ha asignado un intervalo de valores a cada categoría: 1

(*exact*), [0.5-1] (*plug-in*), [0-0.5] (*subsumes*) y 0 (*fail*).

Si bien OWLS-MX también utiliza medidas de similitud (técnicas de IR), lo hace en determinados casos y en combinación con cierto umbral para determinar si clasifica el concepto en cierto grado de equiparación, pero no es usado para calcular el grado de equiparación final.

El hecho de utilizar un valor numérico como resultado en lugar de únicamente el tipo de match (como Paolucci u OWLS-MX, por ejemplo) permite un ordenamiento más preciso de los servicios según su encaje con la consulta, lo que puede facilitar la selección de servicios.

La agregación de los valores de encaje entre roles, se realiza utilizando la funciones mínimo y máximo en el caso de conjunciones y disyunciones donde se permite una fórmula lógica, así como en los casos donde intuitivamente debe considerarse el mejor y peor caso (por ejemplo, cuando el proveedor puede participar en varias interacciones el mejor encaje es elegido). En este sentido, también se siguen los enfoques actuales donde, por ejemplo, si se busca un servicio con una determinada salida, los enfoques anteriormente descritos comparan esa salida con las proporcionadas y se elige la que mejor encaje tenga.

Se ha tenido cuidado de escoger funciones de agregación (*max/min*) que permiten mantener los valores numéricos dentro de las categorías de valores adecuadas, de forma que el refino propuesto es una generalización del enfoque de Paolucci.

Podrían considerarse otras funciones de agregación. Funciones utilizadas en lógica borrosa[12] [3, 66], podrían parecer buenas candidatas, ya que manejan valores numéricos en [0..1] aplicados en fórmulas lógicas. Sin embargo, éstas no son adecuadas, salvo precisamente el *min/max* (las más utilizadas, por cierto). El motivo es que los operadores de agregación utilizados para conjunciones (normas triangulares o t-normas) y disyunciones (conormas triangulares o t-conormas) pueden obtener un valor numérico agregado que se traspase a otra

[12] difusa o fuzzy logic

categoría. En efecto, teniendo en cuenta que la función *mínimo* es la mayor de toda posible t-norma, eso quiere decir que cualquier otra t-norma tendría este problema. Lo mismo se puede aplicar a las t-conormas para las disyunciones.

Como se comentó en el capítulo 3, el enfoque presentado en esta tesis debe entenderse como una extensión de otras propuestas de carácter general. Su implementación se ha evaluado integrado junto con OWLS-MX, uno de los matchmakers más conocidos en la actualidad, y los experimentos realizados han permitido comprobar que la combinación de ambos usando ROWLS como pre-filtro mejora tanto la eficiencia como la efectividad de OWLS-MX. Las pruebas de rendimiento además han reflejado la importancia de minimizar al máximo el número de cargas de los servicios, siendo recomendable, si la arquitectura del sistema lo permite, integrar la equiparación con el directorio de forma que la de servicio se realice al registrarlo (menos frecuente).

Capítulo 5

Filtros genéricos para composición de servicios

En un sistema con una arquitectura orientada a servicios, cuando un agente busca un servicio típicamente lo hace utilizando algún servicio de directorio. Si no se encuentra ningún proveedor del servicio buscado, entonces se suele poder solicitar a un agente de planificación que intente crear un servicio compuesto que incluye varios servicios existentes. Para poder generar un plan que cumpla la consulta original, el planificador necesita como entrada un conjunto de servicios a partir de los cuales intentar componer el plan.

Idealmente, el conjunto de servicios tenidos en cuenta para crear el plan compuesto debería contener todos los servicios registrados en el directorio. Sin embargo, esto puede ser impracticable cuando el número de servicios es alto, como se espera que ocurra en entornos abiertos como Internet. Para superar este problema, es necesario reducir el conjunto de servicios de entrada que se pasan al planificador. Para este propósito, se proponen filtros que eliminen aquellos servicios que sean menos relevantes para el proceso de planificación.

La selección del conjunto de servicios candidatos a formar el plan no es una tarea sencilla. Se pueden pensar varias heurísticas ad-hoc (p. ej. servicios que

comparten al menos una entrada o salida con la consulta, etc.). En este trabajo se propone un método más informado para el filtrado de servicios que hace uso de *información sobre la clases de los servicios*. En primer lugar se desarrolla un marco genérico para el filtrado de servicios basado en clases, y luego se propone instanciarlo para diferentes filtros en base a (a) información organizacional obtenida de la ontología de roles y (b) la categoría del servicio.

5.1. Visión general

En esta sección se da una visión general al enfoque propuesto para el filtrado de servicios para composición. Partiendo de una situación ideal, se resumen las principales dificultades que se necesita superar, para motivar nuestro mecanismo de filtrado de servicios. Los detalles técnicos del proceso se describen en las secciones siguientes.

A un nivel alto de abstracción, el problema de planificación de un servicio compuesto puede concebirse como sigue. Sea $P = \{p_1, p_2, ..., p_m\}$ el conjunto de todos los posibles planes (servicios compuestos) para una determinada consulta Q, y $D = \{s_1, s_2, ..., s_n\}$ el conjunto de los servicios de entrada al planificador (es decir, todos los servicios disponibles en el directorio). El objetivo de un filtro F es seleccionar un número dado l de servicios de D, tal que el espacio de búsqueda se reduce, pero el mejor plan de P todavía puede ser encontrado.

En una *situación ideal* con información completa (i) el conjunto de todos los planes P es conocido y (ii) la *calidad* de cada plan puede ser evaluada (obviamente, el plan que requiere el menor número de servicios no es siempre el mejor plan para una consulta). En este caso, el conjunto de servicios devueltos por el filtro debería incluir todos los servicios del mejor plan (así como otros servicios hasta alcanzar la cantidad l). Sin embargo, es obvio que este caso ideal no es realista ya que el problema ya estaría resuelto (es decir, se conoce de antemano el plan de máxima calidad).

En un siguiente paso, se supone que no es posible para el filtro evaluar la

calidad de los planes de P. En este caso, si el número de servicios necesarios para la ejecución de todos los planes en P es mayor que el número de servicios l que permite pasar el filtro, éste debería estar seguro que la poda del espacio de búsqueda del planificador es mínimo. Dicho de otra manera: cuanto más grande sea el subconjunto de planes $P' \subseteq P$ entre los que el planificador puede elegir, mayor es la probabilidad de que el plan de máxima calidad esté entre ellos. Por tanto, el filtro debería seleccionar aquellos servicios que maximicen la cardinalidad de P', es decir, que maximicen el número de planes de P que estén disponibles para el planificador. Una buena heurística a este respecto está basada en la *dimensión del plan* y en el *número de ocurrencias* de servicios en planes: se supone que un servicio es más importante cuanto mayor es el número de planes de P en los que es necesario, y cuanto más cortos son los planes de P en los que es requerido.

De nuevo, no es realista asumir información completa respecto al conjunto de planes P para una consulta Q en general, y respecto a su longitud y número de ocurrencias de servicios en ellos en particular. No obstante, podemos aproximar esta información almacenando y procesando los planes históricamente creados. Así, en principio, podemos construir matrices como la mostrada en la tabla 5.1 para cada posible consulta. En el ejemplo, para una consulta Q, el servicio s_2 fue parte de 10 planes compuestos de dos servicios y 55 planes formados por 3 servicios. En el ejemplo, la matriz también almacena el número de planes generados de cada dimensión.

Información histórica sobre planes para la consulta de servicio Q				
Dimensión	1	2	3	...
nº de planes	0	50	70	
s_1	0	7	24	
s_2	0	10	55	
s_3	0	13	10	
s_4	0	17	11	
...				

Tabla 5.1: Ejemplo de información histórica sobre planes

Sin embargo, pronto queda evidente que el número de servicios y posibles consultas es demasiado grande para mantener todas las matrices del tipo de las anteriores. Los requerimientos de memoria serían prohibitivamente altos y el proceso de filtrado sería computacionalmente demasiado costoso. Además, la continua repetición de exactamente la misma consulta de servicio Q is bastante improbable e, incluso más importante, este enfoque no sería adecuado cuando una nueva solicitud de servicio (no planificada anteriormente) es requerida (lo cual, por cierto, es bastante habitual).

Para superar este inconveniente, asumimos la disponibilidad de *información sobre la clase del servicio*, para agrupar servicios basándose en ciertas propiedades (en particular, se usarán las *categorías* de los servicios, y la taxonomía de *roles* que refleja la estructura organizacional subyacente a las interacciones entre los agentes, presentada en este trabajo). Por tanto, solamente habrá matrices por cada *clase* de servicio deseado (consulta), y la información almacenada en dichas matrices se referirá a *clases* de servicios en lugar de servicios concretos. Si el número de clases no es demasiado grande, entonces el enfoque propuesto puede ser computacionalmente factible.

Las clases de servicios no tienen porqué ser necesariamente distintas. Esto permite, por ejemplo, describir que un servicio de ambulancia puede pertenecer tanto a la categoría de transporte como tratamiento médico. En particular, para los anuncios de servicios podemos especificar que un servicio pertenece simultáneamente a varias clases distintas (representado mediante un conjunto), y para las consultas se permite una fórmula en forma normal disyuntiva.

Definición 5 *Un filtro para un conjunto de servicios S es una terna $F_S =< C, A, R >$, donde:*

- C es un conjunto de *clases de servicio*.
- $A : S \rightarrow 2^C$, es una función que devuelve el conjunto de clases incluidas en la descripción del anuncio de servicio S.
- $R : S \rightarrow \theta$, siendo $\theta = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} c_{ij}, c_{ij} \in A(S)$, es una función que de-

vuelve la expresión (en FND) formada por las clases de servicio incluidas en la solicitud de servicio S.

La figura 5.1 muestra la estructura de nuestro enfoque para el filtro de servicios para composición. Con cada resultado de una solicitud de composición de servicio, se actualiza una *Matriz de Información Histórica H* (una abstracción de la tabla 5.1). Con esta información, se revisa y refina el contenido de una *Matriz de Relevancia v*. Basándose en esta matriz, la relevancia del servicio se determina de forma bastante sencilla. Para cada solicitud de composición de servicio, el método de filtrado se basa en esta función de relevancia.

En las siguientes secciones se describen los detalles de los diferentes aspectos del enfoque presentado hasta ahora (mostrado en la figura 5.1). Primero, en la sección 5.2 se muestra cómo se obtiene la matriz de relevancia. A continuación (sección 5.3), se describe el cálculo de la relevancia de un servicio (usando la matriz de relevancia). En la sección 5.4, se presentan diferentes opciones para la instrumentación de filtros para composición. A continuación (sección 5.5), se presentan dos filtros concretos, uno basado en los *roles* jugados por los proveedores de servicios en las interacciones en las que puede participar y otro basado en las *categorías* de los servicios. En la sección 5.6 se describen algunos aspectos sobre la implementación de la propuesta. Posteriormente, en la sección 5.7, se realizan una serie de experimentos para evaluar este enfoque. Se termina el capítulo con una discusión sobre los resultados y otros trabajos relacionados.

5.2. Obtención de la matriz de relevancia

En esta sección se describe cómo se puede obtener la matriz de relevancia $v(s, q)$ a partir de planes pasados. Primero se describe cómo se almacena la información sobre los planes y cómo se calcula la matriz de relevancia a partir de esta información. A continuación, se propone un método para refinar la matriz de relevancia. Finalmente, se describen varias opciones para el arranque inicial del filtro.

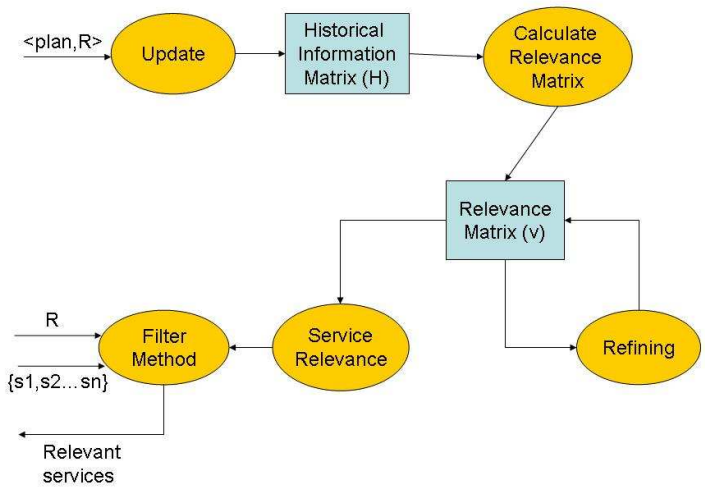


Figura 5.1: Arquitectura del filtro

5.2.1. Información histórica sobre planes

La *matriz de relevancia v* representa la estimación de que una *clase de servicios s* esté incluida en un (el mejor) plan compuesto para proveer una *clase de servicio q*. Para crear la matriz de relevancia partimos de un conjunto de planes (servicios compuestos) que fueron creados en el pasado. Tratamos con versiones simplificadas de esos planes. En particular, únicamente nos interesan las *clases de servicios* que componen cada plan. Más precisamente, para cada plan utilizamos la siguiente información:

- Información sobre clases que se incluyeron en la solicitud a partir de la cual se produjo el plan. Puesto que estamos proponiendo un marco genérico, en el que a partir de una descripción de servicio en cierto lenguaje (por ejemplo, OWL-S) se obtiene información sobre la(s) clase(s) a la que pertenece el servicio, no podemos determinar a priori cómo vendrá expresado ese mapping de servicio a clases, que dependerá de cómo se instancie este marco para un enfoque particular. Por ejemplo,

si se decide que en cada consulta de servicio se especifica una y sólo una *clase* (por ejemplo, una categoría de una taxonomía), entonces sería suficiente considerar una clase. Pero si se admiten varias clases para especificar un mismo servicio, ahora habría que representar esa información mediante alguna estructura compuesta (por ejemplo un conjunto), aunque habría que analizar la correcta interpretación semántica de esa estructura (p. ej. si debe ser un *y* o un *o*). En definitiva, con el fin de ser lo más flexible posible, pero dentro de una complejidad razonable, se opta aquí por permitir expresiones lógicas en forma normal disyuntiva. Se utiliza la notación C_{Q1}, C_{Q2}, ..., C_{Qn} para las clases incluidas en dicha expresión.

- Dimensión y clases de servicios en el plan. La dimensión es el número de servicios que forman el plan (que en general no tiene porqué coincidir con el número de clases). El conjunto de clases $PC = \{C_{P1}, C_{P2}, ..., C_{Pm}\}$ es el obtenido al extraer las clases de los servicios incluidos en el plan compuesto.

Definición 6 *Desde el punto de vista del filtro la información recogida en un plan es* $IP =< Q, D, PC >$*, siendo:*

- Q la consulta para la que se construyó el plan. Viene dada como una expresión en FND de clases $Q = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} c_{ij}, c_{ij} \in C$), siendo C el conjunto de todas las clases de servicio.
- $D \in \mathbb{N}$ es la dimensión (número de servicios) del plan.
- $PC \in C$ es el conjunto de clases de servicios que forman parte del plan.

La información sobre los planes creados en el pasado se almacena en una matriz como la que se muestra en la tabla 5.2 (adaptada a partir de la tabla 5.1 considerando clases en lugar de servicios y donde, por ejemplo, los servicios s_3 y s_4 han sido agrupados en la clase C_3). Matrices H^q de este tipo existen para cada clase de servicio q.

H^q: Información histórica sobre planes para la clase de servicio q (consulta)				
Dimensión	1	2	3	...
nº de planes	0	50	70	
C_1	0	7	24	
C_2	0	10	55	
C_3	0	30	21	
...				

Tabla 5.2: Ejemplo de información histórica de sobre clases de servicios en planes

La información de esta tabla se computa de la siguiente manera. Si la expresión de la solicitud sólo incluye una clase de servicio, el valor del "número de planes" y de cada clase del PC se incrementa en uno, para esa dimensión del plan[1].

Sin embargo, cuando la solicitud incluye más de una clase, las cosas no son tan sencillas. El problema es que, cuando hay más de una clase en la solicitud, no es sencillo determinar para qué partes de la solicitud es relevante una clase servicio del plan compuesto.

En el caso de una conjunción (p. ej. $C_{Q1} \wedge C_{Q2}$), asumimos que todas las clases incluidas en el plan son relevantes tanto para C_{Q1} como para C_{Q2}, ya que ambas condiciones deben cumplirse. Sin embargo,en el caso de una disyunción este no es el caso. Por ejemplo, si la solicitud es $(C_{Q1} \vee C_{Q2})$ y el plan incluye las clases $\{C_{P3}, C_{P4}, C_{P5}\}$, no esta claro si estas tres clases participan en proveer tanto C_{Q1} como C_{Q2} o si, por ejemplo, C_{P3} y C_{P4} son relevantes para C_{Q1}, y C_{P5} sólo para C_{Q2}. En este último caso disminuimos el posible impacto negativo de añadir una unidad a la clase equivocada ponderando la contribución de la información sobre el plan por la inversa del número de términos en la expresión disyunción.

[1]Puesto que todos los servicios tienen que estar disponibles para que ese plan sea ejecutable, cada clase de servicio tienen la misma relevancia para el plan, independientemente de si uno o varios servicios de esa clase se usan en el plan

Por ejemplo, supóngase una (expresión) consulta de la siguiente forma: $(C_{Q1} \vee (C_{Q2} \wedge C_{Q3}) \vee C_{Q4})$. Además, asúmase que las clases de servicios requeridos por el plan (servicio compuesto) que se ha determinado que es el que mejor encaja con la consulta son $PC = \{C_{P1}, C_{P2}\}$. Además, supóngase que la dimensión de dicho plan es 4, y las matrices históricas correspondientes a las cuatro clases en la expresión de la solicitud (C_{Q1}, C_{Q2}, C_{Q3}, C_{Q4}) se actualizan de acuerdo con esto. En particular, los valores de $H^{C_{Q1}}$("nº de planes",4), $H^{C_{Q1}}(C_{P1}, 4)$, $H^{C_{Q1}}(C_{P2}, 4)$, $H^{C_{Q2}}$("nº de planes",4), $H^{C_{Q2}}(C_{P1}, 4)$, $H^{C_{Q2}}(C_{P2}, 4)$, y lo mismo para $H^{C_{Q3}}$ y $H^{C_{Q4}}$, se incrementan en $\frac{1}{3}$ puesto que el número de términos de en la disyunción es 3.

La figura 5.2 muestra el algoritmo de actualización de la matriz histórica con la información de un nuevo plan generado. Recibe la expresión de clases de la consulta, las clases de servicios que componen el plan, así como la dimensión en número de servicios del plan.

```
<PC> - SET OF Class          // plan classes
<RE> - <DisjunctionExpr>     // request expression
<DisjunctionExpr> - SET OF <ConjunctionExpr>
<ConjunctionExpr> - SET OF Class

UpdateHistorical(H: historical matrix; RE: request expression;
                 PC: plan classes; dim: natural)
{
  inc - 1/Card(RE)
  FOR ALL Conj IN RE {
    FOR ALL C IN Conj {
      H^c("# of plans",dim) - H^c("# of plans",dim) + inc
      FOR ALL pc IN PC {
        H^c(pc,dim) - H^c(pc,dim) + inc
      }
    }
  }
  return H
}
```

Figura 5.2: Algoritmo de actualización de la matriz con la información histórica

5.2.2. Cálculo de la matriz de relevancia

Como se comentó anteriormente, mediante el ranking de servicios se pretende seleccionar un conjunto de ellos que cubran el subconjunto del espacio de

planes más amplio posible, en un intento de maximizar la posibilidad de que el mejor plan esté contenido en él. Los servicios que formaron planes más pequeños en el pasado se consideran más relevantes, puesto que es más fácil cubrir planes pequeños que grandes, así pues con menos servicios se pueden cubrir más planes.

Definición 7 *Sea C una clase de servicio y q una clase incluida en una solicitud de servicio, la relevancia de C para q ($Relevancia(C, q)$) se obtiene a partir de la información sobre planes como:*

$$Relevancia(C, q) = \frac{\sum_{d=1}^{m} \frac{n_d}{d^\alpha}}{\sum_{d=1}^{m} \frac{N_d}{d^\alpha}} \tag{5.1}$$

Donde d es la dimensión del plan (número de servicios), m es la dimensión del plan más grande almacenado, n_d es el número de veces que C fue parte de un plan compuesto de dimensión d para la consulta q, y N_d es el número total de planes de dimensión d ("nº de planes") para esa consulta. α es una constante ≥ 0 que permite dar más importancia a planes de dimensión más pequeña (un valor inmediato es $\alpha = 1$, un valor de $\alpha = 0$ desactiva la heurística basada en dimensión de los planes). Sólo se consideran planes de dimensiones mayores que 0.

En el ejemplo de la tabla 5.2,

$$Relevancia(C_1, q) = \frac{7/2 + 24/3}{50/2 + 70/3} = 0.24$$

Con este cálculo se obtiene una relevancia entre 0 y 1 para cada clase de servicio C con respecto a la composición de una clase de servicio q.

La tabla 5.3 muestra un ejemplo parcial de tabla de relevancia. En dicha tabla $v(i, j)$ es la relevancia estimada de una clase de servicio C_i para una clase de servicio C_j de la solicitud (query) .

v		Solicitudes							
		C_1	C_2	...	C_i	...	C_j	...	C_n
Servicios en directorio	C_1		$v(1,2)$		$v(1,i)$		$v(1,j)$		$v(1,n)$
	C_2								
	...								
	C_i				$v(i,i)$		$v(i,j)$		
	...								
	C_j								
	...								
	C_n								

Tabla 5.3: Matriz de Relevancia

Obsérvese que cada celda de esta tabla se obtiene a partir de una fila de la tabla 5.2 y que cada columna de la tabla 5.3 se obtiene de n matrices de información histórica. Por tanto la complejidad computacional para el cálculo de la matriz de relevancia es $O(m \cdot n^2)$, siendo m la dimensión del plan almacenado de mayor longitud y n el número de clases (recuérdese que el número de clases n se supone fijo y no demasiado grande).

5.2.3. Refinamiento de la matriz de relevancia

La matriz $v(s, q)$ especifica la relevancia de una clase de servicio s para ser parte de un plan (servicio compuesto) que encaja con la consulta sobre una cierta clase de servicio q. Sin embargo puede ocurrir una situación como la siguiente. Supongamos que se busca un plan que consiga C_1 y que una solución potencial es componer los servicio C_2 y C_3 (lo notaremos con $C_2 \oplus C_3$). Sin embargo no hay ningún proveedor del servicio C_3, pero en su lugar éste puede componerse como $C_4 \oplus C_5 \oplus C_6$, por lo que el plan final es $C_2 \oplus C_4 \oplus C_5 \oplus C_6$. Desafortunadamente, el valor $v(C_4, C_1)$ es bajo (por ejemplo, la mayoría de las veces que se planificó sí estaba disponible C_3 o no lo estaba C_4) y el servicio que proporciona C_4 es descartado y no tenido en cuenta en el proceso de planificación, por lo que el mencionado plan no puede ser encontrado por el

planificador.

Para solventar este problema podríamos realizar la siguiente actualización: $v(C_4, C_1) = v(C_4, C_3) \cdot v(C_3, C_1)$. Y siguiendo el mismo razonamiento lo mismo se mantiene para dependencias de tercer nivel (p. ej.: $v(C_7, C_1) = v(C_7, C_4) \cdot v(C_4, C_3) \cdot v(C_3, C_1)$).

Este ejemplo motiva la siguiente definición.

Definición 8 *Sea $v(s, q)$ una matriz de relevancia, su matriz refinada en k pasos, $v^k(s, q)$, se calcula como:*

$v^1(s, q) = v(s, q)$

$v^k(s, q) = Max(v^{k-1}(s, q), v^{k-1}(s, s_1) \cdot v^{k-1}(s_1, q), v^{k-1}(s, s_2) \cdot v^{k-1}(s_2, q), ..., v^{k-1}(s, s_n) \cdot v^{k-1}(s_n, q))$

Como se muestra en la fórmula, usamos el producto como función de combinación y el máximo para agregar los resultados (como siempre, otras funciones de agregación más complejas podrían ser posibles). Puesto que el producto de dos números entre 0 y 1 está siempre en ese mismo rango, los valores de relevancia tras el refinamiento siguen perteneciendo al rango [0,1].

La tabla 5.4 muestra una matriz de relevancia para este ejemplo. En este caso,

$v^2(C_4, C_1) = Max(v(C_4, C_1), v(C_4, C_1) \cdot v(C_1, C_1), v(C_4, C_2) \cdot v(C_2, C_1), v(C_4, C_3) \cdot v(C_3, C_1), v(C_4, C_4) \cdot v(C_4, C_1), v(C_4, C_5) \cdot v(C_5, C_1), v(C_4, C_6) \cdot v(C_6, C_1))$

$v^2(C_4, C_1) = Max(0.1, 0.1 \cdot 0.9, 0.7 \cdot 0.8, 0.8 \cdot 0.7, 0.9 \cdot 0.1, 0 \cdot 0.5, 0.2 \cdot 0.6) = 0.56$

Obsérvese que cuanto mayor sea el valor de k mejor es la estimación de la relevancia de las *clases de servicio*. El refinamiento de la matriz de relevancia se repite hasta que converge (es decir $p^{k+1}(s, q) = p^k(s, q)$) o durante un tiempo determinado. La elevada complejidad en tiempo de cómputo ($O(n^3)$) para cada paso de refinamiento se atenúa por las *propiedades anytime*[2] del algoritmo. Además, recuérdese que el número de clases n se supone fijo y no demasiado

[2]su valor puede usarse sin terminar de refinar

v		Solicitudes					
		C_1	C_2	C_3	C_4	C_5	C_6
Servicios en directorio	C_1	0.9	0.3	0.1	0.6	0.4	0
	C_2	0.8	1	0.1	0.2	0.1	0.1
	C_3	0.7	0	1	0.8	0	0.6
	C_4	0.1	0.7	0.8	0.9	0	0.2
	C_5	0.5	0.6	0.2	0.3	1	0.8
	C_6	0.6	0	0.8	0.1	0.7	1

Tabla 5.4: Ejemplo de Matriz de Relevancia antes de refinamiento

grande. Finalmente, obsérvese que se pueden combinar varias actualizaciones y refinamientos en modo *batch* para ser ejecutados juntos cuando la carga del sistema sea baja. El algoritmo de refinamiento se muestra en la figura 5.3.

La tabla 5.5 muestra el resultado del primer paso de refinamiento (v^2) para la matriz de relevancia de la tabla 5.4. Para este ejemplo, en un segundo paso (v^3) ya se alcanza el resultado final ($v^3 = v^4$).

v		Solicitudes					
		C_1	C_2	C_3	C_4	C_5	C_6
Servicios en directorio	C_1	0.90	0.42	0.48	0.60	0.40	0.32
	C_2	0.80	1.00	0.16	0.48	0.32	0.10
	C_3	0.70	0.56	1.00	0.80	0.42	0.60
	C_4	0.56	0.70	0.80	0.90	0.14	0.48
	C_5	0.50	0.60	0.64	0.30	1.00	0.80
	C_6	0.60	0.42	0.80	0.64	0.70	1.00

Tabla 5.5: Matriz de Relevancia tras un paso de refinamiento (v^2)

Debido a la dificultad de realizar un análisis matemático de la función de refino se ha realizado un estudio mediante simulación. El objetivo es observar la velocidad de convergencia de la matriz de relevancia y cómo afecta cada paso de refinamiento. Se han utilizado matrices de relevancia de partida generadas aleatoriamente siguiendo varias distribuciones distintas. Los resultados

```
RefineMatrix(v: relevance matrix)
{
  REPEAT
    v = v_next
    v_next = RefineStep(v)
  UNTIL v = v_next
}
RefineStep(v: relevance matrix)
  v': relevance matrix
{
  FOR ALL s IN C1..Cn {
    FOR ALL q IN C1..Cn {
      v'(s,q) = v(s,q)
      FOR ALL i IN C1..Cn {
        v'(s,q) = max(v'(s,q),v(s,i)*v(i,q))
      }
    }
  }
  return v'
}
```

Figura 5.3: Algoritmo de refinamiento de la matriz de relevancia

obtenidos son similares.

La figura 5.4 muestra el número de pasos de refino medio obtenido hasta conseguir la convergencia de la matriz de relevancia. Se muestra el resultado para dos métodos de generación de la matriz original: una distribución uniforme ($U(0,1)$) y una distribución normal ($N(\mu = 0.5, \sigma = 0.5)$), con valor 0 en caso de valores fuera del rango [0,1]. Como puede observarse en la figura el resultado es similar y sigue una función logarítmica, llegando a un valor medio de menos de 7 pasos para lograr la convergencia con 60 *clases*. Este dato es importante ya que este número es pequeño haciendo manejable el proceso de cómputo de dicho refinamiento.

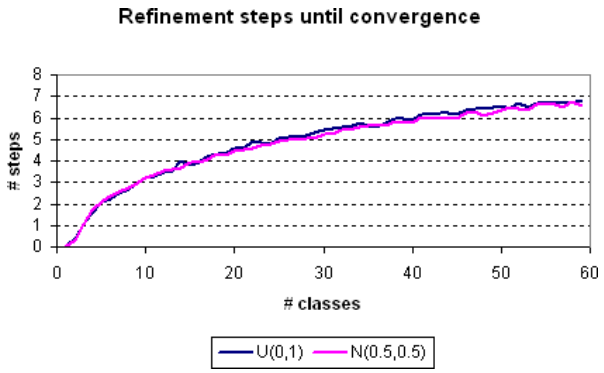


Figura 5.4: Número de pasos de refino necesarios hasta que la matriz de relevancia converge

Además se ha analizado lo que aporta cada paso de refinamiento. Examinando los datos observó que en el primer paso de refinamiento la relevancia media se incrementa aproximadamente un 85 % del total conseguido en el proceso, en el segundo un 15 %, en el tercero un 1 % y a partir de ahí el incremento es mínimo. De hecho, aunque en la gráfica se ve que llega hasta 6 o 7 pasos con muchas clases, en realidad, analizando los cambios en las matrices a partir del cuarto paso el cambio es prácticamente 0. Con este resultado podríamos mejorar la eficiencia del proceso realizando sólo hasta 3 pasos de refinamiento o cuando la diferencia entre una matriz y la siguiente sea pequeña.

La aportación del proceso de refino se analizará en la sección de evaluación

(sección 5.4) en base a los experimentos realizados.

5.2.4. Arranque inicial

Puesto que el método de filtrado propuesto se basa en información sobre planes creados en el pasado, se plantea el problema de qué hacer cuando se usa por primera vez. Hay varias formas de obtener la matriz de relevancia inicial:

a) Si se dispone de registros históricos de planes se pueden usar para calcular la matriz

b) Se puede asignar una distribución a priori usando conocimiento experto (heurístico)

c) El planificador puede trabajar durante un cierto tiempo sin filtrar servicios hasta que el número de planes generados es considerado suficientemente representativo. Es entonces cuando se calcula la matriz de relevancia inicial y se refina.

Estas opciones también se pueden combinar. Por ejemplo, la distribución heurística a priori se puede combinar con una base de datos histórica como matriz de inicio. Además, si esa matriz se supone que no está suficientemente formada, entonces puede ser completada con la opción c. Una distribución a priori basada en conocimiento experto puede incorporarse en nuestro modelo directamente como la matriz de relevancia pero preferiblemente como una matriz histórica (ficticia). Ésto tiene la ventaja de que puede combinarse con datos históricos o/y otros conocimientos expertos (podría haber varias fuentes de conocimiento experto) añadiendo todas las matrices históricas, seguido del cálculo de la matriz de relevancia tal como se explicó anteriormente. Si la distribución a priori fuera incluida directamente como matriz de relevancia no podría combinarse con información histórica puesto que la matriz de relevancia se obtiene de la matriz histórica y, en consecuencia, la información a priori se perdería.

5.3. Cálculo de la relevancia de un servicio

En esta sección se describe cómo se calcula la relevancia de un servicio S para una solicitud de servicio Q, usando la matriz de relevancia v.

El primer paso para calcular la *relevancia* de un servicio S para una consulta Q es la asociación de ambos a *clases de servicios*, para después calcular la relevancia entre clases. Se utilizará la siguiente notación:

$v(s, q)$: relevancia de la *clase* s para la *clase* q de la consulta.

$V(S, Q)$: relevancia del servicio S para la consulta Q

Dependiendo de si los servicios se asocian a una o varias clases se aplica lo siguiente.

1. El caso más simple es que una consulta Q sólo incluya una clase (q) en su descripción. Se pueden dar dos casos:

   a) El servicio S sólo pertenece a una clase (s). En este caso $V(S, Q) = v(s, q)$

   b) El servicio S pertenece a varias clases ($s_1, s_2, ..., s_n$). En este caso se toma la mayor relevancia de las diferentes clases, es decir,

   $$V(S, Q) = \text{máx}(v(s_1, q), v(s_2, q), ..., v(s_n, q))$$

   De nuevo, aunque son concebibles funciones más sofisticadas, usamos el máximo para la agregación ya que es fácil de computar e intuitivo para los humanos.

2. La consulta especifica una expresión lógica que contiene varias clases de servicios ($q_1, q_2, ..., q_m$). Ahora, de nuevo tenemos dos casos:

   a) El servicio S sólo pertenece a una clase (s). Se evalúan las fórmulas lógicas usando el *máximo* para disyunciones y el *mínimo* para conjunciones. Por ejemplo, si la solicitud Q incluye la fórmula

$q_1 \vee (q_2 \wedge q_3)$, entonces

$$V(S,Q) = \text{máx}(v(s,q_1), \text{mín}(v(s,q_2), v(s,q_3)))$$

b) El servicio S pertenece a varias clases $(s_1, s_2, ..., s_n)$. En este caso se combinan las dos opciones previas: la fórmula correspondiente a la solicitud se evalúa descomponiéndola como 2a) y usando dentro el *máximo* para agregar las clases de servicios especificados por el proveedor. Por ejemplo, si en el último caso el servicio S perteneciera a las clases s_1 y s_2 el cálculo sería:

$$V(S,Q) = \text{máx}[\text{máx}(v(s_1,q_1), v(s_2,q_1)), \\ \text{mín}(\text{máx}(v(s_1,q_2), v(s_2,q_2)), \text{máx}(v(s_1,q_3), v(s_2,q_3)))]$$

La figura 5.5 muestra el algoritmo para el cálculo de la relevancia de un servicio S para una solicitud Q. Esto se realiza mediante la función *ServiceRelevance*. Esta función, a su vez, utiliza la función *SingleRelevance*, que devuelve la relevancia del servicio anunciado S para una única clase q de la solicitud. Como se describió anteriormente, la solicitud puede incluir no sólo una *clase de servicio* sino también una expresión (FND). Los dos bucles descomponen esa expresión, usando el *mínimo* como función de combinación para los valores en una conjunción y el *máximo* para disyunciones. Asumiendo que el máximo número clases al que un servicio puede pertenecer es insignificante, la complejidad en tiempo del algoritmo es lineal en el número de literales de la consulta.

5.4. Tipos de filtros genéricos para composición de servicios

Cuando el filtro presentado analiza una solicitud de servicio, el conjunto de servicios primeramente se ordena según la estimación de la relevancia de la(s) clase(s) del servicio para esa solicitud. Después, sólo se pasan al planificador los servicios con mayor relevancia.

Para determinar los servicios concretos que pasan el filtro se consideran tres opciones principales:

```
ServiceRelevance(S: service advertisement; Q:service request;
                 v: relevance matrix)
{
  rel = 0
  FOR ALL ConjunctionExpr IN Q {
    rel' = inf
    FOR ALL q IN ConjunctionExpr {
      rel = min(rel',SingleRelevance(q,S,v))
    }
    rel = max(rel, rel')
  }
  return rel
}
SingleRelevance(q: request class; S: service advertisement;
                v: relevance matrix)
{
  rel = 0
  FOR ALL s IN S {
      rel = max(rel,v(s,q))
  }
  return rel
}
```

Figura 5.5: Algoritmo de cálculo de la relevancia de un servicio

1. Establecer un *umbral* y filtrar aquellos servicios que tienen un grado de relevancia por debajo del umbral.

2. Devolver los *mejores k* servicios estimados basándose en la relevancia de las clases a las que pertenece. En este caso el número de servicios que pasan el filtro es predeterminado, lo que que puede permitir estimar de forma más precisa el tiempo empleado en planificar.

3. Devolver un *porcentaje* del conjunto original de servicios (basándose en la relevancia de sus clases). En este caso el número de servicios considerados en el proceso de planificación depende del tamaño del directorio.

En el diseño de los algoritmos correspondientes a estas configuraciones del filtro, hay que tener en cuenta un problema adicional. Los servicios con valores bajos (o incluso cero) de relevancia podrían no ser considerados nunca para la planificación, por lo que nunca formarían parte de un plan (servicio compuesto), permaneciendo para siempre con poca relevancia. Esto es obviamente muy restrictivo, ya que nuestros valores de relevancia son solamente estimaciones basados en información disponible en algún instante de tiempo. Para superar esto permitimos que algunos servicios también sean pasados al planificador incluso aunque se estime que no son suficientemente relevantes de acuerdo a la política del filtro. Esos servicios adicionales se eligen aleatoriamente. Esta opción aleatoria se combina con los tres tipos de filtros comentados anteriormente para permitir la exploración del espacio de (clases de) servicios.

La figura 5.6 muestra los tres algoritmos para el filtro basado en *clases* de servicios, dependiendo del tipo de filtro deseado. Las tres funciones reciben un conjunto de descripciones de servicios candidatos, una solicitud de servicio, y la matriz de relevancia.

La función *ThresholdFilter* devuelve todos los servicios cuya relevancia supera un umbral dado (recibido como parámetro). Su resultado también incluye servicios adicionales que son elegidos aleatoriamente con una probabilidad dada como parámetro.

La función *K-Filter* devuelve los k estimados mejores servicios. Además, un

```
ThresholdFilter(S: set of services; Q: service request; v: relevance matrix,
                THRESHOLD: [0..1]; RANDOM_PROBABILITY: [0..1])
{
  result = ∅
  FOR ALL s IN S {
    IF (ServiceRelevance(s,Q,v) > THRESHOLD) OR
       (random(0..1) < RANDOM_PROBABILITY) THEN
      result = result ∪ s
  }
  return result
}

K-Filter(S: set of services; Q: service request; v: relevance matrix,
         K: Integer; K-RANDOM: Integer)
{
  relevances = ∅
  FOR ALL s IN S {
    SortInsert(<s, ServiceRelevance(s,Q,v)>,relevances)
  }
  result = relevances[1..K]
  i = 0
  WHILE i < K-RANDOM {
    x = random(K+1..Card(relevances))
    IF relevances[x] ∉ result THEN {
      result = result ∪ relevances[x]
      i := i+1
    }
  }
  return result
}

%-Filter(S: set of services; Q: service request; v: relevance matrix,
         PERCENTAGE: [1..100]; RANDOM_PERCENT: [0..100])
{
  n = Cardinal(S)
  return K_Filter(S,Q,v,n*PERCENTAGE/100,n*RANDOM_PERCENT/100)
}
```

Figura 5.6: Algoritmos para los tres tipos de filtros de servicios

número *K_RANDOM* de servicios son elegidos aleatoriamente entre el resto del directorio.

La función *%-Filter* devuelve un número predeterminado de servicios. En este caso el número no se especifica directamente sino como un porcentaje del tamaño del directorio. Igualmente un porcentaje especifica cuántos servicios deben incluirse aleatoriamente. Obsérvese que el algoritmo para esta función consiste en invocar el *K-Filter* donde los valores k se calculan a partir de la cardinalidad del conjunto de servicios de entrada y el porcentaje deseado.

Obsérvese que, si es necesario, la opción de incluir servicios de forma aleatoria se puede inhibir sencillamente pasando el valor 0 en los parámetros correspondientes.

Los distintos modos de operación y sus parámetros parámetros permiten adaptar el filtro dependiendo del contexto (dominio, recursos disponibles, tiempo de respuesta deseado, etc.). Por ejemplo, si los recursos del planificador son muy limitados se podría usar un filtro *K-Filter* con un valor pequeño de k o un filtro de *umbral* con un valor alto. Si se dispone de más recursos, entonces el umbral puede bajarse y/o se pueden incluir un número mayor de servicios de forma aleatoria.

5.5. Instanciación de filtros

En esta sección se presentan dos enfoques diferentes para aplicar el método de filtrado propuesto en este capítulo. Para cada uno de estos enfoques se define la asociación de servicios a clases, es decir se definen los componentes de $F =< C, A, R >$. Ambos métodos se basan en información disponible en las descripciones de los servicios disponible en OWL-S usados en el contexto de este trabajo.

5.5.1. Filtro basado en roles

Este método se basa en conceptos organizacionales como roles y tipos de interacciones sociales. La idea es relacionar los roles buscados en la consulta con los roles jugados por los agentes en el servicio compuesto, esto es, cuáles son los roles involucrados típicamente en un plan cuando la consulta incluye el rol r. Por ejemplo, es común que un servicio de *asistencia médica* incluya interacciones sobre *gestión de transporte*, *notificación de llegada*, *registro de entrada en el hospital*, *intercambio de información médica* y *consulta de una segunda opinión*.

La información sobre los roles jugados por los servicios, tanto en el caso de anuncios como de consultas, se describió en detalle en el capítulo 3. En concreto, las descripciones de servicios incluyen información relativa a los roles que el agente proveedor del servicio juega en las interacciones en las que puede participar. Para cada servicio se pueden anunciar varios tipos de interacciones. En el caso de las solicitudes de servicio, se permite especificar los roles buscados mediante una expresión en forma normal disyuntiva. Esta información encaja de forma directa en el marco genérico presentado en este capítulo simplemente considerando los roles como clases de servicios, tanto para anuncios como para solicitudes.

Definición 9 *Un filtro basado en roles es una terna* $F_S =< C, A, R >$*, donde:*

- C es el conjunto de roles de la ontología que se especificó en el capítulo 3 ($O =< R, I, \sqsubset, \Delta >$), es decir, en este caso $C = R$
- A es la función que dada la descripción de un servicio anunciado S devuelve el conjunto de roles que sabe jugar. Se recuerda que la descripción de un servicio realizada en la sección 3.4 (ecuación (3.2)) se definió como

$$S \subseteq \left\{ < r, \rho > | r \in R, \rho = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} r_{ij}, r_{ij} \in R \right\}$$

  entonces A devuelve el conjunto $A(S) = \{r| < r, \rho >\in S\}$

Esta información se extrae del parámetro *SERVICE_ROLES* de la descripción de servicio OWL-S.

- R es la función que dada una consulta de servicio Q devuelve la expresión de roles que define la consulta. De nuevo recordamos la descripción de consulta (ecuación (3.3)):

$$Q =< \rho, C >, \rho = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} r_{ij}, r_{ij} \in R, C \subseteq R$$

entonces R devuelve exactamente la expresión definida en la consulta Q, $R(Q) = \rho$.

Se recuerda que esta información se obtiene analizando el parámetro *QUERY_ROLES* de la descripción de servicio OWL-S.

En el enfoque de modelado basado en roles, se utiliza una taxonomía de roles que se supone estática durante gran cantidad de tiempo. Sin embargo, la ontología puede extenderse para incluir nuevos roles y tipos de interacciones no consideradas antes. En ese caso, la matriz de relevancia debe actualizarse con filas y columnas adicionales para esos nuevos roles. Los valores de relevancia para esos nuevos roles son desconocidos inicialmente (valor 0), por lo que sería difícil que fueran estimados relevantes por el filtro pero esto puede mitigarse incluyendo de forma aleatoria algunos servicios con relevancia baja (como se describió en la sección 5.4) y, en general, aplicando las técnicas de inicio descritas en la sección 5.2.4.

5.5.2. Filtro basado en categorías de servicios

Otra estrategia para la clasificación de servicios es basarse en las categorías (viajes, medicina, ...) a las que pertenecen. Las categorías se consideran información importante en las descripciones de servicios (de hecho, el lenguaje OWL-S incluye un campo específico para esta característica). Existen varias taxonomías de categorías establecidas (NAICS [82], UNSPSC [119], ...). En

nuestro caso, igual que ocurre en OWL-S, no nos restringimos a una taxonomía concreta.

En nuestro enfoque, cada categoría se considera una *clase de servicio*. Las descripciones de servicio (por ejemplo en OWL-S) incluyen un conjunto de categorías, ya que un servicio puede pertenecer a varias de ellas. En el caso de los anuncios de servicios, esto encaja exactamente en nuestro enfoque de *clases* (conjunto de clases). En el caso de las consultas de servicios, el conjunto de categorías especificado se interpreta como una fórmula lógica que conecta las categorías mediante el operador *o* ($\vee$).

Si el número de clases (categorías) diferentes es demasiado grande, la complejidad computacional (tanto espacio como tiempo) puede crecer demasiado. En ese caso, la granularidad de las clases puede disminuirse agrupando varias categorías en la misma clase basándose en relaciones de herencia en el árbol de la taxonomía. Según esto, sea C el conjunto de todas las categorías de la taxonomía (todos los conceptos de la ontología), se elegiría un subconjunto $C' \subseteq C$ de categorías. Si en una descripción de servicio aparece una categoría $c \notin C'$ entonces se toma como categoría representativa el *antecesor más específico (msa)* que sí pertenezca a C'. Lo podemos definir mediante una función $cat(c)$ que devuelve la categoría a considerar como *clase*.

Definición 10 *Sea c una categoría de servicio, $c \in C$, su categoría en C' es*

$$cat(c) = c' \text{ tal que } c' \in C' \wedge c \preceq c' \wedge \neg\exists c'' \in C', c \prec c'' \wedge c'' \prec c' \qquad (5.2)$$

Definición 11 *Un filtro basado en categorías es una terna $F_S =< C, A, R >$, donde:*

- C es el conjunto de categorías de la taxonomía que se consideren para representar las clases de servicio. Como se comentó pueden ser todas las de la taxonomía o un subconjunto si son muchas. Sea T una taxonomía de categorías y C_T el conjunto de todas las categorías en ella especificadas,

$$C = \{cat(c) | \forall c \in C_T\}$$

- A es la función que dada la descripción de un servicio S anunciado devuelve el conjunto de categorías. Puesto que en OWL-S se pueden especificar varias categorías, A será justamente el conjunto de todas esas categorías, más precisamente, el conjunto de todos los *cat* de las categorías que aparezcan en la descripción. Sea C_S el conjunto de categorías especificadas para el servicio S,

$$A(S) = \{cat(c) | \forall c \in C_S\}$$

  Esta información se extrae del atributo *serviceCategory* del *ServiceProfile* de la descripción de servicio OWL-S.

- R es la función que dada una consulta de servicio Q devuelve la expresión de de categorías que define la consulta. Puesto que en una descripción de servicio OWL-S se pueden especificar varias categorías, aquí se interpretan como una disyunción. Por tanto, sea C_Q el conjunto de categorías especificadas en la para el la solicitud de servicio Q,

$$R = \left\{ \bigvee cat(c) | \forall c \in C_Q \right\}$$

Diferentes instancias de filtros podrían combinarse para calcular la relevancia. Por ejemplo, una forma de combinar las dos anteriores propuestas de instancias presentadas anteriormente es la siguiente:

$Relevance(S, Q) = \alpha \cdot Rel^{RB}(S, Q) + (1 - \alpha) \cdot Rel^{CB}(S, Q)$, con $\alpha \in [0..1]$.

Siendo $Rel^{RB}(S, Q)$ y $Rel^{CB}(S, Q)$ el valor de la relevancia aplicando el filtro basado en roles y en categorías, respectivamente.

5.6. Implementación

En esta sección se describe brevemente la implementación software del método de filtrado propuesto en este capítulo, en concreto el basado en roles. La figura 5.7 muestra un diagrama de clases simplificado. La clase principal es la

clase abstracta *ServiceCompositionFilter*, que tiene tres métodos para aplicar el tipo filtro deseado a un conjunto de servicios y una consulta: *applyKFilter*, *applyThresholdFilter* y *applyPercentageFilter*. Para cada instancia concreta de filtro (roles, categorías, etc.) debe implementarse una clase que herede de ésta e implemente sus métodos abstractos. En nuestro caso el filtro basado en roles es implementado en la clase *ServiceRoleCompositionFilter*. Para leer la parte específica de los roles de los ficheros OWL-S se apoya en la clase *RolesReader*. Además, las clases *HistoricalMatrix* y *RelevanceMatrix* proporcionan el soporte a dichas estructuras de datos utilizadas en el algoritmo, proporcionando métodos para manipular esa información, como calcular la relevancia a partir de la matriz histórica, refinarla, actualizar con un nuevo plan, etc.

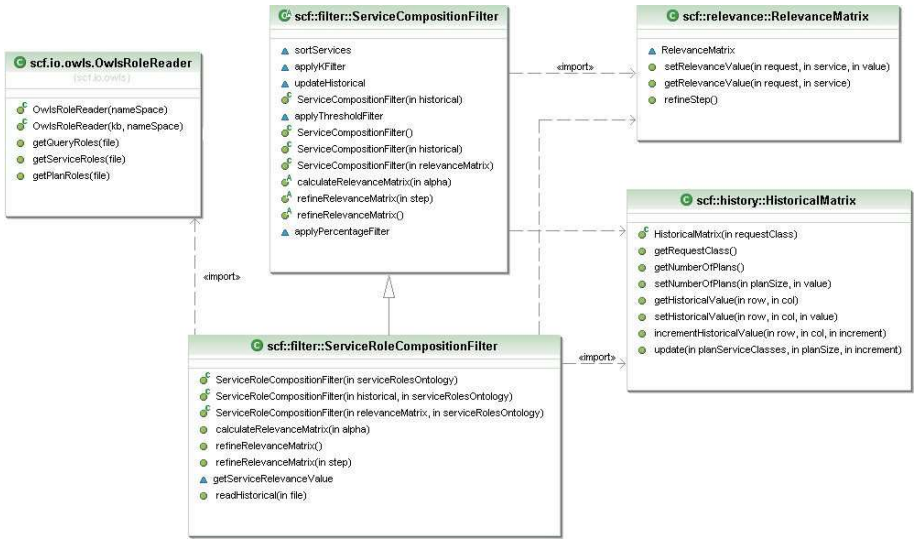


Figura 5.7: Diagrama de clases del filtro

La figura 5.8 muestra el contexto de utilización de un filtro para composición. La arquitectura tiene dos componentes, el filtro y un planificador. Los servicios del directorio pasan previamente por el filtro y sólo parte de ellos son utilizados por el planificador para intentar componer un plan para una consulta. Una vez compuesto el plan, se produce una realimentación hacia el filtro proporcionando la información sobre el plan con el fin de actualizar los datos históricos sobre planes con un nuevo caso y así actualizar la matriz de

relevancia.

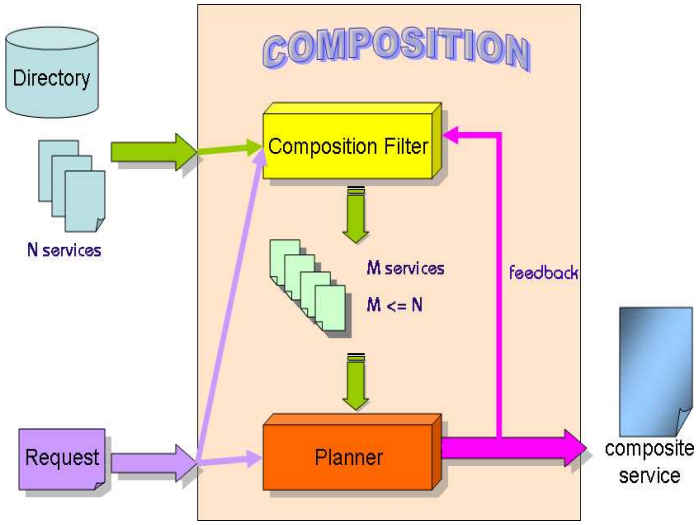


Figura 5.8: Arquitectura típica de integración del filtro con un planificador

5.7. Evaluación

En esta sección se describe la experimentación realizada para evaluar el filtro para composición. La evaluación se realiza a nivel genérico, sin centrarnos en instanciaciones concretas (roles, categorías, ...). Con esto conseguimos evaluar de forma separada en enfoque genérico de la decisión de cómo categorizar. En primer lugar se describe la preparación del banco de pruebas para realizar los tests. Posteriormente se analizan los resultados.

5.7.1. Preparación

A la hora de probar el funcionamiento de nuestro enfoque nos encontramos con un problema importante. Debido a la inmadurez de las tecnologías basadas en servicios web, en especial servicios Web semánticos, y más aún en las técnicas de composición de servicios, no existen en la actualidad buenas colecciones

de planes disponibles. Si ya es complicado en el caso de la evaluación de la equiparación de servicios, ahora el problema se agudiza pues no sólo es necesario de disponer de una colección de servicios, y pares <*consultas, servicios relevantes*>, sino que ahora son necesarios pares <*consulta, planes válidos*>, lo cuál es aún más difícil de conseguir.

Existen escasas colecciones de casos de prueba disponibles. Un caso es el *Web Services Challenge*[3]. Se trata de competición anual tiene una versión sintáctica y otra semántica, tanto para el descubrimiento como composición de servicios. Sin embargo la versión semántica actualmente (2007) sólo es cubierta con una versión simplificada de tipos representados en XML schema (y también en OWL) para definir los tipos de entrada y salida de los servicios, pero no cubre el resto de la descripción de un servicio, lo cuál no es suficiente para nuestras necesidades. Se encuentran disponibles los repositorios de datos de 2005 y 2006, los servicios vienen descritos en WSDL. Además las colecciones disponen de pocas consultas para planificación, aunque sí muchos servicios (generados de forma automática).

Otra iniciativa parecida, pero con otro enfoque distinto es la *Semantic Web Service Challenge*[4]. Ésta no es competición sino un workshop para cooperar. Por esto, definen unos pocos casos de prueba, muy especificados. El objetivo ahí es probar la ejecución e interoperabilidad de los servicios, no sólo el descubrimiento y planificación.

En la siguiente sección se describe el planteamiento seguido para la realización de las pruebas. Se ha tratado de evaluar las cuestiones más relevantes del enfoque propuesto y, en particular, la completitud y la proporción de planes filtrados.

[3] http://ws-challenge.org/
[4] sws-challenge.org

5.7.1.1. Metodología

En los experimentos realizados no se usa un planificador concreto, sino uno "ideal", que se implementa almacenando para cada consulta la lista de todos los planes que se pueden componer. Esto permite separar la evaluación del filtro de la eficacia de un planificador concreto, de forma que el planificador no influya en dicha evaluación.

El proceso seguido para realizar los experimentos es el siguiente. Sea $C = \{c_1, c_2, ..., c_n\}$ el conjunto de todas las *clases de servicios* posibles se genera el conjunto P_C de todos los planes que se pueden componer con esas clases. Si llamamos P_C^c al conjunto de planes para la consulta c, entonces $P_C = \bigcup_{i=1}^{n} P_C^{c_i}$. El planificador "ideal" tendrá para cada consulta c su conjunto de planes P_C^c.

Ahora, se selecciona un subconjunto $P_H \subseteq P_C$ que simulará la experiencia del planificador, planes que ha realizado en el pasado, y que formarán la *Matriz Histórica*, a partir de la que se obtiene la *Matriz de Relevancia*. Obsérvese que cuantos más planes se seleccionen mejor será la estimación que tendrá el filtro sobre la relevancia entre las clases de servicios.

Por otro lado, la situación en un caso real es que no todos los servicios (en este caso *clases*) están siempre disponibles en el directorio D, sino que se encuentran disponibles un subconjunto $C_D \subseteq C$. Ahora el conjunto de todos los planes posibles a partir de C_D es $P_D \subseteq P_C$. Este conjunto P_D está formado por todos aquellos planes de P_C tales que todos sus elementos están en C_D. Se analizará el comportamiento del filtro para diversos subconjuntos C_D.

En este contexto se pasará a utilizar el filtro F, que se aplica entre el directorio y el planificador. Es decir, recibe el conjunto C_D y devuelve $C_D^F \subseteq C_D$. A partir de este conjunto se puede construir el conjunto de planes P_D^F.

El objetivo principal perseguido en esta evaluación es comprobar cómo se ve reducido el espacio de planes al aplicar el filtro. Recuérdese que el objetivo del filtro es reducir lo menos posible ese conjunto de planes, de forma que el abanico de alternativas que tenga el planificador sea lo más grande posible. Se

medirán:

- la proporción de planes que se pueden conseguir tras aplicar el filtro:

$$\frac{|P_D^F|}{|P_D|}$$

- la proporción de consultas para las que al menos se puede conseguir un plan:

Para terminar de definir el marco para realizar las pruebas hay que decidir:

a) Cómo generar el conjunto de todos los posibles planes P_C

b) Cómo elegir $C_D \subseteq C$, el conjunto de servicios disponibles

c) Cómo seleccionar un conjunto de planes históricos $P^H \subseteq P_C$ con los que entrenar el filtro

El conjunto de planes elegidos para entrenar el filtro va a ser el conjunto de planes original, es decir $P^H = P_C$. El motivo es que de esta manera podemos probar de forma más independiente el método de filtrado propuesto. Lógicamente, cuantos menos planes históricos se consideren menos precisa será la relevancia calculada y por tanto menos eficaz el filtro. Un estudio profundo sobre el método de aprendizaje se plantea como posible línea futura.

Lo que sí se va a analizar en detalle es cómo se comporta el filtro para diferentes servicios en el directorio. Para ello se van a realizar los experimentos para diversos porcentajes de servicios en el conjunto C_D del conjunto original C (100 % corresponderá a $C_D = C$).

El problema de la generación del conjunto de planes es más complejo y se describe en detalle en las siguientes dos secciones. Se han elegido dos métodos distintos para generar la colección de planes utilizadas en las pruebas. El motivo es que cada uno tiene unas características y asume ciertas hipótesis. El primero utiliza una matriz de relevancia generadora, y para la generación de

un plan cada servicio pertenece a ese plan según la probabilidad recogida en dicha matriz. El segundo método utiliza un grafo donde cada camino representa un plan. Se analizan todos los posibles planes que se pueden generar.

5.7.1.2. Generación de colección de planes mediante *matriz generadora*

Puesto que se asume como hipótesis la existencia de una relación entre clases de servicios que componen planes, se define una distribución que se va a tomar como partida de la generación de casos de prueba. Esa distribución no sería conocida en el caso real. Se denominará a esta distribución G, siendo $G(i, j)$ la probabilidad de que la clase c_i forme parte de un plan para la clase c_j.

Sea C el conjunto de todas las *clases de servicios*, a partir de esa distribución G se genera, para cada consulta $c \in C$, un conjunto de planes P_C^c que se considerará como el conjunto de todos los posibles planes que se podrían obtener para esa consulta c. Siendo $P = \{< c, P_C^c > | \forall c \in C\}$ el conjunto de todos los pares <*consulta, planes válidos*>.

La generación de los planes se realiza de la siguiente manera. Recuérdese que en cuanto a nuestro interés, un plan está formado por un conjunto de clases de servicios. Para generar un plan para la clase c_j, cada clase c_i pertenece a ese plan con una distribución de $Bernoulli(p = G(i, j))$. Por ejemplo, si $G(c_1, c_2) = 0.3$, entonces c_2 pertenece a un plan para c_2 con probabilidad 0.3. Repitiendo el experimento para cada clase se obtiene un plan. Y repitiendo lo mismo se obtienen nuevos planes, que pueden tener dimensiones y clases distintas según esa distribución.

Análisis del cálculo de Matriz de Relevancia. A continuación se va a analizar cómo afecta la forma en que se han generado los planes a partir de G para el cálculo de la matriz de relevancia.

Se recuerda cómo era el cálculo de la relevancia:

$$Relevancia(c_1, c_2) = \frac{\sum_{d=1}^{m} \frac{n_d}{d^\alpha}}{\sum_{d=1}^{m} \frac{N_d}{d^\alpha}}$$

Siendo n_d el número de planes para conseguir c_2 de dimensión d en los que aparece c_1, y N_d el número total de planes de dimensión d para conseguir c_2.

En el caso de que no se considerase la heurística de dimensión de planes ($\alpha = 0$) se tendría:

$$Relevancia(c_1, c_2) = \frac{\sum_{d=1}^{m} n_d}{\sum_{d=1}^{m} N_d} = \frac{n}{N}$$

Siendo n el número de planes para conseguir c_2 en los que aparece c_1, y N el número total de planes para conseguir c_2.

Si el conjunto de planes considerados (históricos) es suficientemente grande, $\frac{n}{N}$ tiende a un valor próximo a la probabilidad que se usó al generar los planes ($G(c_1, c_2)$), aunque no exactamente ese ya que habría que considerar que al generar los planes se descartaron aquellos casos en los que ninguna clase pertenece al plan (probabilidad(dimensión = 0)).

Si se asume que siempre al menos una clase fue elegida para un plan (razonable si $G(c, c) = 1$), entonces

$$Relevancia(c_1, c_2) = \frac{\sum_{d=1}^{m} n_d}{\sum_{d=1}^{m} N_d} = \frac{n}{N} = p(c_1 \in P_{c_2}) = G(c_1, c_2)$$

En cambio, en el caso en el que hubiera probabilidad de planes vacíos ($prob(dim = 0) > 0$), el análisis sería el siguiente.

El número de planes que realmente se generaron a partir de G fue $N_G = N + N_0$, siendo N_0 el número de planes de dimensión 0. Luego

$$G(c_1, c_2) = \frac{n}{N_G} \tag{5.3}$$

$$N_G = N + N_0 \tag{5.4}$$

$$N = N_G \cdot prob(dimension \neq 0) \tag{5.5}$$

$$N_0 = N_G \cdot prob(dimension = 0) \tag{5.6}$$

Siendo

$$prob(dimension \neq 0) = 1 - \prod_i (1 - G(c_i, c_2))$$

Por tanto,

$$Relevancia(c_1, c_2) = \frac{n}{N} = \frac{n}{N_G \cdot (1 - \prod_i (1 - G(c_i, c_2))} =$$
$$= \frac{G(c_i, c_2)}{(1 - \prod_i (1 - G(c_i, c_2))} = k \cdot G(c_i, c_2)$$

Como se puede observar la *relevancia* tiende a un valor proporcional al original. Esto no afecta en el proceso de filtrado, ya que los servicios se ordenan según su relevancia, por lo que lo importante es el orden de estos servicios y no su valor de relevancia. El orden será el mismo que si se tuviera G.

En el caso de que sí se considerase la heurística de dimensión de planes ($\alpha > 0$), hay que destacar que lo que puede variar es el hecho de que en el cálculo de la *relevancia* la aportación de una clase va ponderada por la inversa de la dimensión del plan al que pertenece (porque la probabilidad de que un plan largo se pueda componer después de filtrar es menor que uno corto).

En este caso, el cálculo de la relevancia es más complejo ya que ahora es importante saber n_d (antes era suficiente con la suma total), que depende no

solo de $G(c_1, c_2)$ sino también del resto ($n_d = N_d \cdot p(c_1 \in P_{c_2}|dim = d)$).

No obstante se puede demostrar que: Si $G(c_1, c) \geq G(c_2, c) \Rightarrow Relevancia(c_1, c) \geq Relevancia(c_2, c)$.

Esto implica que si no se cumple la hipótesis asumida por la heurística de dimensión de planes ("hay servicios que aparecen en planes más largos que otros"), esta heurística no afecta al filtrado de servicios, ya que la relevancia ordena los servicios igual que la generadora G.

Demostración:

Según se han generado los planes, en cada dimensión d el número de planes en los que aparecen c_1 y c_2 es:

$$n_d^{c_1} = prob(c_1 \in P_C^c|dim = d)) \cdot N_d \tag{5.7}$$

$$n_d^{c_2} = prob(c_2 \in P_C^c|dim = d)) \cdot N_d \tag{5.8}$$

Según se generaron los planes, una clase de servicio pertenece a un plan de forma independiente al resto de clases. Por tanto, dado un plan de cierta dimensión si $G(c_1, c) \geq G(c_2, c)$ entonces la probabilidad de que c_1 pertenezca a ese plan es mayor o igual a que c_2 lo haga ($prob(c_1 \in P_C^c|dim = d) \geq prob(c_2 \in P_C^c|dim = d))$). Por tanto, como $\forall d, G(c_1, c) \geq G(c_2, c)$ entonces $\forall d, n_d^{c_1} \geq n_d^{c_2}$.

$$Relevancia(c_1, c) \geq Relevancia(c_2, c) \Leftrightarrow \tag{5.9}$$

$$\frac{\sum_{d=1}^{m} \frac{n_d^{c_1}}{d^\alpha}}{\sum_{d=1}^{m} \frac{N_d}{d^\alpha}} \geq \frac{\sum_{d=1}^{m} \frac{n_d^{c_2}}{d^\alpha}}{\sum_{d=1}^{m} \frac{N_d}{d^\alpha}} \Leftrightarrow \tag{5.10}$$

$$\sum_{d=1}^{m} \frac{n_d^{c_1}}{d^\alpha} \geq \sum_{d=1}^{m} \frac{n_d^{c_2}}{d^\alpha} \tag{5.11}$$

Y como $\forall d, n_d^{c_1} \geq n_d^{c_2}$, entonces $Relevancia(c_1, c) \geq Relevancia(c_2, c)$.

Como conclusión de este análisis se observa que el método diseñado para la obtención de la *matriz de relevancia* a partir de planes históricos permite obtener una relevancia similar a la "generadora" en caso de no aplicar la heurística de la dimensión de los planes. El hecho de generar los planes de la manera que se ha realizado, donde la pertenencia de un servicio a un plan se realiza mediante un proceso independiente al resto de servicios y a la dimensión del plan, provoca que una de las hipótesis principales del enfoque de filtrado desarrollado no se cumpla (unos servicios pueden aparecer con distinta frecuencia según la dimensión de los planes). Sin embargo se ha visto que aun así, el hecho de aplicar dicha heurística consigue los mismos resultados, si bien no la misma matriz de relevancia pero sí la misma ordenación de servicios en cuanto a relevancia, que es lo verdaderamente importante.

5.7.1.3. Generación de colección de planes mediante *grafo de servicios*

El método presentado en la sección 5.7.1.2 para la generación del conjunto de planes de prueba tiene el problema de que los servicios son considerados independientes entre sí en el sentido de que la pertenencia a un plan no depende del resto de servicios (salvo la consulta).

En esta sección se describe un segundo método que sí refleja cierta dependencia de unos servicios con otros. Se utiliza un grafo como el de la figura 5.9 para representar planes. En dicho grafo los vértices (o nodos) representarían estados del mundo y los arcos son los servicios cuya ejecución permiten transitar de un estado a otro. Un camino entre dos vértices representa un plan compuesto de uno o varios servicios. Como en nuestro marco las consultas son descripciones de servicio, un plan será un camino entre los dos vértices del arco del servicio consulta.

Así por ejemplo, si la consulta describe el servicio s_1, un posible plan es $s_3 \oplus s_4$, pero también $s_2 \oplus s_8 \oplus s_4$, ya que $s_2 \oplus s_8$ permiten obtener s_3. Por tanto,

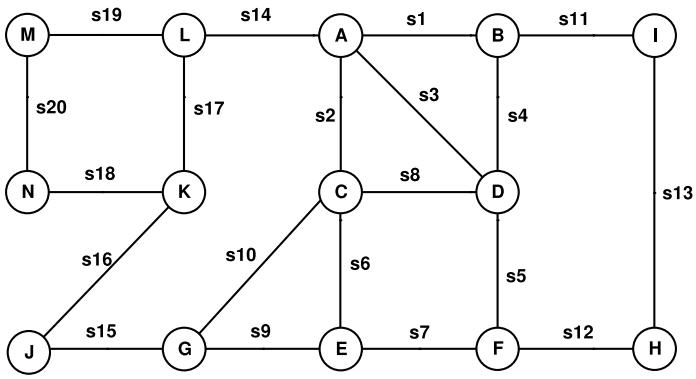


Figura 5.9: Grafo generador de planes

cualquier camino entre A y B (vértices de s_1) será un posible plan para s_1.

Aunque el grafo de la figura 5.9 es conexo[5], en general no tiene porqué serlo. Cada componente conexa representa un conjunto de servicios con cierta relación entre sí, quedando servicios que nunca forman parte de un mismo plan en diferentes componentes conexas.

Partiendo de este grafo, el conjunto de planes P_C^c para una consulta c será el conjunto de todos los caminos entre los vértices que conecta c.

Para los experimentos se utilizará un grafo conexo, que es el peor caso para nuestros intereses, ya que el filtrado de servicios afectará a los planes que se puedan conseguir. En cambio, si hay varias componentes conexas (que es un caso más real) el filtrado de servicios de otras componentes distintas a la consulta no afectará al número de planes por lo que el rendimiento será mejor que el obtenido en estas pruebas.

5.7.2. Análisis de los resultados

En los experimentos se han usado 20 clases de servicios. Se ha analizado el comportamiento del filtro por cada 5 % de servicios filtrados, es decir, pasando

[5]existe un camino entre cualquier par de vértices

todos, pasando el 95 % de los servicios, el 90 %, etc. Además, esas pruebas se han realizado considerando diferentes subconjuntos de servicios en el directorio ($C_D \subseteq C$), también cada 5 % de los originales (aunque por claridad, en las gráficas se van a presentar cada 10 %). Así, para una ausencia de por ejemplo el 20 % de los servicios ($\frac{|C_D|}{|C|} = 0.8$) se eliminan del conjunto total de planes (P_C) aquellos en los que aparece al menos uno de los pertenecientes a esos 20 % eliminados.

Se han analizado varias configuraciones del filtro:

1. Método de generación de planes: se han utilizado los dos métodos presentados en las secciones anteriores. El método de la *matriz generadora* (sección 5.7.1.2) se basa completamente en probabilidades, mientras que el método del *grafo de servicios* (sección 5.7.1.3) refleja mejor la dependencia de unos servicios con otros.

2. Técnica de refino de la matriz de relevancia (sección 5.2.3): se ha analizado usando dicha técnica y sin usarla.

3. Heurística de dimensión de planes (sección 5.2.2). Recuérdese que la aparición de un servicio (clase) en un plan se pondera por $\frac{1}{d^\alpha}$, es decir, $\alpha = 0$ anula la heurística y cuanto mayor sea más importancia se está dando a la dimensión del plan. Se ha analizado para los valores de $\alpha = 0, 1, 2, 3$.

En primer lugar se analizan las diferentes configuraciones de refino y heurística de dimensión de planes. A continuación, con la mejor configuración, se analizan en profundidad otras medidas. Se han realizado experimentos utilizando ambas formas de generar los planes.

5.7.2.1. Análisis de refinamiento y dimensión de planes

La tabla 5.6 muestra para diferentes configuraciones la proporción media de planes que se puede conseguir después del filtrado respecto al conjunto total de planes posibles (si no se filtrase).

Por ejemplo, la primera fila refleja que a partir del conjunto de planes generados mediante el método del grafo y que son tomados como los planes posibles en el experimento, sin refinar la matriz de relevancia y sin considerar la dimensión de los planes en el cálculo de la matriz de relevancia ($\alpha = 0$), el resultado medio de aplicar diferentes niveles de filtro (desde 0 % hasta el 95 %) es que el 12 % de los planes siguen siendo posibles de componer con los servicios que sí pasan el filtro. De momento no nos centraremos en la interpretación de estos valores, ya que se analizará en detalle más adelante. Lo que nos interesa ahora es analizar las diferencias entre unas configuraciones y otras.

Aunque estos datos corresponden a los valores obtenidos para diferentes subconjuntos (100 %, 95 %, ..., 5 %) de planes a partir del original (P_C), también se han analizado algunas porciones de P_C (por ejemplo sólo a partir del 50 %) obteniéndose las mismas conclusiones.

Generación	Refino	α	Proporción
Grafo	No	0	0.12
Grafo	No	1	0.31
Grafo	No	2	0.65
Grafo	No	3	0.64
Grafo	Sí	0	0.33
Grafo	Sí	1	0.41
Grafo	Sí	2	0.62
Grafo	Sí	3	0.65
Matriz	No	0	0.22
Matriz	No	3	0.23
Matriz	Sí	0	0.23
Matriz	Sí	3	0.22

Tabla 5.6: Proporción media de planes que se pueden conseguir después del filtrado de servicios

Para analizar cómo afecta la dimensión de los planes en el cálculo de la relevancia observamos cómo cambia la proporción de planes para distintos valores de α, manteniendo fijo el refino o no refino. Esta información se ha resumido

en la tabla 5.7. Se puede observar que, cuando se usan los planes generados mediante el método del grafo, considerar la dimensión del plan aumenta la proporción media de planes posibles. En concreto, para los valores 1 y 2 de α aumenta aunque a partir de $\alpha = 3$ ya se estabiliza. Además puede ver que el valor incrementado es más relevante cuando no se combina con el refino de la relevancia (un total acumulado de aproximadamente un 50 %) frente a cuando sí se hace ($\approx$ 30 %). Esto es normal pues el refino aporta otra forma de mejorar la relevancia estimada por lo que sin él tiene más margen de mejora.

Sin embargo, cuando se utilizan los planes generados mediante el método de la matriz, los resultados son los mismos independientemente del valor de α. En la tabla sólo se han reflejado los dos valores extremos (1 y 3) porque el resto es similar. Este resultado era el esperado según se demostró en la sección 5.7.1.2. Se recuerda que el motivo es que pertenencia de un servicio a un plan, independientemente de la dimensión del plan, no dependía de qué otros servicios pertenecen también a ese mismo plan.

Generación	Refino	α	Generación	Refino	α	Diferencia
Grafo	No	1	Grafo	No	0	0.18
Grafo	No	2	Grafo	No	1	0.34
Grafo	No	3	Grafo	No	2	-0.02
Grafo	Sí	1	Grafo	Sí	0	0.07
Grafo	Sí	2	Grafo	Sí	1	0.22
Grafo	Sí	3	Grafo	Sí	2	0.03
Matriz	No	3	Matriz	No	0	0.01
Matriz	Sí	3	Matriz	Sí	0	0.01

Tabla 5.7: Diferencia de proporciones para distintos valores de α

Ahora pasamos a analizar cómo afecta la técnica de refinamiento de la matriz de relevancia (sección 5.2.3) en el rendimiento del filtro. De nuevo, comparamos para diferentes configuraciones la diferencia entre aplicar o no refino. Los resultados se resumen en la tabla 5.8.

En este caso, de nuevo se produce un efecto parecido al anterior. En el caso de

Generación	Refino	α	Generación	Refino	α	Diferencia
Grafo	Sí	0	Grafo	No	0	0.21
Grafo	Sí	1	Grafo	No	1	0.10
Grafo	Sí	2	Grafo	No	2	-0.03
Grafo	Sí	3	Grafo	No	3	0.01
Matriz	Sí	0	Matriz	No	0	0.01
Matriz	Sí	3	Matriz	No	0	-0.01

Tabla 5.8: Diferencia de proporciones para refino y no refino

planes generados con el grafo, el refino aporta una mejora de relevancia, que se aprecia más cuanto menos aporta la heurística de planes cortos (α bajo). De nuevo la explicación es que ambos influyen en mejorar la relevancia pero entre ambos consiguen llegar a un valor "ideal" que no coincide con la suma de lo que logran independientemente.

Igualmente, en el caso de los planes generados con la matriz tampoco el refino aporta ninguna mejora. En este caso, podría parecer erróneo, ya que la matriz generadora sí que puede reflejar que un servicio s_1 es muy relevante para s_2, que s_2 es muy relevante para s_3 y, en cambio, s_1 es poco relevante para s_3. El objetivo del refino es aceptar s_1 como relevante para s_3, ya que en caso de ausencia de s_2 quizá se pueda utilizar s_1 en su lugar (posiblemente en combinación de otros servicios). Sin embargo, es muy difícil que esos posibles planes alternativos hayan sido considerados como planes posibles (conjunto P_C) puesto que la probabilidad de que s_1 esté en un plan es muy pequeña.

Como conclusión al análisis de dimensión y refino podríamos concluir que la mejor configuración es con un α igual a 2 o 3, y que para esos valores de α el refino no aporta nada adicional. Sin embargo se va a seguir considerando el refino puesto que las formas de generar los planes han sido mediante heurísticas y puesto que tampoco el refino empeora los resultados frente a otra distribución intrínseca de servicios en planes podría ser interesante.

Esta configuración (Refino y $\alpha = 3$) es la que se va a utilizar en la siguiente sección para profundizar el análisis de los resultados obtenidos.

5.7.2.2. Análisis de la recuperación

En esta sección se va a analizar cómo afecta el filtrado de servicios la recuperación de la planificación, esto es, cuántos planes que se podrían obtener sin filtrar servicios siguen siendo factibles. Este análisis es interesante puesto que el objetivo principal del filtro es descartar servicios pero reduciendo el mínimo posible el espacio de planes, dejando la responsabilidad de componer el plan al planificador con el que se combine.

Se recuerda el planteamiento. Sea C el conjunto de todos los servicios y un directorio D que contiene un subconjunto de ellos ($C_D \subseteq C$), con P_C y P_D el conjunto de todos los posibles planes componibles a partir de esos conjuntos, respectivamente. Sea C_D^F el subconjunto de servicios que pasan el filtro a partir de los contenidos en el directorio ($C_D^F \subseteq C_D$). Se trata de analizar el índice de recuperación que se consigue tras aplicar el filtro, es decir, la proporción de planes que se siguen pudiendo componer:

$$\frac{|P_D^F|}{|P_D|}$$

En la figura 5.10 se muestran los resultados obtenidos para la colección de planes creada con el *grafo generador*. Esta figura muestra varias curvas, cada una de las cuales corresponde al porcentaje de servicios totales que se encuentran en el directorio, es decir $\frac{|C_D|}{|C|}$. Para cada curva se muestra el valor de la proporción de planes que se mantienen para diferentes porcentajes de filtrado (se han medido cada 5 %).

Como era de esperar el número de planes componibles es mayor cuantos más servicios pasan el filtro. Además es interesante observar que cuanto menos completo está el directorio mejores resultados se obtienen: la curva 30 %, que corresponde al caso en el que sólo el 30 % de los servicios se encuentran disponibles es la que mejor índice consigue, le sigue la del 40 % de servicios, etc. Esto no se cumple cuando se filtran muchos servicios (pasan sólo el 15 % o menos) pero no debe considerarse representativo. La explicación a este hecho es la siguiente y se puede pensar en el símil del grafo para en su razonamiento.

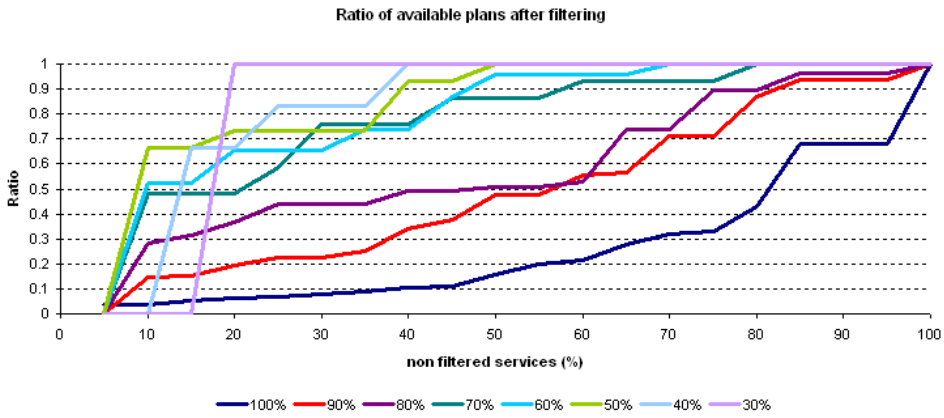


Figura 5.10: Proporción de planes componibles después de filtrar. Colección de planes: Grafo Generador. Cada curva corresponde con el porcentaje de servicios del total que se encuentran disponibles en el directorio

.

Se están considerando todos los posibles planes posibles a partir de un conjunto de servicios, o lo que es lo mismo, todos los caminos en un grafo (que además hemos considerado conexo). Cuando se filtran servicios, por muy bien estimada que esté la relevancia de los servicios, lógicamente algunos planes ya no serán factibles (salvo que existan servicios que no sirvan para ningún plan, que no es nuestro caso), equivale a eliminar arcos del grafo por lo que hay menos caminos.

Ahora bien, cuando en el directorio no están todos los servicios, el conjunto original de planes es menor y el grafo ya no es el original sino que le faltan arcos. Entonces, puede ocurrir que haya servicios de los que permanecen en el directorio que ahora pertenezcan a menos planes o incluso a ninguno. Por ejemplo, para el caso del grafo de la figura 5.9, si en el directorio no está el servicio s_{11}, entonces la aparición de s_{12} y s_{13} en el conjunto de planes se reduce drásticamente (quedan muy pocos caminos que pasen por alguno de esos arcos). Y si faltan s_{14} y s_{16} entonces el grafo no es conexo y los servicios de cada componente conexa no aportan nada a los de las demás.

En estas situaciones donde se va reduciendo el conjunto de planes a partir de los servicios del directorio es cuando más ventaja se puede sacar del filtro aquí propuesto, ya que se va haciendo importante el hecho de beneficiar la presencia de servicios en planes cortos y el refino de la relevancia, precisamente para poder considerar planes alternativos.

Observando los valores de las curvas se puede apreciar el excelente comportamiento cuando en el directorio hay menos del 70 % de los servicios totales.

Además, recuérdese que en un caso real seguramente el grafo implícito que correspondería a todos los planes posibles podría ser no conexo (algunos servicios nunca estarían en planes con otros) o en general tener muy pocos arcos, lo que produciría un efecto similar al de la ausencia de algunos servicios en el directorio, siendo favorecido nuestro filtro.

La figura 5.11 muestra el análisis con planes generados por el método de la matriz. En este caso, puesto que el método de generación es aleatorio los resultados no deben considerarse muy significativos. Se observa que también se mantiene un mejor comportamiento cuantos menos servicios tiene el directorio pero en este caso las diferencias son muy pequeñas. Además, los valores son bastante más bajos que con la colección de planes anterior, debido a que ahora no se refleja en la colección aspectos importantes como la correlación entre los servicios en su pertenencia a los planes.

5.7.2.3. Análisis de la completitud

Además del índice de recuperación de planes estudiado en la sección anterior, otra medida importante es si a pesar del filtrado de algunos servicios es posible todavía encontrar algún plan, lo que llamaremos *completitud*. Esto puede ser de gran importancia puesto que en determinadas situaciones (por ejemplo, escasez de recursos computacionales) puede ser importante reducir el conjunto de servicios a costa de un menor índice de recuperación pero si el índice de completitud es alto.

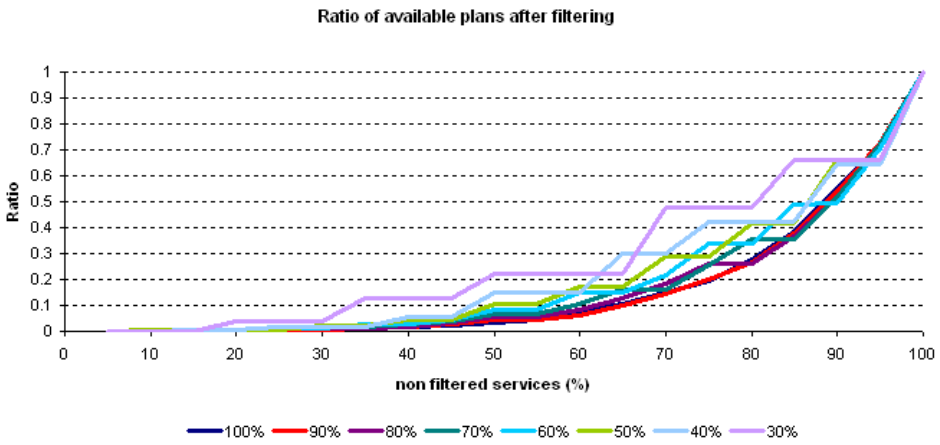


Figura 5.11: Proporción de planes componibles después de filtrar. Colección de planes: Matriz Generadora

.

La figura 5.12 muestra el análisis con planes generados por el método del grafo. La forma de generar las pruebas y la generación de la gráfica es similar al aplicado para estudiar la recuperación. Se observa que, salvo cuando se filtra un porcentaje muy alto de servicios (pasan muy pocos), la completitud es bastante alta (entre 0.9 y 1), es decir que casi siempre se puede conseguir al menos un plan con los servicios que pasan el filtro.

La figura 5.13 muestra el análisis con planes generados por el método de la matriz. De nuevo los resultados deben tomarse con cierta cautela debido a la forma usada para generar los planes. En este caso, el índice de completitud sigue linealmente la proporción de servicios no filtrados.

5.8. Discusión

En esta sección se ha propuesto un método genérico para filtrado de servicios para composición. El objetivo que se ha perseguido es reducir el número de servicios que se pasan como entrada a un planificador, por lo que el tiempo

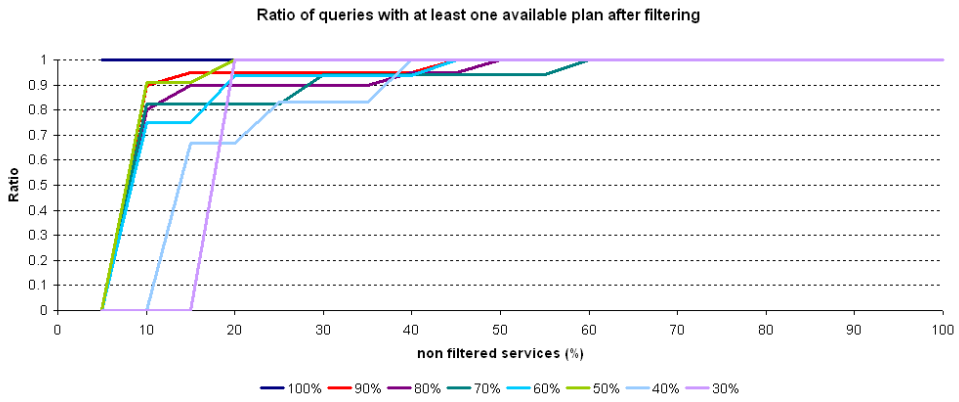


Figura 5.12: Proporción de consultas con al menos un plan componible después de filtrar. Colección de planes: Grafo Generador

.

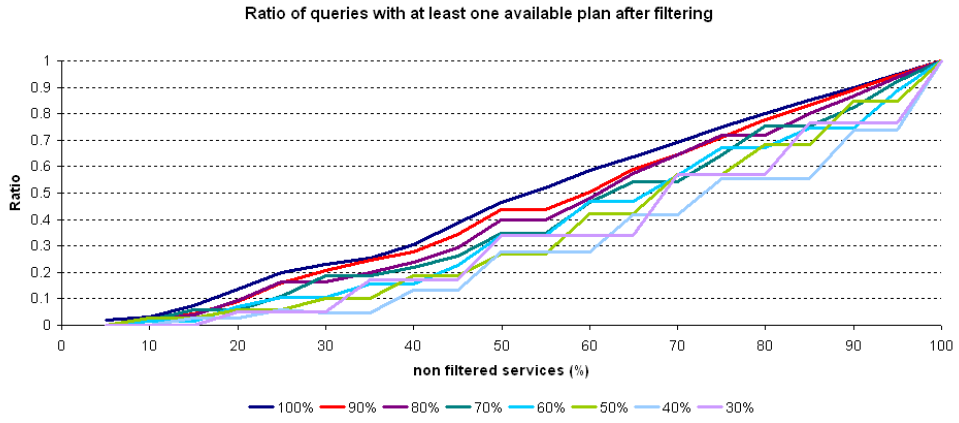


Figura 5.13: Proporción de consultas con al menos un plan componible después de filtrar. Colección planes: Matriz Generadora

.

empleado por éste debe ser menor, pero reduciendo lo mínimo posible el espacio de planes posibles, de forma que afecte lo menos posible al resultado producido por el planificador utilizado. La motivación de esta propuesta es que ante la presencia de un número grande de servicios en un directorio la ejecución de un algoritmo de planificación puede ser computacionalmente inaceptable.

En el método propuesto se realiza una estimación de lo relevante que es un servicio para una consulta determinada, pero lógicamente sin intentar buscar un plan, que es la labor del planificador y, precisamente, cuya eficiencia se pretende mejorar.

El filtro se basa en varias suposiciones. En concreto, se asume que existe una relación entre *clases de servicios* y pertenencia a planes, es decir que unas clases de servicios suelen estar más presentes en los planes que se pueden componer que otras. Además de la existencia de esa relación, también se asumen que se conoce dicha función que para un servicio es capaz de clasificarlo en *clases de servicios*. Estas dos suposiciones son difíciles de demostrar, más aún cuando el campo está todavía tan poco desarrollado. Según avance la aplicación práctica de los enfoques de composición de servicios se podrán analizar bancos de datos históricos, por lo que esa es una línea de trabajo a tener en cuenta a corto y medio plazo. Aquí se han propuesto dos métodos de clasificación de servicios, uno basado en los roles que juegan en las interacciones y otro basado en las categorías de los servicios.

También se asume, por sencillez y porque es lo más habitual, que todos los servicios de un plan son necesarios para poder ejecutar el plan.

Para llevar a cabo la estimación de la relevancia de un servicio se utilizan varias heurísticas. En particular, se considera que un servicio es más relevante cuando aparece en planes más cortos. Esto es así porque, suponiendo que se filtran varios servicios, es más probable que un plan largo (con muchos servicios) deje de cumplirse porque al tener más servicios es más probable que alguno de ellos se filtre.

Puesto que el método propuesto aquí actúa como filtro que sólo proporciona

algunos servicios al planificador, la recuperación (proporción de planes disponibles después de filtrar) puede verse afectada si se filtra algún servicio que permite componer un plan. Esto puede ser un problema pero recordemos que el objetivo es que el planificador pueda en un tiempo razonable proponer una solución que de lo contrario seguramente no podría debido a la escasez de recursos computacionales. Además, puede que sea imposible realizar un filtrado de cierta cantidad de servicios sin que alguno de los servicios filtrados pueda servir para componer un plan. Si el filtro funciona como se espera, la recuperación va a depender de la cantidad de servicios que se filtren.

Igual que ocurría en el caso del descubrimiento de servicios, debido a la inmadurez de la tecnología, no existe todavía un conjunto de casos de prueba representativo que se pueda usar para validar nuestra propuesta. En el caso de la composición se cuenta con muchas menos posibilidades de reutilizar alguna colección de planes ya que los pocos existentes no usan información semántica. Existen algunas colecciones para servicios web (no semánticos), por ejemplo de concursos, pero no nos sirve por falta de semántica (suelen estar generados aleatoriamente).

Para poder realizar experimentos que permitan validar nuestra propuesta se ha realizado generando planes mediante dos métodos distintos, uno basado en una matriz de probabilidad de pertenencia de servicios a planes con generación aleatoria, y otro basado en una representación mediante un grafo de planes. A partir de colecciones generadas a partir de esas representaciones ser han realizado experimentos.

Además de lo anterior, los experimentos se han llevado a cabo de forma independiente a un planificador de servicios compuestos concreto, evitando de esta manera que posibles deficiencias del planificador influya en los resultados obtenidos.

Los resultados de los experimentos se han detallado y discutido en la sección 5.7.2. Se ha comprobado que la tasa de planes que siguen siendo posibles es bastante aceptable incluso tras filtrar bastantes servicios. Además, casi siempre permanece al menos un plan posible (salvo con porcentajes de filtrado muy

altos).

Existen otras propuestas en la literatura sobre filtros aplicados a composición de servicios. En [112] se presenta un filtro interactivo de ayuda al usuario para composición de servicios. El usuario define las entradas y salidas del servicio que busca y el sistema le muestra aquellos que permiten obtener alguna de las salidas o coinciden en alguna entrada. El usuario va componiendo el servicio y el sistema le ayuda mostrándole servicios compatibles con alguna entrada o salida. En MetaComp [12] plantean dos métodos para la recuperación de servicios del directorio que serán entrada al planificador: uno basado en las categorías del servicio, que son definidas por el usuario (cliente), y otro en donde se aceptan aquellos servicios que disponen al menos de una entrada, salida, precondición o efecto compatible con el servicio compuesto buscado. Sin embargo, actualmente ninguno de esos métodos lo han integrado en su planificador. Constantinescu [11] propone un enfoque en el que, con el fin de optimizar la interacción con el directorio por diferentes algoritmos de composición de servicios que exploten heurísticas específicas de la aplicación, el directorio permite que el usuario especifique funciones de selección y ranking escritas en un lenguaje declarativo. En Synthy [1], se utiliza un enfoque similar al nuestro en el sentido de que se aplica antes de planificar, en este caso el filtro considera relevantes aquellos servicios que pueden contribuir a los objetivos (al menos comparte un efecto con un objetivo) o a las precondiciones de cualquier servicio que potencialmente puede contribuir al objetivo.

La forma más intuitiva y sencilla de integrar el método de filtro propuesto es ,como se describió en la sección 5.6, integrado con un planificador de servicios. Con esa arquitectura, el planificador recibe todos los servicios disponibles en el directorio, aplica el filtro y con los servicios que pasan dicho filtro ejecuta el proceso de planificación. Una vez compuesto el plan, en caso de conseguirlo, realimentaría el filtro con información sobre el plan compuesto de forma que se pueda utilizar esta información histórica para aprender de la experiencia. Sin embargo, esta arquitectura tiene dos inconvenientes principales:

1. Se considera un único planificador, lo cual constituye un elemento único

de fallo y un cuello de botella.

2. Cuando se necesita componer un plan, deben comunicarse al planificador, para ser procesados por el filtro, todas las descripciones de los servicios del directorio, lo que puede suponer gran cantidad de tráfico de información por la red.

Una solución al primero de los problemas sería disponer de varios planificadores. Esta idea, aparentemente bastante simple, requiere de un análisis un poco más profundo. Se pueden considerar varias implementaciones de esta idea.

- Cada planificador tiene su propio filtro. Esta implementación es la más sencilla de llevar a cabo. El inconveniente que tiene es que cada filtro aprende la función de relevancia a partir de los planes que ha compuesto el planificador en el que está integrado, y no del conjunto total de planes compuestos. A largo plazo, cuando se han generado un número grande de planes el problema se suaviza.

- Otra posibilidad es disponer de un único filtro utilizado por todos los planificadores. Tiene la ventaja respecto al anterior que puede utilizar todos los planes para aprender la función de relevancia. Por contra se puede convertir en un cuello de botella y punto único de fallo. Además esta solución puede no ser adecuada si existen diferentes tipos de planificadores que, por ejemplo, difieran en la función de utilidad de los planes, esto es, para un mismo conjunto de servicios consideren como óptimo un plan distinto.

- Una solución híbrida consiste en tener un filtro único replicado en cada planificador. Esto requiere una coordinación entre los diferentes planificadores para compartir la información sobre los planes que han compuesto (o sus matrices históricas).

El problema del tráfico por la red podría reducirse si el filtro se desplaza al directorio, con lo que sólo se transmitirían por la red las descripciones de los

servicios que pasan el filtro. Por supuesto, se necesitaría que los planificadores enviasen el resultado de los planes que compongan al filtro, pero ese tráfico sería mucho menor que transmitir todos las descripciones de servicios. Además, esta opción permitiría un acceso incremental a las descripciones de servicios, es decir, el planificador podría solicitar los n servicios más relevantes y, en caso de no encontrar un plan, solicitar los siguientes m más relevantes, de forma parecida a la propuesta por Constantinescu [11].

Otra solución podría consistir en transmitir las descripciones de servicios en dos pasos: en un primer paso se enviaría sólo la información relativa a las *clases de servicios*, que es la que usa el filtro, y en un segundo paso sólo las descripciones completas de los servicios aceptados por el filtro (que también se podría solicitar de forma incremental). Esto requiere integrar en el directorio la función de extracción de clases.

Además de todo lo anterior hay que considerar que el directorio podría estar distribuido, lo cual es bastante habitual y recomendable en grandes sistemas abiertos, lo que incrementaría el nivel de complejidad para las soluciones que requieren llevar componentes del filtro al directorio.

Capítulo 6

Aplicaciones

Este capítulo se dedica a la validación de las propuestas realizadas en esta tesis desde dos perspectivas distintas. En primer lugar, en la sección 6.1, se propone la aplicación de sistemas multiagente orientados a servicios en general, y el enfoque basado en roles para la descripción de servicios en particular, en el dominio de la gestión de transporte mediante dos aplicaciones: una para la gestión de tráfico rodado y otra de líneas de autobuses. En la sección 6.2 se describe la utilización de los componentes software que instrumentan las propuestas realizadas en la construcción de una plataforma para la coordinación de servicios en entornos móviles que ha sido utilizada en la construcción de una aplicación para la asistencia en casos de emergencias médicas.

6.1. Aplicación en la gestión de transporte

Los sistemas de ayuda a la decisión (SAD) proporcionan ayuda a las personas implicadas en procesos complejos de toma de decisiones. Los primeros SAD fueron concebidos como bases de datos simples para el almacenaje y la recuperación de la información relevante a la decisión [111]. Sin embargo, pronto llegó a ser evidente que el problema principal para el decisor no es tener acceso a datos pertinentes, sino entender su significado. Los SAD actuales ayudan a

los tomadores de decisiones a explorar las implicaciones de sus juicios, para tomar las decisiones basadas en el conocimiento.

Los SAD basados en el conocimiento son particularmente relevantes en los dominios donde los operadores humanos tienen que tomar decisiones operacionales con respecto a la gestión de procesos industriales o ambientales complejos. Debido a la inherente distribución (espacial, lógica y/o física) de estos dominios, se ha hecho popular un enfoque distribuido en la construcción de los SAD [26, 87]. Los agentes de ayuda a la decisión son responsables (individualmente) de partes del proceso de toma de decisiones de manera racional y (semi-) autónoma: recogen y facilitan los datos relevantes de la decisión, pero también proporcionan servicios avanzados de razonamiento para analizar el significado de esta información.

En el contexto del proyecto DAMMAD (Diseño y Aplicación de Modelos Multiagente de Ayuda a la Decisión)[1], se desarrollaron dos prototipos de sistemas multi-agente para ayuda a la decisión, uno para la gestión de trafico en el área de Bilbao y otro para gestión de líneas de autobuses en la ciudad de Málaga. Se puede encontrar información detallada sobre estos sistemas en [88, 89].

Estos sistemas se desarrollaron siguiendo un enfoque de sistemas multiagente, donde cada agente es responsable de jugar ciertos roles sociales en una parte del sistema global.

En esta sección se replantea el diseño de estos sistemas para sacar partido al enfoque orientado a servicios y las ideas propuestas en esta tesis sobre la descripción y equiparación de servicios. Con este enfoque los sistemas se convierten en más flexibles y se pueden adaptar más fácilmente a cambios en la organización del sistema, como agrupación o separación de responsabilidades entre agentes, variación de los agentes participantes, etc.

En el siguiente apartado se describe una arquitectura abstracta multi-agente que se propuso para sistemas de ayuda a la decisión. Esa arquitectura se instancia para cada aplicación concreta y encaja en el nuevo enfoque de orientación

[1]Financiado por el Ministerio de Ciencia y Tecnología bajo el código TIC 2000-1370-C04

a servicios.

6.1.1. Arquitectura abstracta

En la línea de las actuales tendencias en la ingeniería del software orientada a agentes, primero se modela el SAD (Sistema de Ayuda a la Decisión) basado en agentes en términos de conceptos organizacionales, que después se refinan para dar lugar a un modelo de agentes que será la base de la arquitectura abstracta.

Un aspecto importante en los SAD actuales es la interacción con el decisor para ayudarle en el análisis de relevancia de sus posibles decisiones. Para desarrollar un modelo organizacional genérico de los SAD, es natural partir de un análisis de los diálogos típicos entre decisores y SAD. En los SAD con múltiples decisores además suelen estar presentes interacciones de intermediación y negociación.

Por otra parte, los roles en los SAD basados en agentes también requieren competencias del dominio, por lo que especializamos los roles comunicativos produciendo roles sociales basados en elementos de una ontología del dominio del cual informan, explican, etc. Esa ontología al menos contendrá conceptos relativos a problemas, causas de problemas y acciones de control. Un SAD por lo menos debería incluir servicios de razonamiento para (1) identificar situaciones con opciones de decisión (clasificación) (2) expresar problemas del sistema en términos de características causales de la situación (diagnosis), (3) generar varias alternativas de decisión (planificación de acciones) y (4) simular consecuencias potenciales de las decisiones (predicción).

Para desarrollar un modelo de agentes para ayuda a la decisión, los roles sociales tienen que hacerse corresponder con tipos de agentes que finalmente jugarán esos roles en el SAD. Normalmente se dan los dos siguientes casos:

- un agente - varios roles: este es el enfoque habitual dirigido a simplificar la arquitectura software (y evitando una replicación de conocimiento

innecesaria)

- un rol - varios agentes: debido a la "distribución a priori" que suele estar presente en dominios de ayuda a la decisión, suele ser conveniente que varios agentes jueguen el mismo rol, pero en diferentes partes de un sistema. De esta forma, el modelo de agentes puede reflejar una organización humana, reducir los requisitos de comunicación, o simplemente reducir la complejidad de los procesos de razonamiento necesarios.

Teniendo en cuenta las consideraciones anteriores, se obtienen diferentes tipos abstractos de agentes para SAD. Hay que tener en cuenta que, con el objetivo de reflejar lo mejor posible la distribución a priori, existirán varias instancias de los siguientes tipos de agentes:

- Hay dos clases de *Agentes de Conexión (Connection Agents)* que interactúan directamente con el entorno: los *Agentes de Datos (DA)* juegan el rol de informador con respecto al estado actual de cierta parte del sistema. Están encargados de recoger información de fuentes diferentes, como sensores o bases de datos, y su distribución a los agentes correspondientes. Por su parte, los *Agentes de Implementación de Acciones (AIA)* están encargados de ejecutar realmente las acciones que el decisor ha elegido.

- Los *Agentes de Gestión (MA, Management Agents)* juegan roles de informadores, asesor y explicador. Por tanto, necesitan modelos de conocimiento que les permitan informar justificadamente de problemas, causas, estados futuros potenciales, etc., así como sugerir acciones de control. Pueden basarse en los servicios de *Facilitadores de Coordinación (CF)* que proporcionan ayuda para interacciones de negociación y descubrimiento de agentes. Estos agentes son especialmente relevantes en SAD cuando hay grupos de decisores.

- Los *Agentes de Interfaz de Usuario (UIA)* juegan roles en representación del usuario. Son capaces, secuenciando o entrelazando conversaciones, de

responder a una variedad de preguntas (p. ej. "¿Qué está pasando en S?", "¿Qué puede pasar en S si ocurre el evento E?", etc.)

- Por último, los *Agentes Periféricos (PA, Peripheral Agents)* representan la infraestructura de soporte para SAD. Los *Directory Facilitators (DF)* y *Agent Management Systems (AMS)* que forman parte de la arquitectura abstracta de FIPA son ejemplo de este tipo de agentes. También incluyen agentes que proporcionen servicios adicionales útiles para el dominio de aplicación.

La figura 6.1 muestra la arquitectura abstracta multiagente para SAD. La instanciación de esta arquitectura a un dominio particular puede inducir una subdivisión adicional de tipos de agentes abstractos (por ejemplo para permitir diálogos más flexibles con el decisor). Además, la integración de software heredado puede requerir que un tipo abstracto de agente finalmente se implemente como una sociedad de agentes.

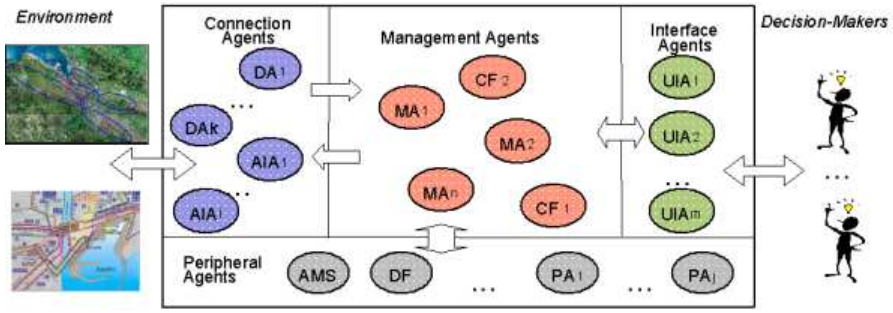


Figura 6.1: Arquitectura abstracta para SAD

En el nuevo diseño de las aplicaciones siguiendo un enfoque orientado a servicios, esta arquitectura sigue siendo válida, donde los agentes proveen y son consumidores de servicios. El agente *DF (Directory Facilitator)* cobra ahora especial importancia ya que es el elemento donde cada agente debe registrar los servicios que proporciona y donde se buscan los servicios necesitados. Ese elemento puede realizar ambas tareas (directorio y búsqueda) o bien separarse si se considera conveniente.

6.1.2. Sistema para gestión de tráfico rodado

6.1.2.1. Descripción del Dominio

Este sistema fue desarrollado pare el control de tráfico rodado en la red principal de carreteras de circunvalación del Gran Bilbao. En el centro de gestión de movilidad se recibe información sobre el tráfico provenientes de sensores situados en las carreteras. En base a estos datos, los operadores de tráfico tienen que tomar decisiones sobre qué acciones de control llevar a cabo para solventar o minimizar las congestiones. Estas acciones incluyen (i) mostrar mensajes en paneles de señalización de mensajes variables (PSMV) instalados en las carreteras para avisar a los conductores sobre problemas de tráfico o recomendar rutas alternativas, y/o (ii) contactar con las autoridades locales para que envíen personal adecuado para gestionar la situación. El sistema de ayuda a la decisión tiene el propósito de asistir a los operadores en su tarea de gestión, ayudándoles a configurar planes de control consistentes para la red de carreteras global, y hacer el mejor uso de los dispositivos de señalización disponibles desde una perspectiva global. Los operadores del centro de control dividen la red de carreteras en *áreas problema* para facilitar la detección y control de problemas de tráfico atendiendo no sólo a su distribución física sino a la lógica del flujo de tráfico.

6.1.2.2. Modelado de Interacciones

La figura 6.2 muestra una conversación típica entre un operador de un centro de control y el SAD[2]. Comienza con una *notificación*, por parte del SAD, de la presencia de cierto problema en alguna parte del área monitorizada. Entonces, el operador debe tomar una decisión tan rápida como sea posible, y pide al SAD *asesoramiento* sobre una propuesta de mensaje(s) para ser mostrados en paneles PSMV. Además, el operador puede pedir *explicaciones* del razonamiento realizado por el SAD, así como requerir una estimación del

[2]DSS en inglés

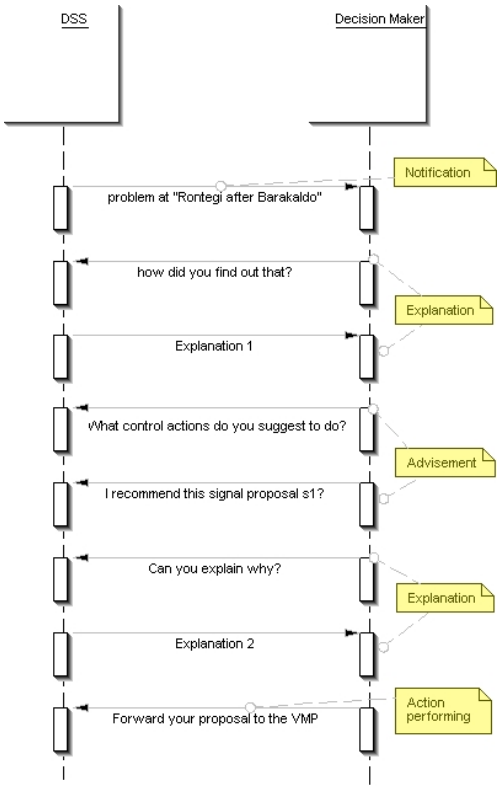


Figura 6.2: Ejemplo de conversación típica entre un operador y el SAD

impacto de algunas acciones de control (aunque esto no está representado en la figura).

Analizando el diálogo anterior, identificamos varios *tipos de interacciones sociales*: (i) *notificación* de problemas de tráfico, (ii) *explicación* de problemas de tráfico, (iii) *asesoramiento* sobre configuración de mensajes en PSMV, (iv) explicación de la propuesta de configuración de mensajes, y (v) *ejecución de acciones*.

Como se comentó en el capítulo 3, las interacciones sociales (dominio) se abstraen en interacciones comunicativas (genéricas). La figura 6.3 muestra un ejemplo donde el rol *VMSConfigAdvisor* es generalizado primero en un rol *TransportationAdvisor* y luego en un rol *Advisor*. Análogamente, la interac-

ción *VMSConfigAdvisement* se puede generalizar primero en una interacción *TransportationAdvisement* y después en una interacción *Advisement*, en la que el *Advisor* informa al *Advisee* sobre sus creencias con el objetivo de persuadirle de la precisión de esas creencias. Obsérvese que la interacción *Advisement* es una especialización de *ClosedActionPerforming* que a su vez lo es de *Communication*, como ya se vio en la sección 3.3.

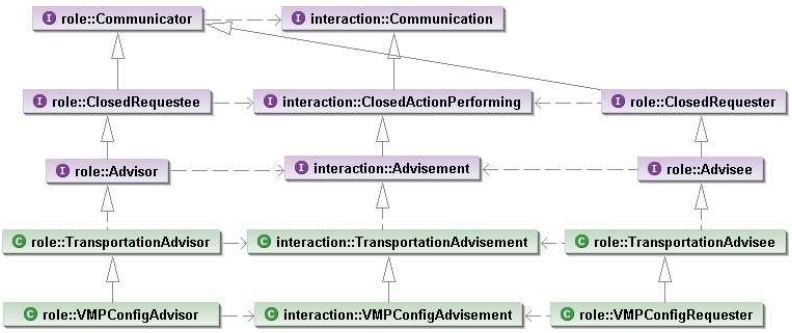


Figura 6.3: Proceso de abstracción de la interacción de asesoramiento de configuración de mensajes en paneles

La figura 6.4 muestra otro ejemplo, en este caso una notificación de problemas de tráfico (*TrafficProblemNotification*), donde un *TrafficProblemNotifier* alerta a un *TrafficProblemNotified* sobre un problema que ha sido identificado a partir del estado actual de la vía. Se observa que una *notification* es una clase específica de interacción de *intercambio de información*.

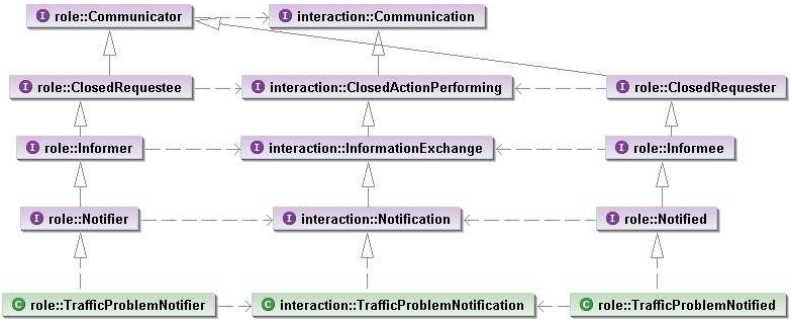


Figura 6.4: Proceso de abstracción de la interacción de notificación de problemas de tráfico

Análogamente, del análisis de diálogos adicionales en este dominio pueden indentificarse otras interacciones y ser generalizadas. Algunas de ellas son varias interacciones de *explicación* (sobre problemas de tráfico, propuestas de control, etc.), *intercambio de información* (estado de tráfico en carreteras, causas de problemas, estimación de impacto de acciones propuestas, estado general de un área problema, ...), *subscripción* (para ser notificado), etc.

El resultado de este análisis es una ontología básica de roles en el dominio de gestión de tráfico rodado que extiende la descrita en la sección 3.3, y donde la mayoría de los roles genéricos pueden reutilizarse en otros dominios.

6.1.2.3. Servicios

Realizando un análisis exhaustivo de los diferentes diálogos que pueden darse se identifican los siguientes servicios.

- *Datos de tráfico.* Este servicio proporciona información sobre datos de tráfico (velocidad, ocupación, intensidad, ...) obtenidos de sensores u otras fuentes de información en un área de la ciudad. Puede realizar un procesamiento sencillo de los datos, como calcular la media de varios instantes temporales, agrupar datos de varios carriles de una misma sección, etc. También puede completar y/o filtrar datos con ruido (por ejemplo debido a problemas de transmisión) haciendo uso de series de datos históricas. Puede participar en dos tipos de interacciones para proveer el servicio:
  - *Intercambio de información*, donde, a petición del usuario del servicio, se proporcionan los datos solicitados para lo cual juega el rol *informer*
  - *Notificación de datos.* En este caso, de forma proactiva, el servicio notifica los datos a las entidades interesadas. El motivo de la notificación puede ser porque se haga periódicamente, o porque se acaba de obtener la información. El servicio juega el rol *notifier* en

la interacción. Además, para poder jugar esta interacción, necesita que los clientes sean capaces de haberse subscrito previamente a este servicio (roles *subscriber*).

- *Detección de problemas de tráfico*. Es capaz de detectar problemas de tráfico que están ocurriendo en un área problema, a partir de la información básica (la obtenida de sensores). Para realizar esta tarea contiene modelos de conocimiento basados en escenarios de problemas prototípicos. A cada problema detectado se le asocia un nivel de severidad que es una función del exceso de tráfico. Igual que ocurría con el servicio de *Datos de tráfico*, este servicio puede proporcionarse mediante interacciones de *intercambio de información*, a petición expresa del cliente, o mediante una *notificación* cada vez que se detecta el problema, de forma que se procese lo antes posible. En este último caso, de nuevo, se requiere que el cliente pueda participar en la interacción de *subscripción*. Además, el usuario, una vez recibida una notificación o información sobre un problema en cierta área, puede solicitar una *explicación* de los motivos que han llevado a esa conclusión.

- *Estado general de área problema*. Este servicio informa sobre el estado general (libre, medio o con problemas) de un área problema. Sólo admite la interacción de *intercambio de información*, es decir, siempre es un cliente el que solicita dicha información.

- *Propuesta de configuración de PSMV*. Es capaz de recomendar una propuesta de configuración de mensajes en paneles PSMV, para minimizar ciertos problemas de tráfico. Para poder componer una propuesta, en esa interacción de *asesoramiento* puede requerir un *intercambio de información* sobre el estado del tráfico de ciertas áreas circundantes. Además, una vez realizada una propuesta también puede ser requerido para que la *explique*.

- *Resolución de conflictos en PSMV*. Los servicios de propuesta de configuración de paneles PSMV suelen realizar propuestas de señalización para resolver los problemas en las áreas para las que han sido encomen-

dados. Normalmente no tienen una visión total de la red de tráfico por lo que es posible que varios de ellos intenten utilizar los mismos paneles para mostrar diferentes mensajes y esto no sea posible. Este servicio se encarga de resolver este tipo de conflictos en base a ciertas políticas de prioridades: en primer lugar a la gravedad del problema y en segundo lugar a su localización. La importancia de la localización varía por área problema y franjas horarias. El servicio juega el rol de *negociador* y además también puede explicar sus conclusiones (*explainer).*

- *Visualización de mensajes en PSMV.* Este servicio se encarga de llevar a cabo las acciones de enviar los mensajes correspondientes a los paneles. Participa en interacciones de *ejecución de acciones* iniciadas por el cliente, por lo que juega el rol *requestee.*

La tabla 6.1 muestra la lista de servicios para la gestión de tráfico rodado, junto con su descripción en términos de roles. Cada tipo de interacción aparece en una línea, con los *provider roles* y, en su caso, los *depending roles.* La última columna indica los tipos de agentes proveedores, que serán analizados en la siguiente sección.

6.1.2.4. Arquitectura

En esta sección se describe la arquitectura de agentes concreta utilizada para la gestión de tráfico, indicando los servicios que proporciona cada agente. La arquitectura del prototipo es una instanciación de la arquitectura abstracta (figura 6.1) en la que se han utilizado los siguientes tipos de agentes:

- *Data Agents (DA).* Están asociados a áreas problema. Cada uno de ellos proporciona el servicio de *Datos de tráfico* para un área problema. Registran dicho servicio en el directorio. Como se comentó anteriormente, pueden proporcionar los datos por petición explícita o periódicamente mediante subscripción. Hay tantos DAs como áreas problema.
- *Management Agents.* Se han definido dos clases de agentes de gestión:

SERVICIO	PROVIDER ROLE	DEPENDING ROLES	TIPO DE AGENTE
Datos de tráfico	informer		DA
	notifier	subscriber	
Detección de problemas de tráfico	informer		PDA
	notifier	subscriber	
	explainer		
Estado general de área problema	informer		PDA
Propuesta de configuración de PSMV	advisor	informer	CA
	explainer		
Resolución de conflictos en PSMV	negotiator		CA
	explainer		
Visualización de mensajes en PSMV	requestee		AIA

Tabla 6.1: Servicios para gestión de tráfico

- *Problem Detection Agents (PDA).* Hay tantos como áreas problema y proporcionan el servicio de *Detección de problemas de tráfico* en su área de responsabilidad, analizando el estado del tráfico y detectando problemas. Si se identifica una situación problemática, solicita al correspondiente agente de control (CA) que proponga acciones de control para resolverla (servicio *Propuesta de configuración de PSMV*). Además estos agentes PDA también proporcionan el servicio *Estado general de área problema* cuando es requerido.
- *Control Agents (CA).* Cada CA es responsable de resolver/minimizar los problemas detectados en una o varias áreas problemas, generando propuestas de señalización, proporcionando el servicio *Propuesta de configuración de PSMV*. Para realizar esta tarea pueden necesitar obtener información de áreas circundantes a la congestión, por lo que buscan e invocan el servicio de *Estado general de área problema* y en ocasiones el de *Detección de problemas de tráfico*, ambos proporcionados por agentes PDA. Una propuesta de control consiste en una colección de mensajes para mostrar en

paneles PSMV con avisos o recomendaciones de rutas alternativas a los conductores que se aproximan a una congestión. Cuando se han dado varias congestiones y dos o más CAs compiten por el uso del mismo panel PSMV, los CAs involucrados deben alcanzar un acuerdo en una propuesta conjunta consistente (servicio *Resolución de conflictos en PSMV*).

La relación entre PDAs y CAs no es uno a uno. Un mismo CA puede ser el responsable de generar propuestas de señalización para varios PDAs. Además, la solución del problema planteado por un PDA puede requerir conocer el estado de otras áreas que vierten tráfico sobre la zona congestionada, y cuyos problemas no son resueltos por el mismo CA. Por las características de la red de tráfico de Bilbao, se diseñaron un total de 12 PDAs y 5 CAs, definiendo completamente sus bases de conocimiento. Los CAs son los siguientes:

- Atena: Genera propuestas de señalización para las áreas 2 y 6. Solicita información del área 12.
- Briseide: Genera propuestas de señalización para las áreas 1, 4 y 8
- Cassandra: Genera propuestas de señalización para las áreas 5 y 10. Solicita información del área 1.
- Demetra: Genera propuestas de señalización para el área 11. Solicita información de las áreas 2 y 7.
- Elena: Genera propuestas de señalización para las áreas 3, 9 y 12. Solicita información del área 7.

- *User Interface Agent (UIA)*. Presenta la información generada por el sistema a los operadores. Esta información incluye los problemas detectados (se subscribe al servicio *Detección de problemas de tráfico*) y los mensajes propuestos para los PSMV que tienen alguna influencia sobre el área congestionada (servicio *Propuesta de configuración de PSMV*).
- Un *Agente de Implementación de Acciones (AIA)* ejecuta las decisiones adoptadas por los operadores (servicio *Visualización de mensajes*

en PSMV): una vez que el operador de tráfico acepta la propuesta de control, el AIA muestra los correspondientes mensajes en los PSMV. Recuérdese que es un sistema de ayuda a la decisión, cuya responsabilidad final recae en el operador.

El resumen de la asociación de servicios a agentes se incluye en la última columna de la tabla 6.1.

La figura 6.5 muestra la parte de la arquitectura relacionada con los agentes de gestión (se completa con un DA por área, un AIA y un UIA). Las flechas continuas entre los PDAs y los CAs identifican para cada PDA cuál es el CA al que notificar los problemas, mientras que las discontinuas identifican a qué PDAs los CAs sólo solicitan el estado general del tráfico.

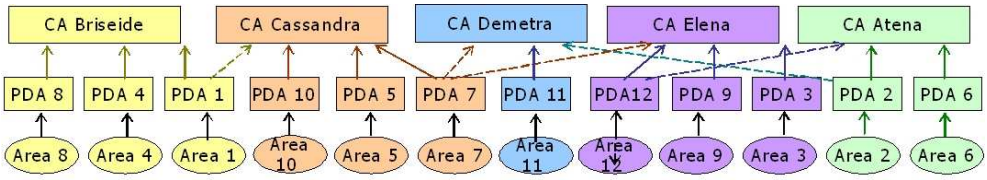


Figura 6.5: Agentes de gestión (PDAs y CAs) para la gestión de tráfico

En la arquitectura propuesta, todos los agentes inicialmente registran las descripciones de los servicios que proporcionan. Cuando un agente necesita un servicio de otro (por ejemplo, un CA necesita información sobre el estado de un área problema), consulta el directorio (DF) buscando qué agente es capaz de proporcionar ese servicio. Se puede mejorar la eficiencia utilizando mecanismos de caché (p.e. hasta que el agente no esté disponible) para evitar tener que consultar constantemente el directorio para cada interacción. De esta manera, el sistema puede sufrir modificaciones en su arquitectura sin apenas verse afectado. Por ejemplo, si por un determinado motivo (p.e. complejidad) se divide un área en varias, implica que ahora surgen nuevos agentes (PDAs y DAs). Estos cambios no deben afectar al resto de la organización, puesto que el directorio almacena los servicios que proporciona cada uno y es el lugar donde buscarlos.

6.1.3. Sistema para la gestión de flotas de autobuses

6.1.3.1. Descripción del Dominio

En muchas de las grandes ciudades, los autobuses están equipados con dispositivos de radio y GPS que proporcionan a los operadores de un centro de control información actualizada sobre la localización de los autobuses, y que se pueden utilizar para comunicarse con los conductores. Una tarea típica de un operador del centro de control es detectar incidentes (autobuses que van retrasados o adelantados respecto a su horario, averías, etc.) comparando la planificación de horarios de un autobús con sus datos de localización actuales, y enviar órdenes o acciones de control a los conductores de los autobuses (incrementar/reducir la velocidad, cambiar de regulación por horario a regulación por frecuencia, ...) para mantener o restablecer una calidad de servicio aceptable.

En Málaga, los operadores del centro de control del consorcio local de transportes (Empresa Malagueña de Transporte, EMT) usan un Sistema de Ayuda a la Explotación (SAE) que muestra información relativa al estado de los autobuses respecto a los servicios planificados. Hay un inspector por cada línea que toma decisiones para ajustar servicios en circunstancias imprevistas, y supervisores que se encargan de tomar decisiones más amplias y complejas para varias líneas. Se construyó un prototipo de SAD que extiende la funcionalidad del SAE que dialoga con los operadores de la EMT respecto a las causas de problemas y las mejores acciones a tomar.

6.1.3.2. Modelado de Interacciones

Tipos de interacción típicos en la gestión de líneas de autobuses son:

- *notificaciones*: de llegadas de autobuses a paradas, incidentes de autobuses, problemas en algún autobús o línea, ...
- *subscripción*: para ser notificado de eventos

- *intercambio de información*: problemas en alguna o varias líneas, sus causas, estado de líneas, etc.
- *asesoramiento*: propuestas de acciones de control
- *explicación*: de problemas, causas, recomendaciones, ...

Obsérvese que muchos de los tipos de interacción (y sus roles) usados en la gestión de tráfico pueden ser reutilizados también para este dominio (asesoramiento, notificación, intercambio de información, etc.).

La figura 6.6 muestra un ejemplo de tipo de interacción de mediación (*brokering*) para la transferencia de un autobús de una línea a otra. Se trata de una clase de interacción de mediación donde un intermediario guía un proceso de negociación para encontrar una línea que transfiera un autobús a otra línea para que esta última se recupere de una avería en un autobús (la EMT no dispone de autobuses de reserva).

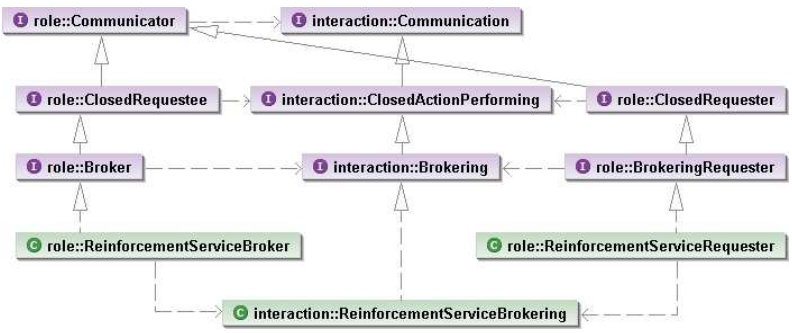


Figura 6.6: Proceso de abstracción para la interacción de mediación para la transferencia de un autobús de una línea a otra

6.1.3.3. Servicios

En el dominio de la gestión de líneas de autobuses de han identificado los siguientes servicios:

- *Llegadas de autobuses a paradas*. Servicio que proporciona información sobre llegadas de autobuses a paradas en de una línea determinada. Se puede acceder a este servicio mediante dos tipos de interacciones. La forma más intuitiva es mediante una interacción de *notificación de llegadas*, donde el servicio notifica las llegadas a los agentes que previamente se hayan *subscrito* a dicha funcionalidad, por lo que es necesario que los clientes sean capaces de jugar el rol de *subscriber*. La otra posibilidad es que, a petición del usuario del servicio, se proporcionen los datos de llegadas hasta ese momento o desde el último instante mediante una interacción de *intercambio de información*.

- *Averías de autobuses*. Con una estructura similar al servicio anterior, en este caso informa de averías de autobuses, tanto mediante interacción de *notificación* como *intercambio de información*.

- *Detección de problemas en líneas de autobuses*. Es un servicio que es capaz de detectar problemas que están ocurriendo en una línea (tanto a nivel global de la línea como individual de un autobús), comparando los horarios previstos con los datos de llegadas de autobuses a paradas (obtenida del servicio *Llegadas de autobuses a paradas*) o averías conocidas (servicio *Averías de autobuses*). A cada problema detectado se le asocia un nivel de gravedad en función de los retrasos producidos y la estimación de pasajeros afectados. Igual que ocurría en el servicio equivalente en el dominio de tráfico (*Detección de problemas de tráfico*), este servicio puede proporcionarse mediante interacciones de *intercambio de información*, a petición expresa del cliente, o mediante una *notificación* cada vez que se detecta el problema, de forma que se procese lo antes posible. En este último caso, de nuevo, se requiere que el cliente pueda participar en la interacción de *subscripción*. Además, el usuario, una vez recibida una notificación o información sobre un problema en la línea, puede solicitar una *explicación* de los motivos que han llevado a esa conclusión.

- *Propuesta de acción de control* Es capaz de recomendar una propuesta de acciones de control a enviar a los conductores para minimizar ciertos problemas. Para poder componer una propuesta, en esa interacción

de *asesoramiento* puede requerir al solicitante un *intercambio de información*. Además, una vez realizada una propuesta también puede ser requerido para que la *explique*.

- *Coordinación de servicio de refuerzo*. Es un servicio que realiza una labor de mediación (interacción *brokering*) para conseguir un autobús de refuerzo para una línea que lo necesita. La mediación consiste en un proceso de consultas a líneas sobre su disponibilidad a ceder uno de sus autobuses (el detalle del algoritmo no es relevante aquí y puede ser obtenido en [9]).

- *Autorización de servicio de refuerzo*. Este servicio realiza una consulta al operador sobre la autorización para considerar como acción de control conseguir un autobús de refuerzo para una línea, que implica moverlo desde otra. En el transcurso de la interacción de *autorización* es posible que el usuario solicite algún tipo de información en base a la que permitir o no esa acción.

- *Aceptación de transferencia de autobús*. Es un servicio en el que se *intercambia información* acerca de la voluntad para ceder un autobús de una línea. Este servicio es utilizado durante el proceso de mediación para conseguir un autobús de refuerzo.

- *Envío de órdenes a conductores*. Este servicio es el encargado de llevar a cabo las acciones de control, consistentes en enviar los mensajes correspondientes a los conductores de los vehículos. Participa en interacciones de *ejecución de acciones* iniciadas por el cliente, por lo que juega el rol *requestee*.

La tabla 6.2 muestra la lista de servicios para la gestión de líneas de autobuses, junto con su descripción en términos de roles.

SERVICIO	PROVIDER ROLE	DEPENDING ROLES	TIPO DE AGENTE
Llegadas de autobuses a paradas	informer		DA
	notifier	subscriber	
Averías de autobuses	informer		DA
	notifier	subscriber	
Detección de problemas en líneas de autobuses	informer		LMA
	notifier	subscriber	
	explainer		
Propuesta de acción de control	advisor	informer	LMA
	explainer		
Coordinación de servicio de refuerzo	broker	informer	CF
Autorización de servicio de refuerzo	authorizer	informer	UIA
Aceptación transferencia de autobús	informer		LMA
Envío de órdenes a conductores	requestee		AIA

Tabla 6.2: Servicios para gestión de líneas de autobuses

6.1.3.4. Arquitectura

En esta sección se describe la arquitectura de agentes concreta utilizada para la gestión de líneas de autobuses, indicando los servicios que proporciona cada tipo de agente.

- *Data Agents (DA).* Existe un DA por cada línea y son encargados de proporcionar los servicios de *Llegadas de autobuses a paradas* y *Averías de autobuses* para la línea de la que se encargan. Esta información la pueden proporcionar bajo demanda o cuando sucede.
- *Action Implementation Agents (AIA).* Simplemente envían las acciones de control a los autobuses implementando el servicio *Envío de órdenes a conductores*
- *Management agents.* Se han definido dos clases de agentes de gestión:
  - *Line Management Agents (LMA).* Hay uno por cada línea de autobuses y está encargado tanto de la detección de problemas (servicio *Detección de problemas en líneas de autobuses*) como de la propuesta de acciones para su minimización (servicio *Propuesta de acción de control*). Se subscribe a los servicios de *Llegadas de autobuses a paradas* y *Averías de autobuses* de la línea que gestiona para ser notificado cada vez que uno de esos eventos ocurre, con el fin de poder detectar los problemas existentes. En determinadas situaciones, la acción de control puede consistir en solicitar un autobús de refuerzo de otra línea, para lo cual en primer lugar debe pedir autorización al operador para intentar esa acción (invoca el servicio *Autorización de servicio de refuerzo*) y, en caso afirmativo, utiliza el servicio *Coordinación de servicio de refuerzo* para concretar, en su caso, el autobús que se traspasará a esa línea. Además, también proporciona el servicio *Aceptación de transferencia de autobús* mediante el que se informa de la voluntad de ceder o no un autobús durante un proceso de negociación.

- *Coordination Facilitator (CF).* Soporta la coordinación entre líneas. Más precisamente, actúa como mediador en el proceso de negociación entre los LMAs para obtener un servicio de refuerzo, esto es, proporciona el servicio *Coordinación de servicio de refuerzo.*

- *User Interface Agents (UIA).* Muestran la información seleccionadas a los operadores del centro de control. Muestran el estado de la línea, informan sobre incidentes o problemas, y notifican recomendaciones de control o propuestas de los LMAs. Para este propósito, un UIA se subscribe a los servicios *Llegadas de autobuses a paradas* (para mostrar el estado de cada autobús), *Averías de autobuses* y *Detección de problemas en líneas de autobuses.* Además, solicita el servicio de *Propuesta de acción de control* y proporciona el servicio de *Autorización de servicio de refuerzo.*

Igual que en el dominio de tráfico, todos los agentes inicialmente registran las descripciones de los servicios que proporcionan y utilizan el directorio para averiguar con qué agentes deben comunicarse (subscribirse, requerir información, etc).

6.1.4. Discusión

Las dos aplicaciones de transporte presentadas en esta sección demostraron la aplicabilidad y reusabilidad de la arquitectura multi-agente abstracta, que es la base en el diseño de ambos demostradores. Esa arquitectura permitió un diseño e implementación poco acoplado de los diferentes módulos (agentes) que componen el SAD. Si un agente no funciona o hay problemas de comunicación sólo el área de influencia de ese agente se ve afectada, pero el resto del sistema puede continuar funcionando correctamente. Por ejemplo, si hay problemas con la notificación de llegadas de autobuses en la línea 7, sólo las áreas circundantes a la línea 7 se verían afectadas de esa irregularidad.

Sin embargo, la adaptabilidad dinámica no estaba completamente resuelta en aquellos prototipos. Los agentes estaban dotados de *modelos de conocidos* que

incluían el conocimiento sobre los agentes con los que tenían que comunicarse para llevar adelante sus tareas. Por ejemplo, en la aplicación de gestión de tráfico, el agente de control *CA Atena* sabe que los agentes *PDA 2* y *PDA 6* informan de los problemas de las áreas de las que es responsable y que debe consultar además al *PDA 12* sobre el estado general de esa área; igual que cada *PDA* sabe cuál es el *CA* encargado de proponer acciones de control para minimizar los problemas que detecta.

En este capítulo se ha aplicado un diseño orientado a servicios para complementar la arquitectura previa y superar los inconvenientes indicados. Para ello se han analizado algunos diálogos típicos, extraído los roles y tipos de interacciones en las que se participa, se han identificado y descrito los servicios en términos de la información organizacional, que es la propuesta de esta tesis.

Según este enfoque, los agentes usan el directorio (*DF*) para buscar los agentes capaces de proporcionar la funcionalidad requerida, evitando así el tener que incluir esa información en sus bases de conocimiento y permitiendo una fácil adaptación frente a cambios en la organización de agentes. Utilizando el ejemplo anterior, el agente de control *CA Atena*, cuando necesita información sobre el estado de un área problema, puede usar el servicio de directorio para averiguar qué agentes son capaces de participar en interacciones de intercambio de información sobre una área concreta de la ciudad.

Gracias al uso de este nuevo enfoque surgen nuevas posibilidades. Por ejemplo, si hay varios proveedores de un cierto servicio, el solicitante puede seleccionar el más adecuado a sus necesidades basado en experiencias pasadas, modelos de reputación y confianza, etc. Además, se podrían tener en cuenta servicios adicionales si se encuentran disponibles. Por ejemplo, servicios de notificación sobre eventos masificados (partidos de fútbol, conciertos, huelgas, accidentes, etc.) pueden ser de gran importancia para regular el tráfico en áreas cercanas. Obsérvese que los servicios se pueden acceder desde diferentes dominios (si se comparte el directorio). Por ejemplo, para la gestión de las líneas de autobuses puede ser de gran utilidad conocer el estado del tráfico. Si se busca un servicio que proporcione esa información podría encontrarse el implementado en el

dominio de gestión de tráfico.

La figura 6.7 muestra una vista parcial de la taxonomía de tipos de interacción construida a partir del análisis de los dos dominios de gestión de transporte presentados aquí. Por motivos de legibilidad sólo se han incluido algunas de las interacciones.

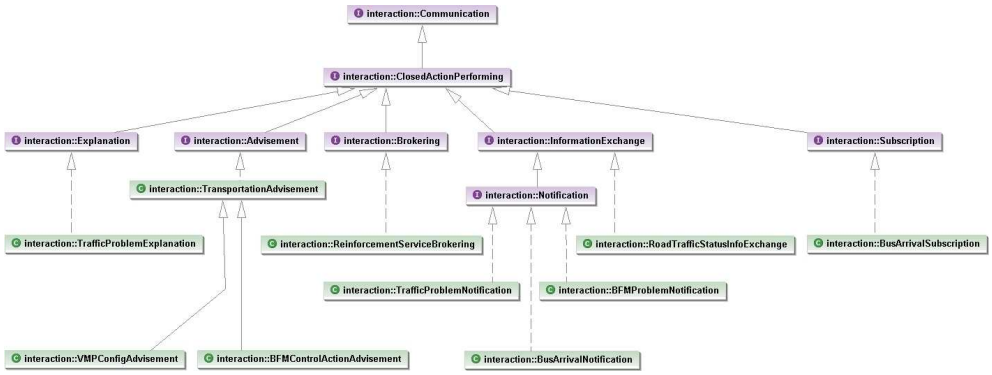


Figura 6.7: Ontología parcial de tipos de interacción en el dominio de transporte

Se puede comprobar del análisis de las interacciones realizadas en esta sección como de las omitidas que gran parte de la ontología (sobre todo la parte genérica) se reutiliza de una aplicación a otra.

6.2. Coordinación de servicios en entornos móviles. Aplicación en la asistencia en emergencias médicas

Las propuestas realizadas en esta tesis han sido desarrolladas y aplicadas en el proyecto CASCOM[3] (Context-Aware Business Application Service Coordination in Mobile Computing Environments), subvencionado por la Unión Europea, bajo el FP6.

[3]http://www.ist-cascom.org

El principal objetivo del proyecto CASCOM fue la implementación de una infraestructura para la coordinación de aplicaciones basadas en servicios Web semánticos a través de redes fijas y móviles. La infraestructura construida ha sido validada mediante la implementación de una aplicación real en el dominio de las emergencias médicas. Una descripción detallada de CASCOM puede encontrarse en [56].

A continuación se describe la arquitectura abstracta de CASCOM. Posteriormente se identifican los elementos de la arquitectura que incluyen los componentes desarrollados en el marco de esta tesis y su integración. Finalmente se describe la aplicación utilizada para validar la plataforma.

6.2.1. La arquitectura de CASCOM

La arquitectura abstracta de CASCOM [18, 19] sigue un enfoque basado en capas (figura 6.8) que combina aspectos a nivel de red, como redes peer-to-peer y móviles, con tecnologías como servicios Web semánticos y agentes inteligentes. Los cuatro componentes principales de esta arquitectura actúan como mediadores entre las aplicaciones y las redes subyacentes.

La capa de red (*Networking Layer*) se encarga de gestionar las heterogeneidades de las redes, proporcionando una infraestructura genérica P2P. Ofrece un canal de comunicación eficiente y seguro basado en FIPA-HTTP [49], y un entorno de ejecución para dispositivos móviles con escasez de recursos. Además, proporciona servicios de directorios (DS) distribuidos y federados, un mecanismo de bajo nivel de búsqueda de servicios anunciados por los proveedores.

La capa de coordinación de servicios (*Service Coordination Layer*) está situada entre la capa de red (*Networking Layer*) y la capa de aplicación (*Application Layer*), y usa los servicios de los subsistemas de contexto (*Context Subsystem*) y seguridad y privacidad (*Security & Privacy Subsystems*). Sus principales funcionalidades son el descubrimiento, equiparación, composición y ejecución de servicios Web semánticos.

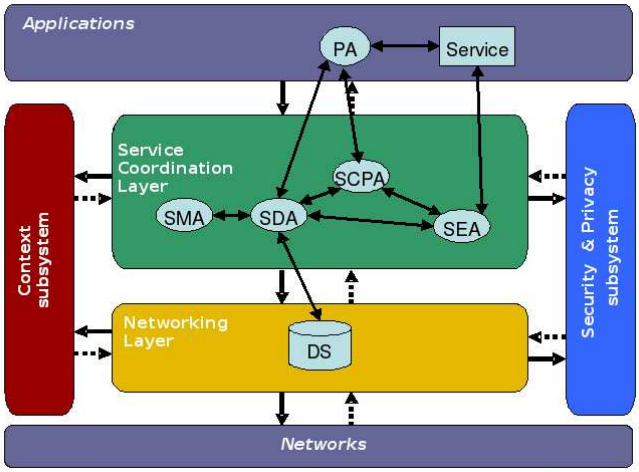


Figura 6.8: Arquitectura de CASCOM

En la arquitectura abstracta de CASCOM, la funcionalidad de descubrimiento semántico de servicios es relizada por dos tipos diferentes de agentes: *Service Discovery Agents (SDA)* y *Service Matchmaking Agents (SMA)*. Los *Service Discovery Agents (SDA)* gestionan el descubrimiento de los servicios requeridos, actuando como interfaz de agente para el servicio de directorio (*DS*). La equiparación de servicios se concibe como un encaje fino, sintáctico y semántico entre un servicio solicitado y los servicios almacenados en el *DS*. Esta funcionalidad es proporcionada por *Service Matchmaking Agents (SMA)*, que tienen en cuenta los conceptos de entrada y salida, las expresiones lógicas de precondiciones y efectos, y también propiedades no funcionales del servicio (como los roles jugados por los agentes en las interacciones).

Normalmente, los SDAs reciben especificaciones de solicitudes de servicios por parte de los usuarios a través de los agentes personales (*Personal Agents (PA)*) y adquieren información contextual relevante del subsistema de contexto. Con esa información, usan la funcionalidad de descubrimiento de servicios de la capa de red y la funcionalidad de encaje semántico de los *SMA*s, para determinar los servicios que cumplen con la solicitud recibida. Los *SDA*s entonces devuel-

ven el conjunto de descripciones/proveedores y sus correspondientes modelos de procesos y/o groundings (*OWL-S*).

Los *Service Composition and Planning Agents (SCPA)* son capaces de crear servicios compuestos que cumplan con especificaciones de servicios concretos, cuando no hay ningún servicio que encaje con una consulta del usuario. Una vez que los *SCPA*s reciben las especificaciones de los servicios requeridos, contactan con los *SDA*s para descubrir los servicios existentes en un dominio restringido al contexto actual, y planifican un servicio compuesto para la especificación recibida.

La ejecución de servicios compuestos es proporcionada por los *Service Execution Agents (SEA)*. Los SEAs coordinan la ejecución de cada servicio atómico que compone el plan, descubriendo los proveedores de servicios disponibles para cada uno de ellos si es necesario utilizando los *SDA*s.

El subsistema de contexto *Context Subsystem* es accedido tanto por la *Networking Layer* por la *Service Coordination Layer*. Su principal funcionalidad es descubrir, adquirir y almacenar información de contexto útil (por ejemplo posición geográfica, preferencias del usuario, disponibilidad de servicios, tiempos de respuesta, etc.).

El susbsistema de seguridad y privacidad (*Security and Privacy Subsystem*), también ortogonal a los de red y coordinación de servicios, es responsable de asegurar la seguridad y privacidad de información por los diferentes componentes de la infraestructura CASCOM.

6.2.2. Aportación en la arquitectura

Las propuestas realizadas en esta tesis han sido integradas dentro de la arquitectura de CASCOM en dos elementos: el *SMA* (que incluye el componente ROWLS) y el *SCPA* (filtro para composición). En esta sección se detalla esta integración.

6.2.2.1. Integración de ROWLS en el SMA

En CASCOM, los agentes *Service Matchmaking Agents (SMA)* se encargan de calcular el grado de encaje de cada uno de los servicios candidatos (extraídos de directorios) con un servicio deseado, y devolver en orden de mayor a menor encaje aquellos servicios que superan un umbral determinado.

Este agente está formado por tres bloques, cada uno de los cuales corresponde a un matchmaker diferente: OWLS-MX (un matchmaker híbrido que analiza las entradas y salidas de los servicios, como se describió en la sección 2.2.3.4), PcEM (un matchmaker que analiza las precondiciones y efectos de los servicios) y ROWLS (basado en roles). Se implementaron dos posibles configuraciones de integración de estos tres componentes:

1. *Secuencial*. Usa los tres matchmakers secuencialmente, donde cada matchmaker actúa como un prefiltro del siguiente en la secuencia. La parte izquierda de la figura 6.9 muestra esta opción, donde ROWLS reduce el número de servicios de entrada a OWLS-MX, que a su vez reduce el número de servicios de entrada al PcEM. Este orden de componentes se eligió de esta manera basándose en la complejidad computacional de los tres matchmakers.

2. *Agregada*. Ejecuta los tres matchmakers con el mismo conjunto de servicios y después combinar los grados de equiparación obtenidos mediante una función de agregación (figura 6.9 derecha).

Cada uno de los tres matchmakers tienen ciertos parámetros de configuración. En el momento que el SMA es invocado por otro agente, se puede seleccionar la configuración de los diferentes módulos que componen el SMA mediante un parámetro especial que se incluye en el mensaje de solicitud.

Configuración Secuencial. En la configuración secuencial en primer lugar se ejecuta el matchmaker basado en roles, que asigna a cada servicio un valor

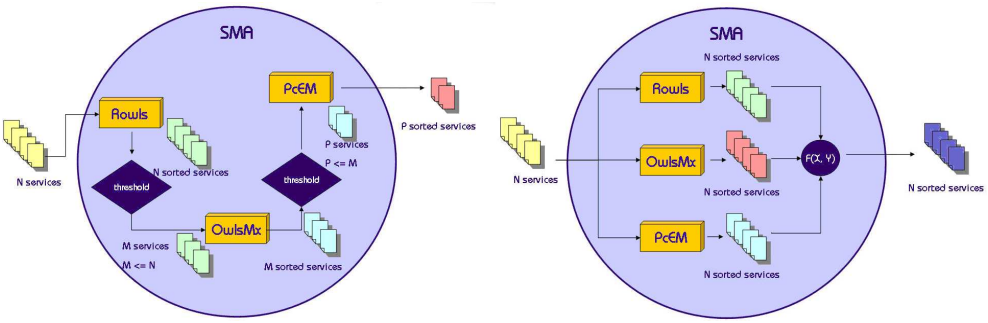


Figura 6.9: Arquitectura interna del SMA: Secuencial (izquierda) o Agregada (derecha)

real entre 0 y 1. Después, un conjunto reducido de estos servicios se pasan al OWLS-MX como entada. Este conjunto reducido puede estar compuesto de:

- Todos los servicios que tienen un grado de encaje basado en roles superior a un umbral
- Los primeros K servicios del conjunto ordenado, donde K es un valor fijo
- Los primeros M servicios del conjunto ordenado, donde M es un porcentaje del tamaño original del conjunto.

OWLS-MX ejecuta el encaje con este conjunto reducido de servicios y devuelve una lista ordenada dependiendo de la parametrización. Hay cinco posibles instanciaciones de OWLS-MX, $M0$ a $M4$, donde $M0$ es el encaje puramente lógico, mientras que las otras usan métricas de similitud sintáctica ("Loss-of-information", "Extended Jacquard", "Cosine" o "Jensen-Shannon divergence based"). Los parámetros configurables son el mínimo grado de encaje (*exact*, *plugin*, *subsumes*, *subsumed-by* o *nearest-neighbour*), el tipo de matchmaker ($M0 - M4$) y, en caso de usar métricas de similitud sintáctica, el umbral de dicha similitud (valor real entre 0 y 1).

Finalmente, la salida de OWLS-MX es pasada al matchmaker PcEM como entrada. Este matchmaker sólo devuelve un valor booleano indicando si el

servicio cumple o no las precondiciones y efectos, por tanto esos servicios serán los finalmente devueltos.

El orden elegido entre los tres matchmakers es debido a que la complejidad computacional de ROWLS es menor que la de OWLS-MX, que a su vez es menor que la de PcEM. Por eso, el matchmaker que trabajará con más servicios será el más rápido.

Teniendo en cuenta lo anterior, los parámetros necesarios en esta configuración del SMA son

- El parámetro del filtrado del matchmaker basado en roles (*threshold* o K o M).
- El tipo de matchmaker OWLS-MX ($M0$, $M1$, $M2$, $M3$, $M4$), el mínimo grado de encaje y el umbral de similaridad.

Configuración Agregada. La otra posible configuración del SMA es la ejecución de los tres matchmakers de forma independiente sobre el mismo conjunto de servicios, y la utilización de funciones de agregación para combinar los resultados.

Para esto, es necesario que cada matchmaker asigne un valor numérico a cada servicio. Sin embargo, sólo ROWLS devuelve un número real entre 0 y 1 como grado de encaje, mientras que el PcEM puede devolver 0 o 1 dependiendo de si el resultado es *cierto* o *falso*, respectivamente. Por otro lado, OWLS-MX devuelve una categoría, por lo que se asocia un valor a cada una, en concreto: *exact*=1, *plugin*=0.7, *subsumes*=0.5, *subsumed-by*=0.3, *nearest-neighbour*=0.1, *fail*=0).

Una vez asignado un grado de encaje para cada servicio por parte de cada matchmaker, es posible combinar los tres valores con una función de agregación (p.e. *mín*, *max*, ...) y ordenar el conjunto original de servicios en base a ese valor resultante.

En resumen, los parámetros necesarios en esta configuración son:

SMA	MODE	ROWLS			OWLS-MX			
Parameter		Threshold	k-filter	%-filter	Type	Min DOM	Threshold	Aggregation
		0..1	Integer	0..100	{M0..M4}	(Exact, plugin,...)	0..1	function
DEFAULT	Sequential	---	---	60	M0	Exact	---	---
Exact	Sequential	1	---	---	M0	Exact	---	---
plugin	Sequential	0,7	---	---	M1	plugin	---	---
subsumes	Sequential	0,5	---	---	M2	subsumes	---	---
subsumed-by	Sequential	0,3	---	---	M3	subsumed-by	0,5	---
nearest-neighbour 1	Sequential	---	20	---	M4	nearest-neighbour	0,2	---
nearest-neighbour 2	Sequential	0,3	---	---	M2	nearest-neighbour	0,5	---
aggregation 1	Aggregation	---	---	---	M1	nearest-neighbour	0,7	Max(0,x+y-1)
aggregation 2	Aggregation	---	---	---	M2	nearest-neighbour	0,5	prod
aggregation 3	Aggregation	---	---	---	M3	nearest-neighbour	0,3	min
aggregation 4	Aggregation	---	---	---	M4	nearest-neighbour	0,5	0.5*OWLS-MX + + 0.3*ROWLS + + 0.2*PcEM
owls-mx	---	---	---	---	M4	nearest-neighbour	0,5	---
rowls+owlsmx	Sequential	0,4	---	---	M4	nearest-neighbour	0,5	---
owls-mx+pcem	Sequential	---	---	---	M4	nearest-neighbour	0,5	---

Figura 6.10: SMA predefined configurations

- El tipo de matchmaker OWLS-MX ($M0$, $M1$, $M2$, $M3$, $M4$), el mínimo grado de encaje y el umbral de similaridad.
- La función de agregación.

Configuraciones predefinidas. Para facilitar la selección de parámetros de los diferentes componentes del SMA, se han definido varias configuraciones que puede seleccionarse cuando se invoca la acción *getMatchingServices* del agente (mediante el protocolo *FIPA-Request*). Estas configuraciones se muestran en la figura 6.10. En caso de que no se proporcione el parámetro se selecciona una configuración por defecto.

6.2.2.2. Integración del filtro en el SCPA

Los agentes SCPA son capaces de crear servicios compuestos que cumplen las especificaciones de servicios concretos solicitados. Una vez que el SCPA recibe una especificación de servicio de los Agentes Personales (PA), consultan a los Service Discovery Agents (SDAs) para buscar servicios existentes en un dominio determinado, restringido al contexto actual, y planifica un servicio compuesto que cumple con las especificaciones. Los planes generados se envían

a Agentes de Ejecución de Servicios (SEA), que gestionan la ejecución de servicios compuestos. Además, el plan y la solicitud original se pasan al filtro para actualizar la información histórica sobre planes.

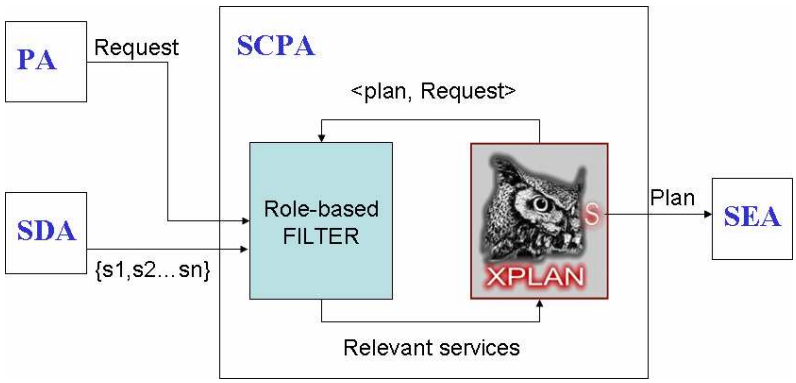


Figura 6.11: Arquitectura del SCPA

En CASCOM, se han implementado dos tipos de SCPA que se diferencian en el método de planificación utilizado.

El primer tipo utiliza como planificador OWLS-XPlan [68] (ver sección 2.2.4), un planificador que usa técnicas heurísticas de IA para composición de servicios descritos en OWL-S (basado en el planificador FF-planner [58]). En este SCPA se ha incluido el filtro basado en roles para reducir el conjunto de servicios utilizados por el planificador, con el objetivo de incrementar la eficiencia global del planificador. La figura 6.11 muestra la arquitectura y contexto del SCPA. Este es el contexto típico para utilizar el filtro, esto es, en combinación con un planificador, en este caso OWLS-XPlan. Obsérvese que el plan compuesto, además de enviarse al *SEA* también se comunica al filtro para que actualice la relevancia de los servicios involucrados para la consulta realizada.

El otro tipo de SCPA, denominado MetaComp, utiliza el planificador SAPA [33] pero no utiliza el filtro ya que sigue otra estrategia distinta para conseguir los servicios candidatos.

6.2.3. Aplicación en la gestión de emergencias médicas

La plataforma CASCOM ha sido validada mediante una aplicación del mundo real en el dominio de la asistencia en emergencias médicas. El escenario concreto elegido, similar al descrito en la sección 3.1, es el mostrado en la figura 6.12.

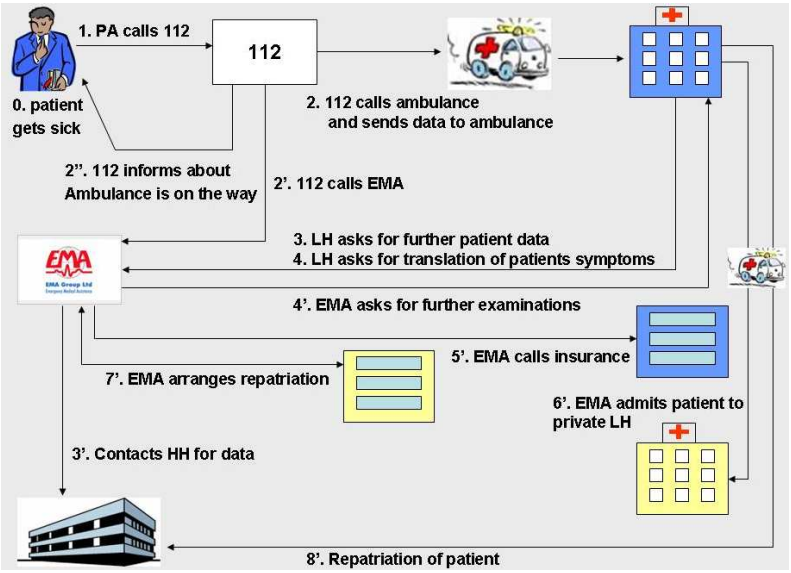


Figura 6.12: Escenario de aplicación usado en las pruebas.

En el escenario se asume que un turista finlandés está de vacaciones en Austria cuando sufre un dolor en el pecho. Antes de salir de Austria, instaló el software CASCOM necesario en su teléfono móvil programable y lo activa para iniciar una llamada de emergencia. Tras la iniciación tiene que contestar algunas preguntas sobre sus síntomas que aparecen en la interfaz gráfica de su teléfono (figura 6.13). A continuación el agente de usuario del paciente establece la llamada de emergencia, y el agente 112 (centro de emergencias) es contactado (1). Este agente contacta con la ambulancia más cercana e informa sobre el caso médico del paciente (2). Esta información básicamente contiene los datos recogidos del cuestionario inicial, datos históricos del paciente (por ejemplo, electrocardiogramas, rayos X), algunos datos personales, además de la posi-

ción del paciente determinada por un servicio de localización (figura 6.15). El servicio de localización permite dirigir la ambulancia directamente al paciente. Con toda esta información los médicos de la ambulancia pueden preparar el tratamiento del paciente. Además, EMA (una organización finlandesa de asistencia en emergencias médicas) es también contactada por el agente 112 (2'). Entretanto (2") el paciente recibe un mensaje de que la ambulancia está de camino (figura 6.14).

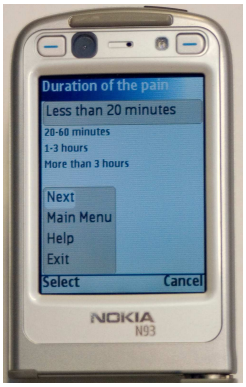


Figura 6.13: Introducción de síntomas

Figura 6.14: Ambulancia en camino

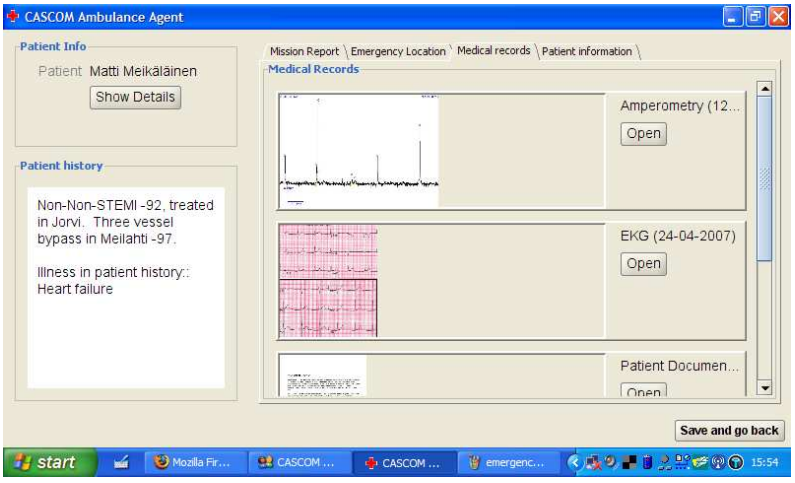


Figura 6.15: Información del paciente

Después de una primera atención, los médicos de la ambulancia deciden el

hospital local (LH) para el paciente y lo llevan allí. Los médicos de urgencia en el hospital también reciben la información del paciente para preparar su tratamiento.

Ya en el hospital, el LH necesita datos adicionales del paciente y se los solicita a EMA (3), que los consigue del hospital de residencia del paciente (HH, Home Hospital) (3') e inmediatamente se los proporciona al LH, ya que puede ahorrar un tiempo importante en esta situación crítica. A continuación, el hospital local pide a EMA una traducción de las pruebas históricas del paciente (4). Además, EMA monitoriza la calidad del tratamiento del paciente. Si no es de la calidad adecuada, solicita al LH realizar pruebas adicionales (4'). Cuando la situación del paciente requiere una repatriación a su hospital de residencia (HH), EMA se encarga de consultar con la compañía de seguros sobre el cubrimiento del coste asociado (5'), y organiza el transporte a un hospital privado (6') o la transferencia a su residencia habitual (7', 8') e informa al hospital de la llegada del paciente. Además el hospital en su ciudad de residencia puede conseguir los datos del paciente via servicios CASCOM y usarlos para preparar la continuación del tratamiento del paciente. Este escenario demanda actividades flexibles de planificación y replanificación que son transparentes al usuario.

En base a este escenario se realizaron tanto pruebas de laboratorio como en entornos reales. En concreto, se han realizado dos pruebas de usabilidad, una en el laboratorio de usabilidad de TeliaSonera[4] en Helsinki (Finlandia), y otra enfocada en el agente personal (PA), realizado en Basel (Suiza) con diferentes usuarios ajenos al proyecto.

Además se han realizado pruebas en un entorno real de hospital y personal médico, realizadas en Innsbruck (Austria). Fueron apoyadas fuertemente tanto por médicos de emergencias como por personal técnico del Hospital Universitario en Innsbruck.

Tuvieron lugar en el departamento de urgencias y un vehículo de emergencias (figura 6.16) en TILAK (Tiroler Krankenanstalten GmbH, la organización

[4]miembro del consorcio del proyecto

de los hospitales públicos del Tyrol). Participaron dos médicos de reconocido prestigio internacional. TILAK y UMIT (University for Health Sciences, Medical Informatics and Technology) ofrecieron su ayuda profesional para la realización de las pruebas.

Los resultados de las pruebas fueron muy satisfactorios, en especial los médicos de urgencias encontraron la aplicación muy útil y comentaron que la usarían tanto en su vida profesional como en su vida privada si disponen de la aplicación.

Figura 6.16: Ambulancia dotada de dispositivos móviles con software CASCOM

Capítulo 7

Conclusiones

En esta tesis doctoral se han logrado los objetivos planteados en la sección 1.1. A continuación se resumen y comentan las principales aportaciones. Posteriormente (sección 7.1) se enumeran las publicaciones que han surgido como consecuencia de la realización de este trabajo. Finalmente se proponen algunas líneas de investigación futuras.

El primero de los objetivos trata de la descripción de servicios proporcionados por agentes y es desarrollado en el capítulo 3. Un estudio profundo del estado actual en la materia relacionada revela que, por un lado, los conceptos organizacionales como roles y tipos de interacciones son fundamentales en la construcción de sistemas multi-agente, como lo apoya el hecho de que prácticamente todas las metodologías conceden gran importancia a ese tipo de modelos. Por otra parte, los enfoques actuales en el campo de la descripción de servicios (normalmente servicios Web semánticos) se centran de momento en las entradas y salidas de los servicios, y en algunos casos en otra información como precondiciones, efectos o categorías. Sin embargo, la información organizacional, tan importante en los modelos de diseño multi-agente, está bastante lejos de ser tenida en cuenta para describir el servicio en su contexto organizativo. En esta tesis se propone reducir esta separación incluyendo en las descripciones la información sobre el contexto organizacional del servicio en

términos de los roles que es capaz de jugar cuando participa en diferentes tipos de interacciones. Se ha utilizado y extendido parte del modelo RICA [105]. La propuesta aquí realizada ha de entenderse como una extensión de los enfoques actuales, es decir, no sustituye aquellos sino que los extiende con información adicional, lo cual permite aumentar la expresividad de la descripción, que puede redundar en una mejora de los métodos de búsqueda semántica de servicios. En particular, se propone añadir a las descripciones de servicios dos tipos de información: (i) los roles que es capaz de jugar el proveedor del servicio, y (ii) roles que debe saber jugar el solicitante. Asimismo, en las descripciones de los servicios buscados se incluye información complementaria a la anterior: (iii) roles deseados y (iv) roles que el solicitante sabe jugar. Los elementos (ii) y (iii) pueden expresarse con determinadas fórmulas lógicas (forma normal disyuntiva) con el fin de facilitar la expresividad aunque manteniendo la sencillez computacional. Se ha propuesto la inclusión de esta información en el lenguaje de descripción de servicios OWL-S.

En el capítulo 4 se propone un algoritmo de equiparación de servicios basado en la información organizacional incluida en sus descripciones. El método propuesto combina varios enfoques actuales sobre la equiparación de servicios semánticos. En primer lugar se tiene en cuenta la relación lógica de subsunción entre concetos (siguiendo la propuesta de Paolucci [92]) para establecer el grado de encaje (*exact, plugin, subsumes, fail*), que es refinado con la distancia entre los conceptos en la ontología (Rada [96]) para obtener un valor numérico en el intervalo [0,1]. Se obtiene, por tanto, un nivel de detalle mayor (un valor numérico) para expresar el grado de encaje entre dos servicios que la propuesta de Paolucci (sólo cuatro grados de encaje) y otras actuales, lo que facilita la ordenación de servicios a la hora de seleccionar los más similares. Se ha implementado el algoritmo propuesto (componente ROWLS) y se han realizado experimentos integrado con el matchmaker de propósito general OWLS-MX [67], uno de los más conocidos en la actualidad (recuérdese que aquí se ha propuesto una extensión a los enfoques actuales). Puesto que no existe en la actualidad una colección adecuada de descripciones de servicios Web semánticos hubo que extender la incluida con OWLS-MX (OWLS-TCv2, quizá la más

completa en la actualidad) para incluir la información organizacional. Los resultados de las pruebas realizadas confirmaron que la combinación de ROWLS con OWLS-MX mejora tanto la eficiencia como la efectividad de OWLS-MX de forma aislada. Además, también se comprobó el gran peso que tiene hasta ahora el tiempo de carga de los servicios (parsing de los ficheros) por lo que se recomienda integrar, si es posible, la equiparación en el directorio para minimizar el número de cargas de servicios (manteniendo los servicios ya parseados en el directorio).

La tercera aportación se describe en el capítulo 5. Cuando no se encuentra disponible ningún servicio para una consulta concreta, se puede utilizar una funcionalidad de planificación para conseguir un servicios compuestos. En sistemas multi-agentes abiertos a gran escala, un enfoque de planificación de IA puro puede ser impracticable debido al gran número de servicios (operadores) que normalmente están registrados en el directorio. En esta tesis se ha propuesto un método para filtrar aquellos servicios que probablemente sean irrelevantes al proceso de planificación. Se ha utilizado como heurística la dimensión del plan y el número de ocurrencias de servicios en planes: cuanto más importante es un servicio, mayor es el número de planes para los que es necesario y más cortos los planes para los que es requerido. Se aproxima esta información almacenando y procesando los planes históricamente creados. Sin embargo, el número de servicios y posibles consultas puede ser demasiado grande y la repetición de una consulta particular es bastante improbable. Para superar este problema, se agrupan los servicios en clases en base a ciertas propiedades. En particular, se han propuesto dos métodos de clasificación de servicios, uno basado en las taxonomías de roles e interacciones que se derivan del modelo organizacional del SMA y otro basado en las categorías de los servicios.

Aunque al filtrar servicios puede verse afectada la completitud, esto puede no ser tan importante ya que de lo contrario es posible que el planificador no pueda proponer un plan en un tiempo razonable. Igual que ocurría en el caso del descubrimiento de servicios, debido a la inmadurez de la tecnología, no existe todavía un conjunto de casos de prueba representativo que se pueda usar pa-

ra validar nuestra propuesta. En el caso de la composición este problema se agrava. Por ello, los experimentos se han realizado con colecciones de planes generados de forma aleatoria. Además, los experimentos se han llevado a cabo de forma independiente a un planificador de servicios compuestos concreto, evitando de esta manera que posibles deficiencias del planificador influya en los resultados obtenidos. Con los experimentos realizados se ha comprobado que la tasa de planes que siguen siendo posibles es bastante aceptable incluso tras filtrar una proporción grande de servicios. Además, casi siempre permanece al menos un plan posible (salvo con porcentajes de filtrado muy altos). Aunque no demasiadas, existen otras propuestas en la literatura sobre filtros aplicados a composición de servicios (Sirin et al. [112], Synthy [1], MetaComp [12], Constantinescu [11]). La principal diferencia del enfoque propuesto en esta tesis respecto a los anteriores es el uso de información histórica para estimar la relevancia de los servicios (junto con las heurísticas) y la agrupación en clases de servicios.

Las propuestas anteriores se han llevado a la práctica mostrando su aplicabilidad en dominios reales. Por un lado, se ha utilizado el dominio de la gestión de transporte (rodado y autobuses) para mostrar cómo un sistema multiagente puede mejorarse en flexibilidad y adaptabilidad frente a cambios siguiendo un enfoque orientado a servicios, y la explotación de sus modelos organizacionales para la descripción de los servicios. Corresponde con la aplicación de la primera aportación realizada en esta tesis (capítulo 3). Por otro lado, los componentes software surgidos de la implementación de las aportaciones para el descubrimiento y composición de servicios, forman parte de una plataforma para la coordinación de servicios en entornos fijos y móviles, que ha sido utilizada en la construcción de una aplicación para la asistencia en caso de emergencias médicas.

7.1. Publicaciones derivadas de este trabajo

Durante la realización de esta tesis doctoral se han publicado los siguientes trabajos en relación con la misma (ordenados por fecha de publicación, comenzando por los más recientes).

1. L. Botelho, A. Fernández, M. Klusch, L. Pereira, T. Santos, P. Pais, and M. Vasirani. *CASCOM: Intelligent Service Coordination in the Semantic Web*, chapter Service Discovery. Birkhäuser, 2008

2. B. Blankenburg, L. Botelho, A. Fernández, M. Klusch, and S. Ossowski. *CASCOM: Intelligent Service Coordination in the Semantic Web*, chapter Service Composition. Birkhäuser, 2008

3. A. Fernández and S. Ossowski. Exploiting Organizational Models for Semantic Service Description, Matchmaking and Composition in Service-Oriented Multi-Agent Systems. *ERCIM News*, 70:53–54, 2007

4. A. Fernández and S. Ossowski. Filters for Semantic Service Composition in Service-oriented Multiagent Systems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 924–926, Honolulu, Hawai, USA, 2007

5. A. Fernández and S. Ossowski. Semantic Service Composition in Service-Oriented Multiagent Systems: A Filtering Approach. In J. Huang, R. Kowalczyk, Z. Maamar, D. L. Martin, I. Müller, S. Stoutenburg, and K. P. Sycara, editors, *Service-Oriented Computing: Agents, Semantics, and Engineering, AAMAS 2007 International Workshop (SOCASE)*, volume 4504 of *Lecture Notes in Computer Science*, pages 78–91. Springer, 2007

6. A. Fernández, M. Vasirani, C. Cáceres, and S. Ossowski. A Role-Based Support Mechanism for Service Description and Discovery. In J. Huang, R. Kowalczyk, Z. Maamar, D. L. Martin, I. Müller, S. Stoutenburg, and K. P. Sycara, editors, *Service-Oriented Computing: Agents, Semantics,*

and Engineering, AAMAS 2007 International Workshop (SOCASE), volume 4504 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2007

7. A. Fernández and S. Ossowski. Role-based Discovery and Coordination of Semantic Transportation Services. In *IEEE Intelligent Transportation Systems Conference*, pages 100–106, Seattle, Washington, USA, 2007

8. A. Fernández, M. Vasirani, C. Cáceres, and S. Ossowski. Role-based Service Description and Discovery. In *Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE)*, Hakodate, Japan, 2006

9. C. Cáceres, A. Fernández, S. Ossowski, and M. Vasirani. Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach. *IEEE Intelligent Systems*, 21(6):11–20, 2006

10. C. Cáceres, A. Fernández, S. Ossowski, and M. Vasirani. An Abstract Architecture for Service Coordination in IP2P Environments. In J. Cordeiro and J. Filipe, editors, *Computer Supported Acitivity Coordination*, pages 13–22. INSTICC Press, 2006

11. C. Cáceres, A. Fernández, S. Ossowski, and M. Vasirani. The CASCOM Abstract Architecture for Semantic Service Discovery and Coordination in IP2P Environments. In Y. Manolopoulos, J. Filipe, P. Constantopoulos, and J. Cordeiro, editors, *ICEIS (4)*, pages 205–208, 2006

12. S. Ossowski, J. Z. Hernández, M.-V. Belmonte, A. Fernández, A. García-Serrano, J.-L. P. de-la Cruz, J. M. Serrano, and F. T. Ruiz. Decision support for traffic management based on organisational and communicative multiagent abstractions. *Transportation Research Part C*, 13:272–298, 2005

13. S. Ossowski, J. Z. Hernández, M.-V. Belmonte, J.-M. Maseda, A. Fernández, A. García-Serrano, F. T. Ruiz, J. M. Serrano, and J.-L. P. de-la Cruz. Multi-Agent Systems For Decision Support: A Case

Study In The Transportation Management Domain. *Applied Artificial Intelligence*, 18(9-10):779–795, 2004

14. A. Fernández, S. Ossowski, and E. Alonso. Multiagent service architectures for bus fleet management. *Integrated Computer-Aided Engineering*, 11(2):101–115, 2004

7.2. Líneas futuras

A partir del trabajo desarrollado en esta tesis doctoral han surgido un conjunto de posibles líneas de investigación futuras en las que profundizar. A continuación se enumeran las más relevantes, clasificadas en cada una de las aportaciones principales.

Respecto a la descripción de servicios:

- La ontología de roles e interacciones no es una ontología fija a lo largo del tiempo sino que se puede extender si como consecuencia del análisis de nuevas dominios de aplicación se identifican nuevos roles/interacciones. Según se aplique el enfoque propuesto aquí a más dominios seguramente provocará la extensión de esta ontología.
- La propuesta realizada para la incorporación de información organizacional en la descripción de servicios utiliza los roles como elementos básicos. Esos roles son genéricos (por ejemplo, *Advisor*), por lo que se facilita la reusabilidad entre diferentes dominios. Además, el conjunto de roles tenidos en cuenta en el algoritmo de encaje no es muy grande por lo que se puede aumentar su eficiencia si se mantienen ya calculados los grados de encaje entre cada par de roles, lo cual es posible al tratarse de una cantidad reducida. Sin embargo, podría ser interesante analizar en detalle las consecuencias si se utilizasen los roles del dominio, lo que redundaría en una mayor expresividad y posiblemente mayor precisión a cambio de eficiencia computacional y reutilización. Esta comparativa puede ser interesante.

- En esta tesis se ha propuesto una forma de incorporar las descripciones de servicios basadas en roles en el lenguaje OWL-S. Es una línea futura la posible extensión también para otros enfoques emergentes como WSMO y SAWSDL.

Respecto a la búsqueda de servicios:

- Aunque el algoritmo de equiparación se ha aplicado sólo a la parte relacionada con la descripción de la información organizacional, igualmente podría extenderse para aplicarlo al resto de componentes de la descripción del servicio (entradas, salidas, parámetros, etc.). El encaje entre dos conceptos se podría reutilizar de forma sencilla, ya que consiste en analizar la posición dentro de la taxonomía correspondientes. En cambio, habría que estudiar la forma de agregación del valor de encaje de los distintos componentes. Además habría que analizar el efecto que tendría en cuanto a eficiencia tener que manejar ontologías bastante más grandes que la ontología de roles.
- Se podrían estudiar variaciones de la función de equiparación, tales como diferentes funciones de similitud entre roles, que utilicen otras medidas o enfoques distintas a la utilizada aquí (la distancia), así como otras medidas de agregación para los operadores lógicos y los diferentes elementos de la descripción. Quizá se puedan utilizar aquí técnicas de los campos de recuperación de información y/o lógica borrosa.
- Una línea de investigación en la que ya se ha comenzado a trabajar ([40]) es la descripción de un marco genérico para la equiparación de servicios Web semánticos. El objetivo es combinar técnicas basadas en lógicas descriptivas con medidas de similitud entre conceptos.

Respecto al filtrado para composición de servicios:

- El enfoque propuesto sobre el filtrado de servicios para composición tiene como una heurística importante la presencia en el pasado de servicios

en planes. Es decir, sin necesidad de realizar el complejo proceso de planificación, se basa en las veces que un servicio formó parte de un plan. Sin embargo, no se tiene en cuenta el resultado de la ejecución de dicho plan, por ejemplo si el plan pudo finalmente llevarse a cabo, si cumplió las expectativas del usuario, etc. Una posible línea futura es la exploración de la manera de incorporar esta información (si se dispone) en el cálculo de la relevancia de los servicios. Entonces el filtro no sólo tendría como objetivo reducir la complejidad computacional del proceso de planificación sino además la efectividad del plan. Sin embargo, esta realimentación es más difícil de conseguir, ya que requiere de la participación del usuario del servicio, mientras que la propuesta actual sólo requiere la información del plan compuesto.

- La evaluación del filtro se ha realizado mediante simulaciones. Conforme avance el desarrollo de la tecnología de servicios Web semánticos en general, y de la composición en particular, podrán realizarse pruebas con grandes colecciones de servicios reales. Por tanto, este evaluación profunda con casos reales será tema de futuro.

- Igualmente, relacionado también con la evaluación empírica, ésta se ha realizado de forma independiente al planificador utilizado, de forma que no se vea afectado por éste. Es tema de trabajo futuro la evaluación combinada con planificadores reales, tanto de forma independiente como combinada (filtro común para varios planificadores). En particular, puede ser interesante analizar el comportamiento de varios planificadores si se utiliza el filtro aprendiendo de sus propios planes o de los planes compuestos por otro planificador.

- La técnica de aprendizaje para obtener la relevancia de los servicios propuesta en esta tesis es puramente estadística, y se basa en la proporción de veces que han aparecido en los planes. Se podrían contemplar otras técnicas de aprendizaje más elaboradas como son aprendizaje por refuerzo o basado en casos, y que tuvieran en cuenta por ejemplo la antigüedad de los planes. Tanto la técnica utilizada como las otras posibles podrían ser analizadas empíricamente en un futuro.

- En el marco de filtro propuesto las clases se consideran independientes entre sí en el sentido que al comparar una clase c_1 con c_2, o son iguales o son distintas. Podría ser interesante analizar si el grado de similitud debido a su definición en la ontología aportaría mejoras en el cálculo de la relevancia. Es decir, si una clase c_1 es muy relevante (dada una solicitud de servicio compuesto) y c_2 es muy similar a c_1 podría también considerarse c_2 muy relevante. Aquí se podría integrar en el cálculo de la relevancia técnicas de similitud de conceptos, o en el caso concreto del filtro basado en roles la propuesta de encaje realizada en el capítulo 4.
- Otro tema pendiente que puede ser interesante es cómo definir las clases a las que pertenecen los servicios. Aquí se han propuesto dos enfoques, uno utilizando los roles y otro la categoría del servicio. En ambos casos, cada concepto de la ontología correspondiente (roles/categorías) se considera una *clase de servicio*. Sin embargo, se podría estudiar la posibilidad de agrupar varios de esos conceptos en una misma clase, lo cual será necesario si la ontología es muy grande. Se trataría, por tanto, de realizar clustering de conceptos en ontologías. Además también se pueden explorar otras posibles instanciaciones del filtro, distintas a los roles y categorías. Relacionado con esto, si se dispone de una base de datos de planes, se podrían utilizar técnicas de data mining para buscar la mejor agrupación en clases de servicios.

Respecto a las aplicaciones:

- La aplicación de las propuestas realizadas en esta tesis a otros dominios es claramente un tema de futuro trabajo. A este respecto se están explorando diferentes escenarios de coordinación de servicios para la ayuda a personas mayores.
- En posibles transferencias de la tecnología de gestión de trafico rodado descrito en la sección 6.1.2 a otras escenarios y ciudades, se prevé una reimplementación de los componentes software actuales de acuerdo con el enfoque expuesto en esta tesis.

Apéndice A

Abreviaturas

ACL: Agent Communication Language

BPEL4WS: Business Process Execution Language for Web Services

FIPA: Foundation for Intelligent Physical Agents

FND: Forma Normal Disyuntiva

MAS: Multiagent System

OWL: Ontology Web Language

OWL-S: OWL Services

PSMV: Paneles de Señalización de Mensajes Variables

RDF: Resource Description Framework

RDFS: RDF Scheme

SAD: Sistema de Ayuda a la Decisión

SAWSDL: Semantic Annotations for WSDL and XML Schema

SMA: Sistema Multi-Agente

SMAOS: Sistema Multi-Agente Orientado a Servicios

SOAP: Simple Object Access Protocol

SOC: Service Oriented Computing

SW: Semantic Web

SWS: Semantic Web Services

UDDI: Universal Description, Discovery and Integration

WSDL: Web Services Description Language

WSMO: Web Service Modeling Ontology

W3C: World Wide Web Consortium

Apéndice B

Conclusions

An analysis of current research trends in the field of agent technology reveals that organizational concepts like roles and types of interactions are fundamental for the construction of multiagent systems. This claim is backed by the fact that most (if not all) multiagent design methodologies concede great importance to these kinds of models. On the other hand, current approaches to service descriptions (usually semantic Web services) are mainly focused on service inputs and outputs and, in some cases, other information such as preconditions, effects or categories. However, organizational information, so important in MAS design models, has not yet been explicitly taken into account for characterising services in the particular context of service-oriented MAS. This thesis has put forward a method to bridge this gap by endowing service descriptions with information about the organizational context in which services can be used, in terms of the roles that they are able to play when they participate in different types of interactions. Our proposal extends existing approaches insofar as it adds organisational information that can increase the expressivity of service descriptions which, in turn, benefits the semantic service discovery methods. The thesis has described a method for incorporating this information into OWL-S service descriptions.

In addition, a service matching algorithm has been proposed, which is based on

the organizational information included in service descriptions. The proposed method combines several current approaches to semantic service matching. In order to calculate the degree of match between two service descriptions, firstly, the logical subsumption relation between concepts is taken into account following Paolucci's proposal [92] (exact, plug-in, subsumes, fail). Then, this value is refined making use of the distance between the two concepts in the ontology (Rada [96]) so as to obtain a numeric value in the real range [0,1]. On this basis, we have built a matching mechanism with a high level of discrimination, a fact that greatly facilitates service ranking. The implementation of the algorithm (the ROWLS software component) has been evaluated experimentally in combination with OWLS-MX [67], one of the leading general-purpose matchmakers available to-date. For the experiments, a subset of the OWLS-TC v2 was used. Selected service descriptions were annotated with organisational information (roles) and new services and queries added. The experimental results showed that the combination of both matchmakers outperforms the standalone version of OWLS-MX in both efficiency and effectiveness. In addition, it revealed that load time (parsing files and creating internal data representation) is decisive, so it is advisable, if possible, to integrate the matching process with the directory so as to minimize the number of service loads (maintaining the parsed services in the directory).

If no adequate services are available for a specific request, a planning functionality can be used to build up composite services. In open large-scale service-oriented MAS, a pure AI planning approach can become impracticable due to the vast number of services (operators) that are usually registered in the directory. In this thesis we propose a generic method for filtering out those services that are probably irrelevant to the planning process. We make use of class based information so as to cluster services based on relevant properties. In particular, two methods to obtain service class information have been described: (i) one obtained from the role and interaction taxonomies that are derived from the MAS organizational model, and (ii) one based on service categories. Although service filtering might affect the completeness of the planning process, this risk can be assumed as otherwise the complexity of

the planning process may become prohibitive.

Despite several initiatives, a suitable test collection for semantic web service composition planning is still to come. For the purpose of this thesis, it was also necessary to include role-based annotations. Therefore, we have generated a test collection randomly following two different methods. Moreover, the evaluation experiments have been carried out independently of a particular service composition planner, so as to avoid that planner behaviour could affect the results. The experiments revealed that the ratio of plans available after filtering is excellent even if a significant number of services is filtered. Furthermore, it is almost always still possible to find at least one plan. This can be relevant as in some situations (e.g. limited resources) one might be willing to reduce the completeness ratio as long as it can be assured that one plan can be found. Though limited in number, there are some other filtering approaches to composition planning in recent literature (Sirin et al. [113], Synthy [1], MetaComp [12], Constantinescu [11]). The main difference with our approach is the use of historical information to estimate service relevance, our heuristics, and the way of clustering services into classes.

The proposals put forward in this thesis have been applied to real world domains. The domain of transportation management has been used to illustrate how the flexibility and adaptability of a MAS can be improved by following a service-oriented approach, and the exploitation of its organizational models for service descriptions. Both the matchmaker and filter components have been incorporated into the service-oriented multiagent platform developed by the EU project CASCOM. Both components are part of a software demonstrator in the field of medical emergency management, which has shown excellent performance in several real-world trials.

Bibliografía

[1] V. Agarwal, G. Chafle, K. Dasgupta, N. M. Karnik, A. Kumar, S. Mittal, and B. Srivastava. Synthy: A System for End to End Composition of Web Services. *Journal of Web Semantics*, 3(4):311–339, 2005.

[2] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. Schmidt, A. Sheth, and K. Verma. Web Service Semantics: WSDL-S. W3C Member Submission, 2005. Available from http://www.w3.org/Submission/WSDL-S/.

[3] C. Alsina, M. Frank, and B. Schweizer. *Associative Functions: Triangular Norms And Copulas.* World Scientific, Singapore, 2006.

[4] K. Arisha, T. Eiter, S. Kraus, F. Ozcan, R. Ross, and V. S. Subrahmanian. IMPACT: Interactive Maryland Platform for Agents Collaborating Together. *IEEE Intelligent Systems*, 14(2):64–72, 1999.

[5] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, New York, NY, USA, 2003.

[6] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison-Wesley, 1999.

[7] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology).* John Wiley & Sons, 2007.

[8] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, and C. von Riegen. Universal Description, Discovery and Integration. http://www.uddi.org/.

[9] M.-V. Belmonte, J. L. P. de-la Cruz, F. Triguero, and A. Fernández. Agent Coordination for Bus Fleet Management. In *SAC '05: Proceedings of the ACM symposium on Applied computing*, pages 462–466, New York, NY, USA, 2005. ACM.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[11] W. Binder, I. Constantinescu, and B. Faltings. Directory Support for Large-Scale, Automated Service Composition. In T. Gschwind, U. Aßmann, and O. Nierstrasz, editors, *Software Composition*, volume 3628 of *Lecture Notes in Computer Science*, pages 57–66. Springer, 2005.

[12] B. Blankenburg, L. Botelho, A. Fernández, M. Klusch, and S. Ossowski. *CASCOM: Intelligent Service Coordination in the Semantic Web*, chapter Service Composition. Birkhäuser, 2008.

[13] A. Borgida, T. Walsh, and H. Hirsh. Towards Measuring Similarity in Description Logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Description Logics*, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[14] L. Botelho, A. Fernández, M. Klusch, L. Pereira, T. Santos, P. Pais, and M. Vasirani. *CASCOM: Intelligent Service Coordination in the Semantic Web*, chapter Service Discovery. Birkhäuser, 2008.

[15] BPEL4WS. Business Process Execution Language for Web Services, version 1.1. http://www.ibm.com/developerworks/library/specification/ws-bpel/.

[16] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004.

[17] J. Bryson, D. Martin, S. McIlraith, and L. Stein. Toward Behavioral Intelligence in the Semantic Web. *IEEE Computer*, 35(11):48–54, 2002.

[18] C. Cáceres, A. Fernández, S. Ossowski, and M. Vasirani. An Abstract Architecture for Service Coordination in IP2P Environments. In J. Cordeiro and J. Filipe, editors, *Computer Supported Acitivity Coordination*, pages 13–22. INSTICC Press, 2006.

[19] C. Cáceres, A. Fernández, S. Ossowski, and M. Vasirani. The CASCOM Abstract Architecture for Semantic Service Discovery and Coordination in IP2P Environments. In Y. Manolopoulos, J. Filipe, P. Constantopoulos, and J. Cordeiro, editors, *ICEIS (4)*, pages 205–208, 2006.

[20] G. Caire, W. Coulier, F. J. Garijo, J. Gomez, J. Pavón, F. Leal, P. Chainho, P. E. Kearney, J. Stark, R. Evans, and P. Massonet. Agent Oriented Analysis Using MESSAGE/UML. In *AOSE'01: Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering II*, pages 119–135, London, UK, 2002. Springer-Verlag.

[21] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M.-C. Shan. Adaptive and Dynamic Service Composition in eFlow. In *Conference on Advanced Information Systems Engineering*, pages 13–31, 2000.

[22] CASCOM. Use Case Scenarios, CASCOM Deliverable D3.1. http://www.ist-cascom.org/dmdocuments/d3.1-v1.1.pdf, 2004.

[23] A. R. Cassandra, D. Chandrasekara, and M. H. Nodine. Capability-based Agent Matchmaking. In *Agents-2000. Conference on Autonomous Agents*, pages 201–202, 2000.

[24] C. Cáceres, A. Fernández, S. Ossowski, and M. Vasirani. Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach. *IEEE Intelligent Systems*, 21(6):11–20, 2006.

[25] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001.

[26] J. Cuena and S. Ossowski. Distributed Models for Decision Support. In G. Weiss, editor, *Multi-agent Systems – A Modern Approach to DAI*, pages 459–504. MIT Press: Cambridge, 1999.

[27] K. Dam and M. Winikoff. Comparing Agent-Oriented Methodologies. In *Proceedings of the 5th Int'l Bi-Conference Workshop on AgentOriented Information Systems (AOIS)*, 2003.

[28] M. Dastani, J. Hulstijn, F. Dignum, and J.-J. C. Meyer. Issues in Multiagent System Development. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 922–929, Washington, DC, USA, 2004. IEEE Computer Society.

[29] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, H. Lausen, E. Oren, A. Polleres, D. Roman, J. Scicluna, and M. Stollberg. Web Service Modeling Ontology (WSMO). W3C Member Submission, 2005. Available from http://www.w3.org/Submission/WSMO/.

[30] J. de Bruijn, D. Fensel, U. Keller, M. K. H. Lausen, R. Krummenacher, A. Polleres, and L. Predoiu. Web Service Modeling Language (WSML). W3C Member Submission, 2005. Available from http://www.w3.org/Submission/WSML/.

[31] K. Decker, K. Sycara, and M. Williamson. Middle-Agents for the Internet. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[32] S. A. DeLoach, M. F. Wood, and C. H. Sparkman. Multiagent Systems Engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):231–258, 2001.

[33] M. B. Do and S. Kambhampati. Sapa: A Multi-objective Metric Temporal Planner. *Journal of Artificial Intelligence Research (JAIR)*, 20:155–194, 2003.

[34] M. Ehrig, P. Haase, N. Stojanovic, and M. Hefke. Similarity for Ontologies - A Comprehensive Framework. In *Proceedings of the 13th European Conference on Information Systems (ECIS 2005), Regensburg, Germany, May 26-28, 2005*, Regensburg, 2005.

[35] N. Fanizzi and C. d'Amato. A Similarity Measure for the ALN Description Logic. In *Proceedings of CILC 2006 - Italian Conference on Computational Logic, Bari, Italy, June 26-27, 2006*, 2006.

[36] J. Farrell and H. Lausen. Semantic Annotations for WSDL and XML Schema (SAWSDL). W3C Recommendation 28 August 2007. Available from http://www.w3.org/TR/sawsdl/.

[37] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[38] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services : The Web Service Modeling Ontology*. Springer, 2006.

[39] A. Fernández, S. Ossowski, and E. Alonso. Multiagent service architectures for bus fleet management. *Integrated Computer-Aided Engineering*, 11(2):101–115, 2004.

[40] A. Fernández, A. Polleres, and S. Ossowski. Towards Fine-grained Service Matchmaking by Using Concept Similarity. In *First International Joint Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at ISWC 2007*, pages 29–43, 2007.

[41] A. Fernández, M. Vasirani, C. Cáceres, and S. Ossowski. Role-based Service Description and Discovery. In *Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE)*, Hakodate, Japan, 2006.

[42] A. Fernández and S. Ossowski. Exploiting Organizational Models for Semantic Service Description, Matchmaking and Composition in Service-Oriented Multi-Agent Systems. *ERCIM News*, 70:53–54, 2007.

[43] A. Fernández and S. Ossowski. Filters for Semantic Service Composition in Service-oriented Multiagent Systems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 924–926, Honolulu, Hawai, USA, 2007.

[44] A. Fernández and S. Ossowski. Role-based Discovery and Coordination of Semantic Transportation Services. In *IEEE Intelligent Transportation Systems Conference*, pages 100–106, Seattle, Washington, USA, 2007.

[45] A. Fernández and S. Ossowski. Semantic Service Composition in Service-Oriented Multiagent Systems: A Filtering Approach. In J. Huang, R. Kowalczyk, Z. Maamar, D. L. Martin, I. Müller, S. Stoutenburg, and K. P. Sycara, editors, *Service-Oriented Computing: Agents, Semantics, and Engineering, AAMAS 2007 International Workshop (SOCASE)*, volume 4504 of *Lecture Notes in Computer Science*, pages 78–91. Springer, 2007.

[46] A. Fernández, M. Vasirani, C. Cáceres, and S. Ossowski. A Role-Based Support Mechanism for Service Description and Discovery. In J. Huang, R. Kowalczyk, Z. Maamar, D. L. Martin, I. Müller, S. Stoutenburg, and K. P. Sycara, editors, *Service-Oriented Computing: Agents, Semantics, and Engineering, AAMAS 2007 International Workshop (SOCASE)*, volume 4504 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2007.

[47] Foundation for Intelligent Physical Agents. *FIPA ACL Message Structure Specification*. http://www.fipa.org/specs/fipa00061, 2002.

[48] Foundation for Intelligent Physical Agents. *FIPA Agent Communication Language*. http://www.fipa.org/repository/aclspecs.html, 2002.

[49] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Protocol for HTTP Specification*. http://www.fipa.org/specs/fipa00084/SC00084F.html, 2002.

[50] Foundation for Intelligent Physical Agents. *FIPA Communicative Act Library Specification*. http://www.fipa.org/specs/fipa00037, 2002.

[51] Foundation for Intelligent Physical Agents. *FIPA Interaction Protocol Library Specification*. http://www.fipa.org/repository/ips.html, 2002.

[52] J. A. Giampapa, M. Paolucci, and K. Sycara. Agent Interoperation across Multiagent System Boundaries. In C. Sierra, M. Gini, and J. S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 179–186, Barcelona, Catalonia, Spain, 2000. ACM Press.

[53] N. Glaser. The CoMoMAS Methodology and Environment for Multi-Agent System Development. In *Revised Papers from the Second Australian Workshop on Distributed Artificial Intelligence*, pages 1–16, London, UK, 1997. Springer-Verlag.

[54] R. Hamadi and B. Benatallah. A Petri net-based model for web service composition. In *ADC '03: Proceedings of the 14th Australasian database conference*, pages 191–200, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[55] H. Helin, M. Klusch, A. Lopes, A. Fernández, M. Schumacher, H. Schuldt, F. Bergenti, and A. Kinnunen. Context-aware Business Application Service Co-ordination in Mobile Computing Environments. In *AAMAS05 Workshop on Ambient Intelligence - Agents for Ubiquitous Computing*, 2005.

[56] H. Helin, M. Klusch, and M. Schumacher, editors. *CASCOM: Intelligent Service Coordination in the Semantic Web*. Birkhäuser Verlag, 2008.

[57] E. F. Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003.

[58] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR)*, 14:253–302, 2001.

[59] M. N. Huhns and M. P. Singh. *Service-Oriented Computing*. John Wiley & Sons, 2005.

[60] M. N. Huhns and M. P. Singh. Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

[61] C. Iglesias, M. Garrijo, and J. Gonzalez. A Survey of Agent-Oriented Methodologies. In J. Müller, M. P. Singh, and A. S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *LNAI*, pages 317–330, Berlin, July 04–07 1999. Springer.

[62] C. A. Iglesias, M. Garijo, J. Centeno-Gonzalez, and J. R. Velasco. Analysis and Design of Multiagent Systems Using MAS-Common KADS. In *Agent Theories, Architectures, and Languages*, pages 313–327, 1997.

[63] J. J. Jiang and D. W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *CoRR*, cmp-lg/9709008, 1997.

[64] T. Juan, A. Pearce, and L. Sterling. ROADMAP: Extending the Gaia Methodology for Complex Open Systems. In *AAMAS'02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 3–10, New York, NY, USA, 2002. ACM Press.

[65] D. Kinny and M. P. Georgeff. Modelling and Design of Multi-Agent Systems. In M. P. Singh, A. S. Rao, and M. Wooldridge, editors, *ATAL*, volume 1365 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 1997.

[66] E. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer, Dordrecht, 2000.

[67] M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with OWLS-MX. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922, New York, NY, USA, 2006. ACM Press.

[68] M. Klusch, A. Gerber, and M. Schmitdt. Semantic Web Service Composition Planning with OWLS-XPlan. In *AAAI Fall Symposium on Agents and the Semantic Web (Arlington VA, USA)*, 2005.

[69] M. Klusch and K. P. Sycara. Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In *Coordination of Internet Agents: Models, Technologies, and Applications*, pages 197–224. Springer-Verlag, 2001.

[70] C. Leacock and M. Chodorow. Combining local context and WordNet similarity for word sense identification. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 265–283. MIT Press, 1998.

[71] F. Leymann. Web Services: Distributed Applications Without Limits. In G. Weikum, H. Schöning, and E. Rahm, editors, *In Proceedings of Database Systems for Business, Technology and Web (BTW)*, volume 26 of *Lecture Notes in Informatics*, pages 2–23. GI, 2003.

[72] F. Leymann. Combining Web Services and the Grid: Towards Adaptive Enterprise Applications. In *Proceedings of the 17th International Conference on Advanced Information Systems Engineering, CAiSE 2005 Workshops, Vol. 2, Porto, Portugal*, pages 9–21, 2005.

[73] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. *Int. J. of Electronic Commerce*, 8(4):39–60, 2004.

[74] Y. Li, Z. Bandar, and D. McLean. An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882, 2003.

[75] D. Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[76] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005.

[77] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDemott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for

Web Services. W3C Member Submission, 2004. Available from http://www.w3.org/Submission/OWL-S/.

[78] D. Martin, M. Burstein, S. McIlraith, M. Paolucci, and K. Sycara. *Extending Web Services Technologies*, chapter OWL-S and Agent-Based Systems. Springer, 2004.

[79] D. Mcdermott. PDDL — the planning domain definition language, 1998.

[80] S. A. McIlraith and T. C. Son. Adapting Golog for Composition of Semantic Web Services. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, pages 482–496, 2002.

[81] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing Web services on the Semantic Web. *The VLDB Journal*, 12(4):333–351, 2003.

[82] NAICS Association. NAICS code searching. http://www.naics.com/search.htm, 2004.

[83] D. Nau, T. C. Au, O. Ilghami, U. Kuter, W. J. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research*, 20:379–404, December 2003.

[84] M. H. Nodine, A. H. H. Ngu, A. R. Cassandra, and W. Bohrer. Scalable Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth$^{\mathrm{TM}}$. *IEEE Transactions on Knowledge and Data Engineering.*, 15(5):1082–1098, 2003.

[85] T. D. Noia, E. D. Sciascio, F. M. Donini, and M. Mongiello. Semantic Matchmaking in a P-2-P Electronic Marketplace. In *In Proceedings of the 2003 ACM Symposium on Applied Computing (SAC)*, pages 582–586. ACM, 2003.

[86] OMG. The Object Management Group: Meta-Object Facility. http://www.omg.org/technology/documents/formal/mof.htm, 2002.

[87] S. Ossowski. *Co-ordination in Artificial Agent Societies, Social Structure and Its Implications for Autonomous Problem-Solving Agents*, volume 1535 of *Lecture Notes in Computer Science*. Springer, 1999.

[88] S. Ossowski, J. Z. Hernández, M.-V. Belmonte, A. Fernández, A. García-Serrano, J.-L. P. de-la Cruz, J. M. Serrano, and F. T. Ruiz. Decision support for traffic management based on organisational and communicative multiagent abstractions. *Transportation Research Part C*, 13:272–298, 2005.

[89] S. Ossowski, J. Z. Hernández, M.-V. Belmonte, J.-M. Maseda, A. Fernández, A. García-Serrano, F. T. Ruiz, J. M. Serrano, and J.-L. P. de-la Cruz. Multi-Agent Systems For Decision Support: A Case Study In The Transportation Management Domain. *Applied Artificial Intelligence*, 18(9-10):779–795, 2004.

[90] OWL. Web Ontology Language. http://www.w3.org/2004/OWL/.

[91] L. Padgham and M. Winikoff. Prometheus: A Methodology for Developing Intelligent Agents. In F. Giunchiglia, J. Odell, and G. Weiß, editors, *Proceedings of the Third International Workshop on AgentOriented Software Engineering (AOSE), at AAMAS 2002*, volume 2585 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2002.

[92] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic Matching of Web Services Capabilities. In I. Horrocks and J. A. Hendler, editors, *International Semantic Web Conference*, volume 2342 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 2002.

[93] M. P. Papazoglou and D. Georgakopoulos. Service-Oriented Computing: Introduction. *Communications of the ACM*, 46(10):24–28, 2003.

[94] J. Pavón and J. J. Gómez-Sanz. Agent Oriented Software Engineering with INGENIAS. In V. Maríık, J. P. Müller, and M. Pechoucek, editors, *Multi-Agent Systems and Applications III, 3rd International Central and*

Eastern European Conference on Multi-Agent Systems, (CEEMAS), volume 2691 of *Lecture Notes in Computer Science*, pages 394–403. Springer, 2003.

[95] S. R. Ponnekanti and A. Fox. SWORD: A developer toolkit for web service composition. In *Proceedings of the 11th International WWW Conference (WWW2002)*, Honolulu, HI, USA, 2002.

[96] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics,*, 19(1):17–30, 1989.

[97] RDF. Resource Description Framework. http://www.w3.org/RDF/.

[98] RDF. Resource Description Framework. http://www.w3.org/TR/rdf-schema/.

[99] P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the XI International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 448–453, 1995.

[100] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.

[101] G. Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[102] G. Schreiber and R. de Hoog. *Knowledge Engineering and Management: The CommonKADS Methodology.* MIT Press, 1999.

[103] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and Composing Service-Based nd Reference Process-Based Multi-enterprise Processes. In *CAiSE '00: Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, pages 247–263, London, UK, 2000. Springer-Verlag.

[104] J. Searle. *Speech Acts*. Cambridge University Press, 1969.

[105] J. Serrano. *Pragmática de los agentes software: análisis y diseño de lenguajes de comunicación artificiales*. PhD thesis, Universidad Rey Juan Carlos, 2004.

[106] J. M. Serrano and S. Ossowski. An Organisational Approach to the Design of Interaction Protocols. In M.-P. Huget, editor, *Communication in Multiagent Systems*, volume 2650 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2003.

[107] J. M. Serrano and S. Ossowski. A compositional framework for the specification of interaction protocols in multiagent organizations. *Web Intelligence and Agent Systems*, 5(2):197–214, 2007.

[108] J. M. Serrano, S. Ossowski, and A. Fernández. The Pragmatics of Software Agents - Analysis and Design of Agent Communication Languages. *Intelligent Information Agents - An AgentLink Perspective (Klusch, Bergamaschi, Edwards & Petta, ed.), Lecture Notes in Computer Science*, 2586:234–274, 2003.

[109] M. Sheshagiri, M. desJardins, and T. Finin. A Planner for Composing Services Described in DAML-S. In *Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering,*, June 2003.

[110] C. Sierra and J. Debenham. Trust and honour in information-based agency. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1225–1232, New York, NY, USA, 2006. ACM.

[111] M. S. Silver. *Systems that support decision makers: description and analysis*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[112] E. Sirin, B. Parsia, and J. A. Hendler. Filtering and Selecting Semantic Web Services with Interactive Composition Techniques. *IEEE Intelligent Systems*, 19(4):42–49, 2004.

[113] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for Web Service composition using SHOP2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396, October 2004.

[114] K. Sycara, M. Paolucci, M. V. Velsen, and J. Giampapa. The RETSINA MAS Infrastructure. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):29–48, 2003.

[115] K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems*, pages 173–203, 2002.

[116] S. Tarkoma and M. Laukkanen. Adaptive Agent-Based Service Composition for Wireless Terminals. In M. Klusch, S. Ossowski, A. Omicini, and H. Laamanen, editors, *CIA*, volume 2782 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2003.

[117] A. Tveit. A Survey of Agent-Oriented Software Engineering. Proc. of the First NTNU CSGS Conference (http://www.amundt.org), May 2001.

[118] A. Tversky. Features of Similarity. *Psychological Review*, 84(4):327–352, 1977.

[119] UNDP, Dun and Bradstreet Corporation. UNSPSC code searching. http://www.unspsc.org/, 2004.

[120] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

[121] W3C. SOAP Version 1.2 Part 0: Primer. http://www.w3.org/TR/soap/, April 2007.

[122] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[123] G. Weiss. *Multi-agent systems*. MIT Press, 1999.

[124] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

[125] H. C. Wong and K. P. Sycara. Adding Security and Trust to Multiagent Systems. *Applied Artificial Intelligence*, 14(9):927–941, 2000.

[126] M. Wooldridge. Agent-based software engineering. *IEE Proceedings Software Engineering*, 144(1):26–37, 1997.

[127] M. Wooldridge and P. Ciancarini. Agent-oriented software engineering: the state of the art. In *First international workshop, AOSE 2000 on Agent-oriented software engineering*, pages 1–28, Secaucus, NJ, USA, 2001. Springer-Verlag New York, Inc.

[128] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[129] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[130] D. Wu, B. Parsia, E. Sirin, J. A. Hendler, and D. S. Nau. Automating DAML-S Web Services Composition Using SHOP2. In D. Fensel, K. P. Sycara, and J. Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2003.

[131] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[132] J. Yang and M. P. Papazoglou. Service components for managing the life-cycle of service compositions. *Information Systems*, 29(2):97–125, 2004.

[133] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organizational abstractions for the analysis and design of multi-agent system. In *First international workshop, AOSE 2000 on Agent-oriented software engineering*, pages 235–251, Secaucus, NJ, USA, 2001. Springer-Verlag New York, Inc.

[134] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, July 2003.