Universidad de Granada

Escuela Técnica Superior de Ingeniería Informática
Departamento Ciencias de la Computación e I. A.

Knowledge Mobilization: Architectures, Models and Applications

TESIS DOCTORAL

Juan Gómez Romero

Editor: Editorial de la Universidad de Granada
Autor: Juan Gómez Romero
D.L.: GR.1795-2008
ISBN: 978-84-691-5643-8

Universidad de Granada

Knowledge Mobilization: Architectures, Models and Applications

Juan Gómez Romero

DIRECTOR
Miguel Delgado Calvo-Flores

Granada, Junio 2008

La memoria **Knowledge Mobilization: Architectures, Models and Applications**, que presenta D. Juan Gómez Romero para optar al grado de Doctor en Informática, ha sido realizada en el departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección del Profesor Dr. D. Miguel Delgado Calvo-Flores, Catedrático de Universidad.

Granada, Junio de 2008.

Miguel Delgado Calvo-Flores Juan Gómez Romero

A mis padres.

. . . The endless cycle of idea and action,
Endless invention, endless experiment,
Brings knowledge of motion, but not of
stillness;
Knowledge of speech, but not of silence;
Knowledge of words, and ignorance of the
Word.
All our knowledge brings us nearer to our
ignorance,
All our ignorance brings us nearer to
death,
But nearness to death no nearer to GOD.
Where is the Life we have lost in living?
Where is the wisdom we have lost in
knowledge?
Where is the knowledge we have lost in
information?

T.S. Eliot, *The Rock*

Abstract

Mobile network technologies are representative of a paradigm shift from classical desktop applications to highly-distributed nomadic systems. Mobile technologies make it possible to deliver information anywhere at anytime, and provide nomadic users with up-to-date information ready for decision-making processes. Nevertheless, the management of structured information (i.e. knowledge) for delivery to mobile users poses several challenges. Besides the limited computational capabilities of mobile devices, mobile systems face specific problems that cannot be solved by traditional knowledge management methodologies and tools, and thus require creative new solutions.

Knowledge Mobilization is an innovative approach that integrates contributions from several areas such as Knowledge Engineering, Distributed Artificial Intelligence, Soft Computing, and the Semantic Web, to develop effective knowledge-intensive mobile applications. In this doctoral thesis we explore possible computational solutions for problems of knowledge distribution and information overload in Knowledge Mobilization systems. More precisely, we describe a new architecture for Knowledge Mobilization systems (based on the multi-agent paradigm) and an innovative context-aware knowledge representation model (which relies on Description Logics ontologies). Both elements provide support for effectively delivering knowledge obtained from large, heterogeneous information sources to nomadic users. The architecture and knowledge representation model were used to design and implement a Knowledge Mobilization system in the domain of Healthcare. The system creates and distributes to nomadic doctors summaries of the clinical histo-

ries of their patients. The prototype thus implemented achieves the level of adequacy necessary to meet Knowledge Mobilization requirements.

Agradecimientos

Suele decirse que la investigación científica es una tarea que requiere vocación y dedicación. No poseo ninguna de estas dos cualidades, por lo que quiero agradecer el apoyo de las muchas personas que me han descubierto esta profesión y me han prestado su ayuda para la consecución de esta tesis.

En primer lugar, quiero expresar mi gratitud al profesor Miguel Delgado, mi director de tesis, que ha sido el principal impulsor de este trabajo. En el desarrollo de una tesis es habitual que el doctorando aporte el entusiasmo y el tutor los conocimientos para guiarlo. Me siento afortunado porque, en mi caso, Miguel ha sido quien ha puesto tanto la ilusión como la experiencia desde el primer día.

También me gustaría mencionar a mis compañeros del departamento de Ciencias de la Computación e Inteligencia Artificial y a los becarios de la sala XX. Un recuerdo especial es para Pedro Magaña, Fernando Bobillo y Mariló Ruiz; con ellos he compartido cafés, tertulias y viajes, y en ninguno de esos momentos han dejado de enseñarme cómo ser mejor investigador y, sobre todo, mejor persona.

Asimismo, quiero agradecer las revisiones de la profesora Pamela Faber, que han tenido un valor mucho más allá de lo puramente lingüístico. Si el lenguaje de esta tesis es correcto, es gracias su trabajo incansable y a su enorme capacidad de comprensión, en todos los sentidos. Igualmente, doy las gracias a Pilar León por ponerme en contacto con ella.

Por otra parte, me gustaría destacar la ayuda de los investigadores Enrico Motta y Bernardo Cuenca-Grau, quienes, a pesar de mis urgencias, han acce-

dido a preparar los informes necesarios para que esta tesis opte a la mención de Doctorado Europeo.

También agradezco el apoyo económico prestado por Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía, que ha financiado la realización de esta tesis dentro de su programa de Formación de Doctores (2004).

Una fase de este trabajo de investigación ha sido desarrollada en el Institute for Advanced Management Systems Research (Åbo Akademi, Finlandia). Quiero expresar mi gratitud hacia el profesor Christer Carlsson por su atención durante mi estancia en Turku y sus consejos científicos, que me sirvieron para deshacer mi maraña de ideas y palabras. Este agradecimiento es extensivo a todos los miembros del instituto, que me acogieron amablemente, y particularmente a Dan, Franck, Kristian y Alessandro. *Tack!*, compañeros.

Allí en Turku conocí a otros muchos *nómadas*, tantos que no cabrían en esta sección, que me ayudaron a sentir que uno no pertenece al lugar en el que ha nacido, sino a donde están las personas a las que quiere. Una parte de esta tesis se la debo a ellos.

Durante el período de trabajo más intenso de esta tesis, han sido muchos los amigos que me han demostrado su aprecio y su paciencia infinita. Ni siquiera las incontables veces en que he roto mis compromisos con ellos han hecho que dejen de darme ánimos. Gracias a mi gente de Granada y a mis *jarotes* por comprenderme, soportarme y disculparme.

Para terminar, quiero mencionar a mi familia, que ha sido mi principal apoyo durante toda la investigación. A mis hermanos, Ángel e Isabel, que me han acompañado en dos etapas diferentes, pero igualmente difíciles, de mis estudios, y me han dado todo su cariño. Nunca me acostumbraré a estar lejos de ellos. Y, por supuesto, a mis padres, Pedro e Isabel, de los que he aprendido las lecciones más valiosas de mi vida. Entre ellas, a no olvidar de dónde vengo, y a saber que con tesón puedo llegar a cualquier sitio que me proponga. A ellos va dedicada esta tesis.

Resumen

Motivación

La Ingeniería del Conocimiento es el área de la Inteligencia Artificial que investiga cómo representar las entidades (concretas o abstractas) de un dominio de aplicación mediante un modelo de conocimiento de modo que éste pueda ser usado por un procedimiento automático para resolver un problema complejo. La Ingeniería del Conocimiento se basa en otros campos clásicos como la Ontología, que estudia cómo se clasifican las entidades del dominio, y la Lógica, que se ocupa de los lenguajes formales y las reglas de inferencia aplicables a los modelos de representación. Estos modelos de conocimiento proporcionan el soporte elemental para la implementación de Sistemas Inteligentes; cuanto más conocimiento maneje un agente, más *inteligente* será, y por lo tanto podrá resolver más problemas mejor.

Un Sistema Basado en Conocimiento (SBC) es un sistema software que gestiona y aplica conocimiento representado explícitamente para resolver problemas de decisión complejos. El objetivo de los SBC es asistir a los usuarios en los procesos de toma de decisiones, proporcionando la información apropiada, a la persona adecuada y en el momento justo.

Esta definición de SBC sigue siendo válida hoy en día, pero debe tenerse en cuenta que los SBC actuales se enfrentan a nuevos problemas. La tecnologías de telecomunicaciones han hecho que la "persona adecuada" pueda estar situada en cualquier lugar, y que el "momento justo" sea cualquier instante. Asimismo, Internet ha hecho que sea posible acceder a grandes colec-

ciones de información útil que antes no estaban disponibles, lo que significa que la "información apropiada" es el resultado de integrar datos obtenidos de diferentes fuentes, posiblemente distribuidas y con diferente semántica y formato.

En el caso más extremo, los SBC deben proporcionar soporte a usuarios móviles (o *nómadas*). Un sistema móvil es una aplicación a la que puede accederse remotamente utilizando un dispositivo de bolsillo (como un teléfono móvil o una PDA) y una red de comunicaciones inalámbrica. Las tecnologías móviles, que abarcan desde el estudio de las propiedades físicas de los canales de comunicación (menor nivel de abstracción) hasta el desarrollo de aplicaciones para dispositivos de bolsillo (mayor nivel de abstracción), permiten dotar de nuevas funcionalidades a los SBC.

En consecuencia, un SBC moderno debe incorporar las fuentes disponibles que contengan información útil y facilitar el acceso a los usuarios móviles. Esto supone todo un reto para el desarrollo de Sistemas Inteligentes, debido a las limitaciones computacionales de los dispositivos móviles: tamaño de pantalla reducido, poca fiabilidad y limitado ancho de banda de las redes de comunicación móviles, duración de la batería, heterogeneidad de plataformas de desarrollo y dispositivos, etc.

Algunas de estas dificultades pueden superarse con la eventual mejora de las tecnologías móviles. Por ejemplo, las nuevas redes móviles ofrecen velocidades de transferencia del orden de Mb/s. No obstante, otros problemas son intrínsecos a los SBC móviles y, aunque podrán disminuirse conforme las tecnologías evolucionen, requieren soluciones desde el punto de vista de las Ciencias de la Computación. El conocimiento en estos sistemas debe obtenerse de fuentes de información de gran tamaño, distribuidas y heterogéneas, y debe ser transmitdo a los dispositivos móviles de los usuarios. Además, estos sistemas deben ser conscientes de contexto, en el sentido de que su funcionamiento debe adaptarse al entorno del usuario, y evitar la sobrecarga de información, esto es, no es admisible que los usuarios reciban más información de la que pueden asimilar. Todo esto requiere el uso de modelos de representación y arquitecturas software que permitan extraer, representar,

integrar, filtrar y presentar este conocimiento adecuadamente.

En las últimas décadas, la Inteligencia Artificial Distribuida ha profundizado en el desarrollo de sistemas inteligentes distribuidos. La Inteligencia Artificial Distribuida propone tecnologías y paradigmas de computación para diseñar e implementar aplicaciones inteligentes distribuidas. El paradigma multiagente o, más recientemente, las tecnologías basadas en Web, han sido utilizadas por la Inteligencia Artificial Distribuida para implementar este tipo de sistemas.

El problema de la integración de fuentes de información ha dado lugar a la aparición de nuevos modelos de representación y metodologías de adquisición del conocimiento en el ámbito de la Ingeniería del Conocimiento, dado que las técnicas clásicas no facilitan esta tarea. Así, las ontologías, que se definen como un formalismo de representación que promueve la integración y la reutilización de conocimiento previo, han tenido bastante éxito en los últimos años.

Una fuente de información muy extensa, al tiempo que heterogénea, que contiene recursos que forman parte de los SBC actuales es la Web. La Web Semántica es una iniciativa reciente que propone describir la semántica de los contenidos de la Web actual mediante metadatos, de manera que sea posible buscar, recuperar e integrar información automáticamente. La Web Semántica recoge numerosas contribuciones de la Ingeniería del Conocimiento (y viceversa), como por ejemplo las ontologías, que son utilizadas para representar formalmente los metadatos.

A pesar de las contribuciones de la Inteligencia Artificial Distribuida y la Web Semántica, entre otras, los problemas computacionales de los SBC con soporte móvil no han sido satisfactoriamente resueltos hasta la fecha. El diseño de una aplicación móvil que necesite gestionar una cantidad considerable de conocimiento es más un arte que una ciencia, ya que las metodologías de Ingeniería del Software que se utilizan en los sistemas de escritorio no son directamente aplicables a los sistemas móviles. Por este motivo, se necesitan nuevos patrones de diseño específicamente orientados a este tipo de

sistemas. Por otra parte, la continua evolución de las tecnologías de comunicación móvil favorece la implementación de características innovadoras, pero al mismo tiempo dificulta el establecimiento de un marco de desarrollo duradero. En lo que respecta a los modelos de representación, existen muy pocos formalismos que solucionen los problemas que plantea la distribución de conocimiento en sistemas móviles inteligentes. Aunque las ontologías son una aproximación prometedora, y pueden usarse en combinación con las tecnologías de Web Semántica, esta línea de trabajo todavía no ha sido suficientemente explorada.

Propuesta

La Movilización del Conocimiento es una nueva perspectiva descrita por autores como Keen y Carlsson que propone combinar contribuciones de diferentes áreas, como la Inteligencia Artificial Distribuida, las ontologías, la Web Semántica y el Soft Computing (principalmente la Lógica Difusa), para ofrecer soluciones integrales a los problemas que presenta la construcción de SBC móviles. La Movilización del Conocimiento tiene como objectivo proporcionar un marco conceptual y de desarrollo común para el desarrollo de sistemas móviles inteligentes.

Esta tesis doctoral presenta un trabajo de investigación en Movilización del Conocimiento y, por extensión, en Ingeniería del Conocimiento e Inteligencia Artificial Distribuida. De acuerdo a los problemas anteriormente mencionados que dificultan la movilización del conocimiento (limitaciones de los dispositivos móviles, integración de fuentes de información heterogéneas, consciencia de contexto, sobrecarga de información, etc.), en esta tesis se estudian arquitecturas software y modelos de representación para gestionar y distribuir conocimiento obtenido de fuentes de información de gran tamaño y heterogéneas a usuarios nómadas.

De acuerdo a este planteamiento, el documento de tesis expone las dos aportaciones principales de este trabajo: una arquitectura organizativa y un

modelo de representación de conocimiento contextual para sistemas de Movilización del Conocimiento. Las contribuciones de estas dos propuestas se muestran con el diseño y la implementación de la aplicación IASO, un sistema de Movilización del Conocimiento en el área de la salud.

La arquitectura presentada describe los componentes de un sistema de Movilización del Conocimiento y las relaciones entre los mismos. Esta arquitectura puede adaptarse para diseñar distintos tipos de aplicaciones móviles que requieran diferentes esquemas de comunicación. Junto a la arquitectura, también se estudian varias tecnologías que puede utilizarse para implementar sistemas de Movilización del Conocimiento basados en ella. En consecuencia, la nueva arquitectura y las tecnologías de implementación asociadas sirven como soporte a la implementación de Sistemas de Movilización del Conocimiento.

El modelo de representación del conocimiento ha sido planteado como un patrón de diseño de ontologías. Este patrón de diseño establece un conjunto de reglas para construir ontologías que representan qué información del dominio es interesante o *significativa* en cada contexto. A partir de una ontología de significancia y utilizando el algoritmo de inferencia propuesto, puede inferirse qué información es relevante en una situación dada. En consecuencia, este modelo permite la activación selectiva del conocimiento según el contexto, lo cual puede utilizarse para resolver el problema de la sobrecarga de información.

La aplicación IASO es un sistema de Movilización del Conocimiento en el ámbito del cuidado de la salud diseñado e implementado en esta tesis. El desarrollo de IASO ha estado guidado por los principios de la Movilización del Conocimiento, y está basado en la nueva arquitectura abstracta y el modelo de representación. IASO permite a los profesionales médicos acceder a las historias clínicas almacenadas en el Sistema de Información de un Hospital utilizando dispositivos móviles. La característica más importante de IASO es que proporciona a los usuarios del sistema únicamente los registros que son significativos dada la situación clínica en la que se encuentran. De esta forma, los usuarios reciben un resumen de toda la información disponible

creado automáticamente a partir de conocimiento sobre el contexto de la consulta. El prototipo de aplicación presentado en esta tesis incorpora una adaptación de la base de datos del Hospital Clínico San Cecilio de Granada.

El documento de tesis presenta con detalle estas propuestas y discute la motivación de cada una de ellas, así como las ventajas que aportan respecto a los trabajos relacionados existentes en la literatura.

Contenidos

La memoria de tesis resume el trabajo de investigación que ha sido llevado a cabo para desarrollar las contribuciones mencionadas en la sección anterior. En el documento se ha distinguido claramente entre las aportaciones originales (nuevas propuestas formuladas en esta tesis) y los trabajos de otros autores (propuestas creadas en disciplinas relacionadas que son aplicadas en la tesis).

Los capítulos 2-5 presentan las aportaciones originales de la tesis, respectivamente: (i) una revisión del concepto de Movilización del Conocimiento; (ii) una arquitectura abstracta para sistemas de Movilización del Conocimiento; (iii) un modelo de representación del conocimiento contextual; (iv) el diseño y la implementación de una aplicación de Movilización del Conocimiento. Los apéndices A y B repasan, respectivamente, la representación de conocimiento mediante Lógica de Descripciones y la Web Semántica, dos disciplinas que componen la base de este trabajo.

A continuación, se detallan los contenidos de cada una de estas secciones.

Capítulo 1. Preliminares

En este capítulo introductorio se explica la motivación global de esta tesis doctoral y los objetivos que se persiguen en este trabajo. Además, se describe la metodología constructivista, que ha sido aplicada para llevar a cabo esta investigación.

Capítulo 2. Movilización del Conocimiento

El capítulo 2 describe qué es la Movilización del Conocimiento y qué tipo de problemas pretende resolver. A partir de varias aproximaciones previas, se propone una definición pragmática de Movilización del Conocimiento, estableciendo que esta disciplina investiga los Sistemas de Movilización del Conocimiento, que son sistemas ubicuos, proactivos, declarativos, conscientes de contexto, integradores y concisos. En cierto modo, la Movilización del Conocimiento puede verse como un paradigma particular dentro de la Computación Ubicua, circunstancia que también se discute en este capítulo.

Para ilustrar el tipo de aplicaciones de las que se ocupa la Movilización del Conocimiento, se presentan tres ejemplos de casos de uso: el Médico Nómada, la Guía de Turismo de Bolsillo y el Guarda de Seguridad Ubicuo. El primero de estos escenarios es utilizado a lo largo de la tesis para crear los ejemplos que ilustran nuestras propuestas.

Adicionalmente, en este capítulo se estudian las áreas de investigación que están relacionadas con la Movilización del Conocimiento. Las disciplinas relacionadas se clasifican en cuatro niveles (datos, información, conocimiento y aplicaciones), que se corresponden a distintos grados de elaboración del conocimiento gestionado. Asimismo, además de estas tecnologías y herramientas de soporte, se analizan algunas investigaciones recientes que proponen soluciones integrales al desarrollo de sistemas móviles inteligentes, en la misma línea de la Movilización del Conocimiento.

Capítulo 3. Una Arquitectura para Movilización del Conocimiento

En el capítulo 3 se presenta nuestra propuesta de arquitectura para sistemas de Movilización del Conocimiento. El objetivo de esta arquitectura es ofrecer un esquema general que facilite el diseño durante el ciclo de desarrollo del software para Movilización del Conocimiento.

En primer lugar, en este capítulo se estudia cómo se organizan los elementos que participan en un sistema de Movilización para conseguir distribuir efectivamente el conocimiento recuperado de grandes volúmenes de información. A partir de estas ideas iniciales, se introduce la arquitectura para Movilización del Conocimiento, que describe las entidades, dependencias y esquemas de comunicación entre los componentes distribuidos. La descripción de la arquitectura ha sido creada con AML, un lenguaje semiformal de modelado de software para especificación de sistemas basados en agentes.

La arquitectura propuesta es general y puede adaptarse a diferentes dominios de aplicación. En la última sección del capítulo se estudian tres materializaciones de la arquitectura, concretamente, las adaptaciones multiagente, de espacio de tuplas y cliente-servidor del esquema abstracto. Además, se recomiendan varias tecnologías (la mayoría de ellas, surgidas en el ámbito de la Web Semántica) que pueden aplicarse para implementar las distintas instanciaciones de la arquitectura.

Capítulo 4. Un Modelo de Representación dependiente de Contexto para Movilización del Conocimiento

El capítulo 4 detalla nuestra propuesta de modelo de representación dependiente de contexto para Movilización del Conocimiento. Este modelo resuelve el problema de la sobrecarga de información mediante la creación de una base de conocimiento que relaciona explícitamente descripciones de contexto con el conocimiento del dominio que es significativo en dicho contexto.

El modelo de representación ha sido formulado como un patrón de diseño que permite crear ontologías de significancia en OWL, es decir, ontologías que caracterizan qué información del dominio debe ser considerada en cada escenario. En este capítulo se presenta la descripción formal del patrón de diseño (expresada en Lógica de Descripciones), así como un procedimiento para determinar el conocimiento relevante según contexto. Además, se demuestra que este algoritmo de inferencia es decidible y completo, y se estudian algunas propiedades adicionales del modelo (complejidad compu-

tacional, modularidad, etc.). Como contribución adicional, también se ofrecen una herramienta visual y una interfaz de programación en Java que han sido implementadas para crear y utilizar el modelo de representación.

En la última parte de este capítulo, se describe una extensión del patrón de diseño para la creación de ontologías de significancia difusas. Las ontologías de significancia difusas extienden la versión no difusa, permitiendo la representación de contextos y dominios imprecisos, así como la cuantificación de la importancia de las relaciones de relevancia. De la misma forma que en el caso no difuso, se propone un algoritmo de inferencia y se estudia la completitud, decidibilidad y complejidad del razonamiento con las ontologías resultantes de aplicar el patrón.

Capítulo 5. Una Aplicación de Movilización del Conocimiento: el Sistema IASO

El capítulo 5 presenta el diseño y la implementación de IASO, un sistema de Movilización del Conocimiento que resuelve el caso de uso del Médico Nómada descrito en el Capítulo 2. IASO distribuye resúmenes de historias clínicas a los dispositivos móviles de los usuarios del sistema; los contenidos de estos resúmenes se seleccionan automáticamente dependiendo de la situación del paciente que se está atendiendo.

IASO está basado en la arquitectura del capítulo 3, mientras que la base de conocimiento que soporta al sistema ha sido creadad con el patrón de diseño explicado en el capítulo 4. El sistema ha sido diseñado sobre el sistema de información del Hospital Clínico San Cecilio de Granada, una aplicación real que es utilizada para gestionar las historias clínicas en este hospital, y que en nuestra aplicación actúa como un repositorio que almacena los datos de los pacientes.

El diseño de IASO sigue las especificaciones de la adaptación cliente-servidor de la arquitectura general. Los dos principales actores del sistema son el cliente IASO y el servidor IASO. El cliente se corresponde con el dispo-

sitivo móvil del usuario, que puede utilizar un navegador web para consultar información (significativa) del sistema de información del hospital. La consulta incluye la descripción de la situación del paciente, que se construye con un vocabulario formal prestablecido. El servidor es responsable de procesar las consultas de los usuarios, inferir qué información es relevante y transmitirla al cliente en un formato apropiado. Para ello, utiliza la ontología de significancia que sirve de soporte al sistema IASO, que ha sido construida a partir de un vocabulario de contexto (basado en la ontología médica Galen) y un modelo de dominio (que representa de forma abstracta los registros que están almacenados en la base de datos del hospital).

Este diseño ha sido implementado en el prototipo IASO, que también se presenta en este capítulo. Este prototipo utiliza una versión reducida de las ontologías y las bases de datos necesarias para el sistema. No obstante, sirve para demostrar que las contribuciones de la tesis son apropiadas de cara al desarrollo de sistemas de Movilización del Conocimiento efectivos, y que pueden extenderse para implementar aplicaciones en otros dominios.

Capítulo 6. Conclusiones y Trabajo Futuro

El último capítulo de esta tesis doctoral resume las propuestas presentadas y concluye que las mismas alcanzan los objetivos planteados en el capítulo preliminar. La tesis finaliza planteando algunas direcciones interesantes de trabajo futuro.

Apéndice A. Ontologías y Lógica de Descripciones

En este primer apéndice se estudian las ontologías como formalismo de representación y su relación con la Lógica de Descripciones, una familia de lógicas orientadas a la codificación de conocimiento estructurado. En primer lugar, se presentan los aspectos básicos de la ontologías, seguidos de una introducción a la representación de conocimiento y al razonamiento con Lógicas de Descripciones. Este apéndice se centra en $\mathcal{ALC}$, la Lógica de Descrip-

ciones principalmente usada en la tesis. El apéndice termina con una breve descripción de las Lógicas de Descripciones Difusas.

Apéndice B. La Web Semántica

El apéndice B ofrece una introducción a la Web Semántica. En él se describen los objetivos principales de la iniciativa de la Web Semántica, así como la estructura general de la Web Semántica en relación a la Web actual. También se estudian los lenguajes RDF y OWL, los estándares para representación de conocimiento en Web Semántica, y se presentan varias tecnologías de Web Semántica utilizadas en la tesis, como son los editores de ontologías, los motores de inferencia, las interfaces de programación y las herramientas para publicación de bases de datos relacionales. Además, se incluye una revisión de los estándares para Servicios Web Semánticos, la extensión de los Servicios Web en el ámbito de la Web Semántica. El apéndice termina apuntando algunas líneas de trabajo actuales y futuras de la Web Semántica.

Contribuciones

El objetivo general de esta tesis doctoral es la investigación de teorías, técnicas y herramientas que proporcionen soluciones a los problemas que plantea la gestión y utilización de conocimiento extraído de fuentes de información de gran tamaño, distribuidas y heterogéneas en sistemas móviles. En concreto, se han estudiado dos problemas que plantea la *Movilización del Conocimiento*: (i) el acceso y transmisión del conocimiento disponible; (ii) la sobrecarga de información.

La principal conclusión de la tesis es que las contribuciones originales presentadas (la arquitectura para Sistemas de Movilización del Conocimiento y el modelo de representación contextual) satisfacen los subobjetivos del trabajo, y por extensión, este objetivo general. Así ha quedado patente en el diseño y la implementación del sistema IASO, una aplicación de Movilización

del Conocimiento en el ámbito de la salud que ha sido desarrollada a partir de estas dos aportaciones.

Además de estas contribuciones concretas, debemos destacar el carácter innovador de esta tesis doctoral, que explora un área de investigación relativamente nueva. Esto ha hecho que se que se hayan descubierto diferentes campos de aplicación y se planteen numerosas direcciones de trabajo futuro, lo cual es otra de las aportaciones más importantes de esta investigación. Entre otras líneas de trabajo futuro, destacamos la utilización de la arquitectura y el modelo de representación en otras áreas de aplicación para la implementación de sistemas de Movilización del Conocimiento prácticos y funcionales.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Antecedents

Knowledge Engineering is the subarea of Artificial Intelligence that studies how to represent both the concrete and abstract aspects of an application domain in a model that can be used by an automatic procedure to solve a complex problem. Knowledge Engineering is based on classical fields such as Ontology, which studies how the entities of a domain are classified, and Logic, which studies formal languages and inference rules for representation models [238].

The formal models studied by Knowledge Engineering are the core support of Intelligent Systems. An agent that handles more knowledge is more *intelligent*, and consequently, able to better resolve problems. A Knowledge Based System (KBS) is a software system that manages and applies represented knowledge to solve complex decision problems [89]. KBSs provide support for decision-making processes, and supply the right person with the right information at the right time [151].

Although this definition is still valid, current KBSs face new challenges. Nowadays, it is assumed that the "right person" may be located just about

anywhere, and that the "right time" is any moment at all. In the most extreme case, this means that KBSs must provide support for mobility. This signifies that they must be remotely accessible by means of a mobile device and a wireless communication network. Likewise, thanks to the Internet, users are now able to access to large quantities of useful information that was previously unavailable. Hence, the "right information" is obtained by integrating data retrieved from several sources, which may be large and distributed, as well as having different formats and semantics.

Modern KBSs are expected to provide nomadic with support through the use of mobile technologies and the incorporation of all available information sources. This allows the implementation of new functionalities, but also poses significant challenges to developers. Most of these difficulties stem from the computational limitations of mobile devices, e.g. reduced screen size, unreliability and limited bandwidth of mobile communication networks, battery duration, and the heterogeneity of development platforms and devices.

Some of these difficulties can be solved by improving mobile technology. For instance, recent mobile networks guarantee Mb/s transfer speeds. Nevertheless, other problems are intrinsic to mobile KBSs, and though they may gradually lessen as technology evolves, they still require computational solutions. Knowledge must be retrieved from large, distributed and heterogeneous information sources, and must be delivered to the mobile devices of decision makers. Mobile systems must be context-aware. That means that they must adapt their behavior to the user's environment, and must avoid information overload (i.e. overwhelming users with more data than they can digest). This requires representation models and software architectures that can extract, represent, integrate, and present this knowledge properly.

One of the focuses of Distributed Artificial Intelligence has always been the development of intelligent distributed systems. Distributed Artificial Intelligence proposes computational paradigms and technologies for the design and implementation of distributed KBSs. For instance, the multi-agent paradigm, or more recently, Web-based technologies, have defined organi-

zational patterns, communication protocols, and technologies to implement these systems. Furthermore, Distributed Artificial Intelligence has incorporated contributions from Mobile Computing (and vice versa), thus giving rise to Ambient Intelligence.

New knowledge representation and acquisition models are necessary to integrate distributed, heterogeneous information sources since more traditional techniques have shown themselves to be incapable of accomplishing this task. In this sense, one of the goals of Knowledge Engineering is to create knowledge models and methods that support the integration of information. For this reason, ontologies (a knowledge representation formalism that promotes the integration and re-use of previous knowledge) have become increasingly popular in recent years.

As often stated, the most complete (as well as heterogeneous) information source is the Web. Therefore, Web resources must be incorporated into KBSs. The Semantic Web is a recent initiative that aims at automatically searching, retrieving, and integrating Web information by endowing Web contents with semantics. The Semantic Web uses premises from Knowledge Engineering, since document metadata is represented by means of formal ontologies.

Despite such contributions, mobility and information integration problems in KBSs have not as yet been satisfactorily resolved. At the present time, mobile communication technologies are in a state of constant evolution. Although this is an exciting and dynamic situation, it also means that it is extremely difficult to establish a lasting implementation framework. The design of a knowledge-intensive mobile application is currently more of an art than a science, since well-established software engineering methodologies for desktop systems are not directly applicable to mobile ones. What is urgently needed are new software design guidelines such as architectural patterns. Regarding knowledge models, there are very few formalisms that solve knowledge delivery requirements in intelligent mobile systems. Ontologies are a promising approach, and Semantic Web proposals can be used to handle representation issues. However, this is a new research field that

needs to be further developed.

Knowledge Mobilization is a proposal that combines some of the approaches mentioned above (Distributed Artificial Intelligence, Ontologies, Semantic Web) with Soft Computing contributions (mainly Fuzzy Logic) in order to offer integral solutions for the difficulties that arise in mobile KBSs. Knowledge Mobilization provides a common framework, both from a conceptual as well as a practical perspective, for the development of mobile intelligent systems.

This research is on Knowledge Mobilization, and also pertains to Knowledge Engineering and Distributed Artificial Intelligence. With a view to dealing with the problems inherent in Knowledge Mobilization (limitations of mobile devices, integration of heterogeneous sources, context-awareness, information overload, etc.), we studied the architectures and models required to manage and distribute the knowledge obtained from large, heterogeneous information sources to nomadic users. The remainder of this thesis gives a detailed description of our research and the results obtained. We discuss the rationale underlying our proposals, as well as their advantages over previous work.

1.2 Objectives

The general objective of this research work was to find computational solutions for the problems of knowledge distribution and information overload in Knowledge Mobilization systems. More precisely, we focus on the study and creation of software architectures and formal representation models capable of overcoming these difficulties. Such architectures and models must be able to support the effective delivery of the knowledge obtained from large and heterogeneous information sources to nomadic users.

Accordingly, the objectives of the thesis are the following:

1. To review and analyze the state of the art in Knowledge Mobilization and related research areas.

    a) To examine the problems inherent in Knowledge Mobilization and related research areas;

    b) To analyze the benefits provided by Knowledge Mobilization systems in different application domains;

    c) To review relevant contributions from areas such as Mobile Computing, Distributed Artificial Intelligence, Ubiquitous Computing, the Semantic Web, and Soft Computing.

2. To propose an abstract architecture to support the design of Knowledge Mobilization systems.

    a) To provide an overview of the structure of a general Knowledge Mobilization system;

    b) To facilitate the development process of Knowledge Mobilization Systems by providing the developer with a global and customizable schema that can be specialized to meet the requirements of each concrete application;

    c) To discover which technologies are most suitable for the implementation of a Knowledge Mobilization system in each specific scenario, and how they should be integrated to achieve optimal results.

3. To create a context-aware knowledge representation model for Knowledge Mobilization.

4. To develop a proof-of-concept Knowledge Mobilization system that demonstrates the validity of our research.

1.3 Methodology

Constructivism is a methodology that solves problems by creating models, methods, diagrams, plans, organizations, etc. [135]. It is mostly used in Design Sciences. Design Sciences are applied sciences that have both a descriptive component (i.e. they can describe reality) and a prospective component (i.e. they can change reality) [188]. For example, if target B must be achieved and the current situation is A, Design Sciences study which artifact X needs to be designed and built in order to evolve from A to B. In the same regard, [125] explains the differences between constructive, nomothetic, and idiographic methods. The objective of constructive methods is the creation of new artifacts, in contrast to idiographic and nomothetic methods, which study real phenomena, and social relations and behaviors, respectively.

Information Systems (and by extension, Knowledge-Based Systems) are Design Sciences [126, 169]. Therefore, research in Information Systems consists of creating *meta-artifacts* that are useful for the development of concrete artifacts [127]: languages, models, methods, frameworks, software packages, environments, etc. This aspect is highlighted in [114], which distinguishes between routine design and research. Research must inevitably try to solve new problems ('wicked problems') and/or apply innovative procedures, whereas routine design uses well-established practices and techniques to solve known problems.

This author describes seven features that are present in most Design Science research work, namely, artifact design, relevance of the problem, evaluation of the design, contributions of the research, rigor of the research, design as a search process, and communication of the results. In [193], an schema to achieve rigor in the research process is proposed. This procedure consists of the following steps: (i) elaboration of a conceptual framework; (ii) description of the architecture of the system; (iii) analysis and design of the system; (iv) implementation of a prototype; (v) observation and evaluation of the solution. This approach is known as *Development as a Research Method* because it reproduces the steps of a development process.

Information Systems is a Design Science, but it is also a Behavioral Science since it studies the behavior and utility of existing systems [114]. Generally speaking, because of its multidisciplinary nature, Information Systems research also presupposes studying the state of the art in related fields.

Our research follows the principles of constructive methodology. We targeted a general problem, in this case, Knowledge Mobilization, and we created meta-artifacts, namely, an architecture and representation model capable of facilitating the development of Knowledge Mobilization systems. The validity of the meta-artifacts was shown by implementing a software prototype that provides a solution for a specific Knowledge Mobilization problem (Nomadic Healthcare). We also provided other supporting tools, such as the API and the visual tool to manage and create ontologies within the framework of our knowledge representation model.

The structure of our research study follows the iterative process proposed in [193]. First, we describe the conceptual framework of Knowledge Mobilization, i.e. the nature of Knowledge Mobilization and the properties of Knowledge Mobilization systems. Then, we present a general architecture of Knowledge Mobilization systems. The architecture is described in terms of a semi-formal language, which facilitates its interpretation and reutilization by different developers. Additionally, we created a second meta-artifact, namely, a model to represent knowledge in Knowledge Mobilization systems. Interestingly enough, reasoning with this model proved to be complete. This was possible since it is expressed in a formal language. The design of the resulting software prototype is based on the architecture, and relies on the knowledge model. This prototype satisfactorily solves a crucial Knowledge Mobilization problem. Consequently, it validates both the architecture and the model created, and shows that the results obtained in this research can be extended to other application domains.

1.4 Thesis structure

Throughout the description of our research, we clearly distinguish between original contributions, i.e. new proposals formulated in this thesis, and the contributions of others in related disciplines.

Chapters 2 to 5 present the original contributions of this thesis, namely, a review of Knowledge Mobilization, an abstract architecture for Knowledge Mobilization Systems, a context-aware knowledge representation model, and the implementation of a Knowledge Mobilization application. Appendices A and B presents an overview, respectively, of related topics such as knowledge representation with Description Logics ontologies and the Semantic Web.

Chapter 2 offers a panorama of Knowledge Mobilization and the problems that it targets. We give a pragmatic definition of Knowledge Mobilization and study the features that Knowledge Mobilization systems should offer. In order to illustrate the benefits of Knowledge Mobilization, we provide three examples of Knowledge Mobilization use cases. Additionally, we review other research areas and related contributions that offer integral solutions for intelligent mobile systems development.

Chapter 3 presents our proposal for a new architecture for Knowledge Mobilization systems. The chapter studies how the entities participating in Knowledge Mobilization systems should be organized in order to achieve the effective delivery of knowledge retrieved from large information sources. Accordingly, we introduce an abstract architecture for Knowledge Mobilization systems, consisting of entities, dependencies, and communication schemas between distributed components. This architecture is general and can be adapted to different application domains. It is expressed in the semi-formal language AML, a language for the specification of agent-based systems. Moreover, in this chapter we recommend various technologies that can be applied to implement different adaptations of this architecture.

Chapter 4 describes our proposal of a context-aware knowledge representation model for Knowledge Mobilization. This model is a design pattern

for the creation of OWL significance ontologies, i.e. ontologies that specify which information from the domain ought to be considered in each use scenario. The model solves the problem of information overload by creating a knowledge base that explicitly connects context descriptions with the domain expressions significant in each context. The formal description of the pattern is presented in this chapter as well as a reasoning algorithm to infer relevant knowledge. We demonstrate that the reasoning algorithm is complete, and we also study other features of the model (complexity, modularity, etc.). Furthermore, in order to facilitate the creation and use of significance-aware ontologies, a visual tool and a Java API were implemented. In the last part of this chapter, we describe an extension of the design pattern, which contemplates the creation of fuzzy significance ontologies. Fuzzy significance ontologies improve the crisp approach by allowing the representation of imprecise contexts and domains, and the quantification of the importance of relevance relations.

Chapter 5 presents the design and implementation of IASO, a Knowledge Mobilization system that solves the Nomadic Healthcare use case studied in Chapter 2. Thus, IASO provides mobile doctors with summaries of clinical histories, whose contents depend on the situation of the patient who is being attended. The design of IASO is based on the architecture described in Chapter 3, whereas the knowledge base that supports the system was created with the design pattern explained in Chapter 4. IASO shows that the results of our research led to the development of a successful Knowledge Mobilization system.

The last chapter summarizes the new proposals presented and highlights their contributions to Knowledge Mobilization. We analyze the results of this work in accordance with the objectives established in this introductory chapter. Finally, the thesis concludes with plans for future research work.

CHAPTER 2

Knowledge Mobilization

In this chapter, Knowledge Mobilization is defined in detail on the basis of earlier contributions in the literature. This chapter also provides a pragmatic description of the problems faced by Knowledge Mobilization as well as the characteristic features of Knowledge Mobilization systems. Given the close relation between Knowledge Mobilization and other areas of Information Systems and Artificial Intelligence, relevant technologies, methodologies and paradigms that are used to support Knowledge Mobilization are overviewed. Finally, some integral approaches to the development of Intelligent Mobile and Ubiquitous Knowledge-Based Systems are studied.

2.1 Definition

Knowledge Mobilization (KMob[1]) comprises all the efforts aimed to take advantage of the features provided by new mobile networks and devices in order to improve Knowledge Management (KM) procedures. KMob research faces certain problems which tend to arise when classical Knowledge-Based

[1]From this point on we will note Knowledge Mobilization as KMob, in contrast with Knowledge Management, which is usually shortened as KM. This abbreviation if firstly proposed in this work.

Systems (KBSs) are applied in current organizations, and proposes mobile technologies to overcome them. As a result, KMob is closely related to other topics such as Knowledge Engineering, Artificial Intelligence and Mobile Computing.

This notion of KMob appeared for the first time in [138]. In this work, Keen and Mackintosh present KMob as one of the three dimensions of modern companies (besides logistics and customer relationship) which can be improved by using mobile technologies. The *mobilization of knowledge* consists of making "knowledge available for real-time use in a form which is adapted to the context of use and to the needs and cognitive profile of the user". In other words, KMob should furnish managers with suitable knowledge in order to support them in decision-making processes, wherever they are located. Accordingly, KMob would provide the means to accomplish requirements of current organizations resulting from information immediacy and user mobility needs.

The authors make the following distinction between: (i) *conveniences*, namely, new features which are truly useful for the companies; (ii) *freedoms*, or new services which "change the structure of everyday life". In this sense, KMob is regarded as being a paradigm shift from traditional supply-center KM conveniences to KMob demand-driven freedoms. By providing (better) support to (better) KM practices and systems, KMob would help to diminish KM failure, which is a frequent issue caused by both people (employees do not share or update knowledge) and software (systems do not represent or distribute knowledge accurately).

Carlsson follows this approach and stresses that KMob systems must be intended to identify, delimit, and provide solutions to fulfill current KBS requirements [61]. The author defines four main tasks to be solved by KMob: creation of knowledge, activation of latent knowledge, retrieval of hidden knowledge, and delivery of knowledge. Paraphrasing F. Braudel[2], this will

[2]Fernand Braudel is one of the most important historians in 20th century. He is the author of *Civilization and Capitalism: 15th-18th Century*, a broad study of the economy in the pre-industrial era which analyzes the impact of economic events on everyday social behavior.

result in mobile value services and applications that will “expand the limits of the possible in the structures of everyday life”.

Recalling the definition of KBS in the introductory chapter –“to provide the right information to the right people in the right time, in such a way that it is useful in the decision-making process” [151]–, it should be noted that KMob shares with traditional KBSs time, place, and person requirements. The main contribution of KMob is that it is particularly concerned about the complex environments where decisions are made in current organizations, and proposes mobile technologies as a valuable tool to cope with them. Besides, as mentioned, real time, context, and a priori information are important to discriminate which information is significant to the user.

In addition, Carlsson points out a plethora of theories and technologies, most of them from Intelligent Systems and Soft Computing areas, which could be used to carry out each one of these KMob tasks: Semantic Web, Ontologies and Fuzzy Logic for building knowledge; Multicriteria Optimization, Evolutionary Computing and Simulation for activating latent knowledge; Data and Text Mining and Text Summarization for finding hidden knowledge; and Multi-Agent Systems for effective delivery of knowledge to ubiquitous users.

Our approximation is based on the proposal by Carlsson, concentrating on further studies on the computational difficulties that the implementation of KMob poses. From a pragmatic point of view, we believe that KMob addresses the challenge of building Knowledge Mobilization Systems, which are:

- *Ubiquitous*. KMob systems offer services that can be accessed from anywhere, at anytime, and using any device, particularly mobile ones, which have limited computing capabilities.
- *Proactive*. KMob systems are expected to discover what information is needed by the users and to act consequently.

- *Declarative*. KMob systems transform users' questions, which may be expressed in natural language, to queries that the resolving subsystem can understand.

- *Context-aware*. KMob systems take into account information about users' context and incorporate it to the decision process. Context knowledge may be used to adapt system behavior automatically depending on the current scenario and the ongoing activity.

- *Integrative*. Several and heterogeneous information sources, technologies and devices are meant to participate in KMob systems.

- *Concise*. KMob systems must summarize gathered data and tailor it to user needs and device capabilities.

- Finally, since KMob systems are intended to be a key tool to manage knowledge inside organizations, they must comply with common requirements of corporative software, such as *reliability*, *extensibility*, *security*, *easy maintenance*, etc.

We must point out that though these three definitions state KMob applications as targeted to assist expert decision, its contributions are suitable to be extended to less specialized areas. KMob also provides a proper support for developing services which offer advices or suggestions to mobile users facing non-critical tasks. For instance, let us imagine a shopping guide software which schedules in the customer's mobile phone a shopping spree to the local boutiques according to his location and preferences. Accuracy of results is (probably) not as vital as it would be in a corporate KBS, but needs of interoperation, integration of information sources, summarization of results, or use of context data may be even harder to resolve.

2.2 KMob and Ubiquitous Computing

KMob is inevitably related to Ubiquitous Computing (UC) and, to some extent, KMob can be regarded as a particular paradigm in this area. UC, often named alternatively Pervasive Computing (PC), was introduced in 1991 by Weiser, who envisioned a world where a multitude of computational objects (software and hardware: sensors, actuators, communication networks, portable devices, etc.), which are part of the ambience, communicate and interact with each other in order to carry out different jobs to help humans in daily activities [258]. Nowadays, UC systems obtain data from remote tiny sensors, transmit information using wireless communications, retrieve information from the Web, use Internet protocols, etc., aimed at automating tasks in different everyday life situations.

UC is regarded as the result of the evolution of Distributed Computing [226]. From early personal networks to current high-scale massive internets, problems have arisen (e.g. reliability, interoperability, or security) and solutions have been developed (e.g. routing algorithms, standard protocols, or encryption techniques) with a view to making remote communication a freedom, using the terminology of Keen and Mackintosh. A major step in this progress was the advent of mobile systems, which allowed voice and data networks to be accessed from anywhere at anytime. Mobile Computing poses new challenges to Distributed Computing, both hardware and software, while it also offers new opportunities: ubiquitous access to data, wireless communication, use of portable devices, etc.

Relying on Distributed and Mobile Computing, several UC subareas and related topics have been developed, each with its own particular focus. For instance, Smart Rooms [204], a combination of Domotics and Image Recognition to support an inhabitant's activities with computational facilities, are defined similarly to UC. Computationally-enhanced environments are also studied in Ambient Intelligence, which claims that Artificial Intelligence techniques are the key to building effective pervasive applications [214]. UC overlaps considerably with Embedded Computing [260]. Embedded devices

are simple specific-purpose controllers, which are encapsulated inside the larger machinery that they manage. When devices are expected to be carried by the users, ideally indistinguishable from their clothes, systems are usually included in Wearable Computing [220].

KMob, in turn, can be regarded as the UC subarea interested in delivering refined information (i.e., knowledge) to specialized nomadic users. It shares with the aforementioned ones several issues to be resolved, e.g. mobile networking, information delivery to nomadic users, integration of devices, context-awareness, adaptive behavior, transparency, and scalability. Artificial Intelligence theories and techniques are expected to play a key role, a principle which is shared with Ambient Intelligence. On the other hand, KMob has specific challenges that require particular solutions, namely, knowledge acquisition, representation, and presentation. KMob involves providing users with more elaborated information than general pervasive applications, which usually manage lower-level data collected from environmental sensors. As a result, an expressive formalism must be used. In addition, since devices with limited computational capabilities are used, representation formalisms must generate tractable knowledge bases. Likewise, person to machine interaction is more frequent in KMob, since more critical decisions must be made based on the available information. Thus, interfacing with the user is a priority.

2.3 Use cases

In this section, we present three scenarios that exemplify the kinds of applications that can best take advantage of KMob proposals. We follow the schema of the W3C Semantic Web Activity[3], which describes use cases for Semantic Web technologies. In this context, a use case is an example of a prototype system which, event though it is not currently being used by an organization, demonstrates the benefits of applying Semantic Web tools. In this section, our use cases show the possibilities of KMob technologies. The

[3] http://www.w3.org/2001/sw/sweo/public/UseCases/

first is the Nomadic Healthcare use case, which is exemplified in Chapter 4, and implemented in the application described in Chapter 5.

Nomadic Healthcare use case

The problem. Dr. Greg is a physician who needs to consult the clinical data of a patient in order to prescribe the best treatment. If the healthcare service occurs inside the hospital, Dr. Greg will be allowed to access the Hospital Information System (HIS) and to retrieve all of the patient's Electronic Health Records (EHRs). Since he has enough time and knowledge, he will be able to rule out all the useless pieces of information and will only obtain those that he is interested in.

In the event that Dr. Greg is in an emergency-assistance ambulance unit, and he is caring for a patient injured in a car accident, it will be helpful to be able to access some data about the patient's clinical history. For instance, data concerning the patient's adverse drug events (ADEs) may have been recorded. Nevertheless it is improbable that the doctor will be able to access HIS data from outside the hospital (much less by using a portable device such as that available in emergency healthcare units). Even if it were possible, the doctor would not have enough time to review all the stored electronic records.

The solution. In such a situation, a brief report including those pieces of the patient's clinical data which ought to be considered would be very valuable. The clinical procedure to be carried out would determine which data should be part of this summary. For example, if the patient is unconscious and has a hemorrhagic laceration, information regarding whether he has an allergy to procaine (an anesthetic drug which reduces bleeding but is also often badly metabolized and triggers allergic reactions) should be taken into account, among other things. Thus, some kind of rule asserting that 'data about previous anesthetic drug reactions' ought to be considered when 'the patient has a penetrating wound' should be created in the knowledge base of the system. Following recommendations for clinical and ADE guidelines,

other similar rules can be included in the knowledge base in order to support the creation of context-dependant EHR summaries.

Assigning a priority to these links among patient states and registers of the HIS would also be desirable. For example, allergic reactions to anesthetic drugs are more important than blood-borne diseases and should be presented firstly to the doctor since avoiding an anaphylactic shock is a major priority, and medical protocols prevent the doctor from being in contact with patient's blood. Ranking the relevance relations would allow system responses to be sorted by precedence and a threshold to be fixed to filter information.

Benefits of using KMob contributions. This example present various of the characteristic features of KMob systems (as they have been defined in this chapter). For instance, the system must be context-aware, i.e. it needs to detect which is the situation of the patient and behave accordingly. Likewise, the eventual Nomadic Healthcare system must be concise, since doctors are provided with summaries only including significant information about the patient. Necessarily, the system will integrate different technologies (mobile platforms, communication protocols, etc.) and information sources (the HIS, the vocabulary to describe patient situations, etc.). All these tasks can be faced by applying KMob technologies and tools.

Portable Tourism Guide

The problem. Travelling to a strange city is often an unnerving experience, whether the trip is for business or pleasure. If the traveller wants to know how to get from the airport to his hotel, he has to check the timetables and prices of the airport buses, trains and taxis, calculate the estimated time and cost of the trip, and decide which means of transportation is best for him. A similar situation occurs when the traveler decides to go sightseeing and visit one of the tourist spots in the city. He must retrieve information from different providers, and then make his choice accordingly.

Moreover, the traveler may be interested in checking out the opinions

of other people who have visited the city. Webs offering reviews of movies, restaurants, cinemas, hotels, companies, etc. have proliferated over the past years. Providing the user with this information in addition to the official data is very advantageous, especially if he is visiting a city that he is unfamiliar with.

None of these tasks is easily carried out by a mobile device operating with current technologies. All this information is probably available on the Web and can be accessed, but it is difficult to integrate the different information sources. Furthermore, it does not make sense to present a huge volume of data to the user because he will not have the energy or the time to read pages and pages of results on his mobile device.

The solution. A possible solution to this problem is the publication of all the data in a format that can be automatically processed. This enables a software agent to collect and integrate information, which is subsequently delivered to users in a proper format. The information received is then filtered according to the user profile, which includes previous actions, preferences, location, etc. It would be also interesting to implement the means to automatically update this profile. For example, if the user changes location, the profile should be changed without the user having to explicitly specify his new geographic position. Moreover, the profile could be learnt as the user utilizes the system. For instance, if the user frequently consults sports results, the profile should be adapted so that it takes this preference into account.

Currently, there are various research initiatives aimed at describing published information with well-defined metadata. Actually, this is the objective of the Semantic Web, which can be applied to building advanced mobile systems. In our example, the information pertaining to public transportation, places, and users reviews should be published in the RDF language so that an automatic agent can automatically extract, integrate and consult the semantically-enriched data. Additionally, the user profile should be expressed in a logic-based language such as OWL. Thus, user preferences and available information could be matched, and only relevant information retrieved.

Benefits of using KMob contributions. The Semantic Web, which is one of enabling technologies of KMob, offers a framework for the formal description of information published in the Web. Resources with a well-defined meaning can be better located, integrated, and delivered. Soft Computing and Machine Learning techniques can be applied to adapt the behavior of applications without user intervention. KMob, as an integral approach, establishes how these technologies can be combined to develop an effective mobile application.

Ubiquitous Security Guard

The problem. Security in large installations is controlled by teams of guards that patrol the surveillance area. Guards can either be walking the beat or watching the cameras installed in the secured area. The guards can communicate by using radio transmitters. Usually, when a security alarm is triggered, guards follow a predefined protocol for each possible risk situation.

Mobile technology allow security systems to be improved by implementing software that provides guards with data, which may include any kind of multimedia content, independently of the location of the user. Moreover, mobile communications and devices can be used to better coordinate the response against a security alarm. Another benefit of mobile technologies is that they can be combined with intelligent procedures in such a way that the signals captured by ubiquitous devices (sensors, cameras, or other devices carried by the guards) can be interpreted, and a suitable plan can be automatically scheduled in real time.

Such an Ubiquitous Security Guard application must integrate different communication technologies, but even more important, it must rely on a sound knowledge model to detect security threats and generate reaction plans, as well as a suitable mechanism to coordinate the distributed components of the system. These components are both computational (sensors, cameras, etc.) and human (guards).

The solution. The software entities participating in the Ubiquitous Security Guard application must have a high degree of autonomy. At the same time, they must cooperate and coordinate their actions in order to react to and effectively deal with security threats. The design of such system can be based on the multi-agent paradigm, which establishes a framework for the design and implementation of distributed intelligent systems.

Additionally, the knowledge of the system can be represented if the form of ontologies. Maps of the secured area can be tagged with semantic metadata, which allow agents to precisely state the location of an object or an event. Likewise, ontologies that describe possible risk situations could be created. These knowledge models would be used by the reasoning procedures to deduce the protocol to be activated in a given situation, and to create a plan to deal with the incident.

Benefits of using KMob contributions. KMob proposes multi-agent systems to achieve effective intercommunication and interoperation between entities in a distributed system. In the same regard, ontologies are applied in KMob as a suitable formalism to create a language, understandable by a wide range of different agents. Moreover, KMob provides a framework that decouples the design and the implementation of an intelligent mobile system. Thus, the structure and behavior of the entities of the system can be abstractly described as participants of a KMob application, and then implemented using a suitable technology.

2.4 Methods, technologies and tools

In the previous section we have described three use cases that illustrate some of the benefits which KMob afford to nomadic knowledge workers. Now we shall point out some recent technologies which are crucial to the development of KMob systems.

As is well known, there are currently development platforms that allow *ad hoc* mobile solutions for concrete applications to be implemented, covering:

(i) software running on mobile devices (e.g. J2ME, .NET Compact Framework, Symbian OS programming toolkits); (ii) server logic (J2EE, .NET); (iii) communications (WAP, HTTP, SOAP/Web Services, etc). These technologies are the framework supporting development of mobile services. Building a KMob application from scratch with these technologies can be rewarding, but also frustrating and (even more important) costly, especially when maintenance is required or when there are new demands to fulfill. Participation of new devices, update of technologies, change of user roles, or need of incorporating new interaction patterns pose a great challenge to people in charge of the system because some parts of the system (if not all of it) will have to be redesigned and redeveloped from the basis.

Therefore we consider that a unified, extensible, and adaptable computational architecture for KMob applications must be defined. This architecture must identify the components of a KMob system, the relations between them, and the actions they can perform. Such an architecture would allow KMob systems to be described in terms of abstract modules and operations. This avoids the specification of the low-level aspects of mobilization, both hardware (which devices will be used) and software (which technologies will be applied to implement the system). The architecture will be conveniently implemented using different technologies depending on the system size, purpose and restrictions. Our proposal of an architecture and an associated supporting framework is extensively discussed in Chapter 3. The following sections review some of the existing methods, technologies and tools in different areas which KMob applications will rely on.

Figure 2.1 shows our vision of KMob and related topics. It depicts the research areas and technologies that we consider to be appropriate to make data develop into knowledge ready for mobilization in different application domains. Interestingly enough, we propose relying on Intelligent Systems (IS) solutions to meet the challenges of KMob challenges, in line with Carlsson's (as well as other UC researchers' [253, 257]) view. We focus on three main (non-disjoint) contributions of IS: Knowledge Representation (see Appendix A), Intelligent agents, and Soft Computing. Besides IS, other tech-

nologies can be used: mobile devices and networks, database and data warehouse systems, middlewares, etc.

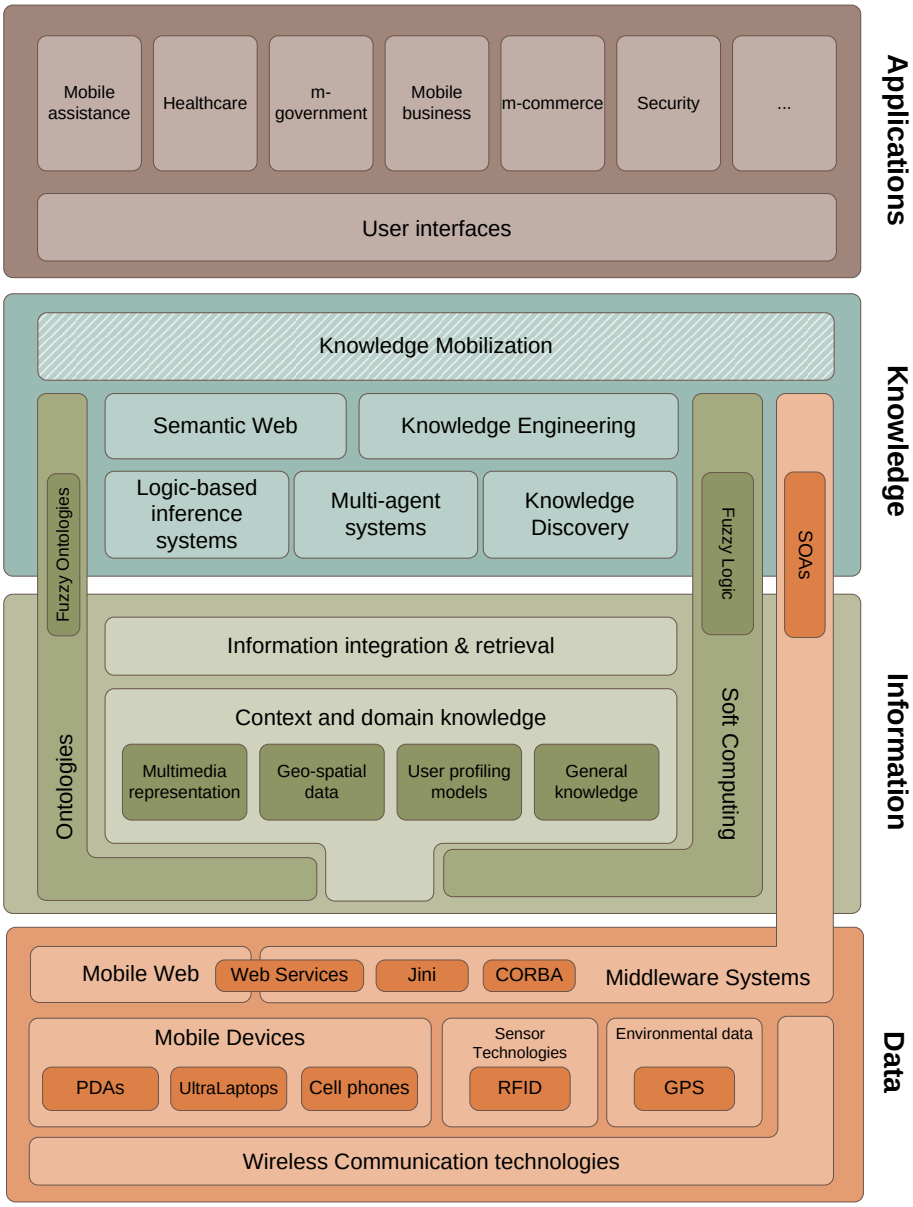
In this figure, four main layers are distinguished: data, information, knowledge and applications. This organization reflects a distinction usually stated in Information Systems theory which mirrors the process of rough data becoming useful knowledge [1]:

- data are syntactic patterns with no associated meaning, which are used in the initial step of a decision-making process;
- information is the result of interpreting data and providing it with meaning;
- knowledge is learned information which is conducive to inferring new knowledge to be used in decision-making.

In the case of the doctor in the emergency ambulance service consulting the HIS (Section 2.3), the patients' EHRs can be considered data, since they have little meaning; EHRs merely store figures, strings and perhaps images with the values of different healthcare parameters. If semantics are added to these parameters, they can be considered information; an ontology can be used to state that blood pressure is a clinical sign, or that anemia is a blood disorder. This ontology can be further enriched to have relate profiles, diseases and EHRs; it thus becomes knowledge useful for decision-making. For example, the ontology can tell us that anemia occurs when the hemoglobin level is below 13.0 g/dl, and that it is commonplace to have anemia after radio and chemotherapy treatments have been carried out.

Next we discuss relevant research related to topics mentioned in Figure 2.1.

Figure 2.1: Knowledge Mobilization related research areas



Data

Wireless broadband communication technologies and portable computational devices have emerged to become the enabling technology of new mobile services, such as those provided by KMob.

On the one hand, third-generation cell technologies (3G) are being currently deployed in the market, still dominated by GSM (2G), and new services, such as videoconference or MBMS (Multimedia Broadcast Multicast Services), are now being offered thanks to higher transfer speeds (up to 2Mbits/s versus 144Kbits/s) and more reliable connections. Regardless of these advantages, dissemination of 3G technologies has not been as fast and rewarding as expected. However new extensions to them are being introduced, namely, the 3.5G and 4G. Poole provides a comprehensive introduction to cellular communications, describing issues, solutions, and standards in [207].

Wireless network technologies are seen to complement or to build on current third-generation mobile technologies. In recent years, Bluetooth (for short-range *ad hoc* connections, i.e. WPANs) and Wi-Fi (for local IP-based networks, i.e. WLANs) have become the means to communicate small and mid-sized systems [153]. In fact, voice-over-IP (VoIP) services, which allow voice communications to be performed throughout IP networks, are regarded as a considerable threat to carrier operators, since they eliminate subscribers' dependence on cell infrastructure (at least within the last mile of the loop). Possibilities will be extended with the advent of WiMax, a successor of Wi-Fi which promises wider covertures and higher transfer speeds (up to 70 Mbits/s and 110 km, in the best of cases and not simultaneously). Both cell and WLAN/WPAN technologies will be fully interoperable in the near future (Mobile IP and IPv6 share this aim). The result will be that all significant data communication will work on IP protocols, which is usually known as an all-IP development.

On the other hand, as a result of technology convergence – mainly handheld devices (PDAs), cell phones, and laptop computers –, mobile devices

have evolved from simple voice-transmission terminals to smart computation gadgets equipped with 3G, Wi-Fi, GPS, video recorder, etc. which are able to fulfill complex tasks, such as browsing the Internet, e-mailing, running business software, etc. This trend will predictably continue in the near future [194], in consonance with the impressive penetration rates of these technologies. As a matter of fact, consultant firm Strategy Analytics estimates in its report for the last three months of 2007 that 332 million cellphones were shipped worldwide [174], an increment of 13% year-over-year, with strong demand in emerging markets like Africa, India and China, whereas Informa Telecoms & Media forecasts that 121 million converged devices will be sold by 2012 [128].

Software programmers can make the most of mobile technologies by using development platforms and APIs adapted to the flavor of the target device operating system. Currently, there are two main branches of mobile devices, differentiated by the operating system used: Symbian-based (mostly owned by Nokia, and the most widespread) and Windows Mobile-based (developed by Microsoft, and included since 2006 in Palm Treo Series). Each has its own programming platforms. An alternative is to use J2ME (Java2 Mobile Edition), which can be run on different OS. J2ME defines a set of profiles and configurations to adapt software to device capabilities, as well as additional pluggable interfaces for new specifications (JSRs).

Other minority development frameworks worth mentioning are those intended for Linux-based mobile devices: Maemo, for Nokia N770 and above; Android SDK, for Google's Android OS (not included in any device to this date); or Qtopia, provided by Trolltech (recently bought by Nokia) and used in some Motorola and Panasonic cell phones. It is also possible to develop software for (more) closed platforms like iPhone OS (non-web applications are officially supported only since February 2008) and RIM Blackberry (which provides a Java Development Kit). Finally, web development is an alternative to ensure portability: mobile phones can access the Web and invoke applications running on remote servers using HTTP. These applications can be enriched with recent web programming technologies such as AJAX or Flash,

which can be interpreted by most of current mobile browsers. Recently, platforms which split the execution of web applications between the client and the server have been proposed, namely, the Rich Internet Applications (RIAs) (e.g. Microsoft Silverlight and Adobe AIR).

Additional sensor systems may be involved in KMob applications. In this sense, there are two that have become increasingly popular in the last decade: GPS (Global Positioning System) and RFID (RadioFrecuency IDentification). GPS technology allows users to calculate the absolute position on Earth (latitude, longitude) of a GPS receiver by the interpolation of the signals transmitted from four (or more) satellites. RFID, in turn, is based on remote read and writing of tags storing identification data, a simple mechanism which may be used to track tagged objects during their lifetime. These can be used beside other environmental sensor in context-aware systems, i.e. systems that acquire, deliver, and process environment data in order to apply it to subsequently customize system behavior to the users' ambience.

In order to deal with this wide range of technologies and facilitate mobile application development, middlewares have been developed. In general, middlewares are programs which enable interoperation between application and system software. In our case, a KMob middleware is a set of (software) high-level primitives which cope with low-level aspects of mobilization (software and hardware) and embodies a reference framework for developers, whose objective is to hide device and communication details as much as possible, especially from users.

There are middleware platforms at different abstraction levels, ranging from standards aimed at interconnecting distributed components of a system (syntactic interoperability) to complex software able to automatically locate, invoke and coordinate system nodes (semantic interoperability). Some of the more interesting technologies currently used are: (i) RPC protocols, for remote procedure calling over TCP/IP; (ii) CORBA and its sucessor ICE, two specifications for remote procedure calling in object-oriented architectures; (iii) Jini, a Java technology for federated systems; (iv) SOAP, which together with WSDL and UDDI is the standard to implement Web Services, allowing

remote procedures to be requested through elaborated HTTP calls; (v) REST paradigm, which promotes the access to remote resources using *ad hoc* simple XML and HTTP messages. More abstract middlewares usually rely on these technologies, and these are described in Section 2.5.

Mobile Web is a related initiative: since it strives to make possible surfing the whole web using mobile devices, it is also suitable to allow interchange of information (via Web protocols) and the remote running of applications (via Web Services) in mobile systems without having to be concerned about all the details of the underlying technologies.

Information

Since applications (and, consequently, supporting middlewares) are expected to manage different and heterogeneous data sources, different data models must be used to endorse semantics to data. Among others, the information layer ought to consider multimedia (images, sounds, video streams), context data and user profiles, apart from general domain information.

Therefore, formal theories to represent sparse and heterogeneous information are required in KMob. This has been a long-established issue in Knowledge Engineering that nowadays is being tackled by using ontologies. Ontologies are a knowledge representation formalism which offer interesting features such as the standardization, sharing and reutilization of knowledge [64]. They are thus suitable for representing knowledge in KMob applications. Appendix A includes a detailed study of ontologies and the ontology language OWL.

The role of context knowledge in intelligent mobile systems is worth mentioning, since it may be used to automatically customize responses according to user circumstances and preferences [158], leading to the UC aspiration of a semi-invisible computer [258]. Summarization, as a form of personalization, is an especially desirable feature in KMob applications, where presentation of large volumes of data on mobile devices is critical and may result in information overload.

Accordingly, knowledge models can be roughly classified as application-specific or context-related, depending on the role that they play in a KMob application. The application-specific knowledge base contains expert information about the specific problem to be dealt with by KMob systems, whereas context knowledge is used to specify under which conditions subsets of the latter ought to be considered. Chapter 4 describes how context and specific-domain knowledge can be represented by using ontologies, and proposes a design pattern to create and reason with context-aware ontologies.

By information integration and retrieval, we mean that information could be automatically integrated and conveniently stored in a data warehouse to perform further knowledge discovery processes. Information may be tagged with terms of an ontology, which is known as ontological annotation, or alternatively, it may be used to build a new ontology, which is known as ontology learning. Both processes can be performed automatically with proper machine learning techniques (maybe based on Soft Computing and Fuzzy Logic).

It should be highlighted that the representation primitives of ontologies are crisp, that is, they are logic-based constructors which evaluate either to true or false. For instance, regarding concept inclusion – the basic inference procedure in DLs –, it can only be stated that a concept is or is not subsumed by another one. This can be a serious shortcoming when imprecise information must be represented in a knowledge base, something far from unusual in several domains. For that reason, extending ontologies (and consequently ontology languages) to handle imprecise and vague knowledge in KMob applications is a very promising research line. Fuzzy ontologies are an extension of classical ontologies that allow such knowledge to be represented [225].

Fuzzy extensions of DL also support enhanced information retrieval processes, for instance (partial) matching of user preferences against service capabilities or result rating based on different criterions. Such tasks would be difficult to implement using a crisp representation formalism. In this work (see Chapter 4), we propose a fuzzy extension to rank significance relations

in context-aware ontologies. This approach is based on previous studies on reducing reasoning within Fuzzy DLs to reasoning within Crisp DLs in order to make possible the use of currently available inference engines [39, 41] (see Appendix A.

Knowledge

Built upon the information management layer, IS technologies dig into information stores to extract valuable knowledge from less elaborated data. Furthermore, intelligent appliances can be used to deliver useful knowledge to ubiquitous users, providing them wherever they are with information generated at any point of the system using any device to make effective KMob.

Agents are one of the abstractions that have been most frequently used to describe and implement proactive intercommunication among modules in intelligent distributed systems. The Multi-Agent Systems (MAS) paradigm proposes a scenario where independent, goal-directed, and environment-aware units (the agents) become coordinated (by collaborating or competing) to accomplish complex tasks [259]. Adaptive learning may be applied to dynamically adjust agent behavior. In KMob, agents collect and distribute information across the distributed modules of the system. Some advantages of MAS are a solid conceptual grounding (they can be depicted using well-defined abstract entities and operations), encapsulation of their components (which hides agent policies and promotes scalability), communication facilities (high-level protocols are used), and parallel execution (resulting in better performance and robustness) [244].

From a practical perspective, the MAS paradigm eases system development as standards and frameworks to build them have been proposed. For instance, FIPA describes an extensive standard ranging from system architecture (FIPA Abstract Architecture) to agent communication (ACL, Agent Communication Language) [208]. Related approaches are KQML (Knowledge Query Manipulation Language) [93], a communication language, and MASIF [179], a collection of standards for agent interoperability. There are

also development platforms which implements these standard, e.g. JADE [23] (FIPA-compliant), JatLITE [129] (KQML- and FIPA-compliant) or Grasshopper [19] (MASIF- and FIPA-compliant). Extreme decoupling of processing modules is driving MAS towards peer-to-peer systems (P2P), i.e. highly distributed systems where central coordinators are reduced to the minimum.

Our vision of a KMob middleware is not very far away from these MAS platforms, although it should be located on a high abstraction level. In our opinion, developers should concentrate on knowledge management rather than on data communication details. Nevertheless, these platforms can provide a sound framework from which richer tools can be built, above all if we bear in mind that MAS research has faced mobility problems in preceding years. In the literature, MAS mobility has usually referred to the capacity of running agents to be transferred from one computer to other [68, 6], but the explosion of mobile computing has led to an increased focus on agents executing on mobile devices. As a result, some platforms as JADE-LEAP allow the deployment of agent software in J2ME-enabled mobile phones [25].

MAS are strongly related to Service Oriented Architectures (SOAs). SOAs, implemented with Web Services [75], have been used respectively as an alternative or a complement to other paradigms like MAS, and to other transport protocols like CORBA/ICE or RPC. Recently, the OSGi platform has been presented as a complement or a replacement for Web Services. OSGi is a Java-based framework which provides support to create communicative services that can be executed in different computational environments, including mobile devices [5]. However, understanding between pure agents platforms and services (whatever they are implemented) must be explicitly achieved, since different communication technologies are used. Ontologies, employed as intermediate terminologies, are expected to play a key role in solving this issue [112].

The Semantic Web is proposed, in addition to MAS, as the key technology for knowledge building in KMob. The Semantic Web (SW) is defined as “an extension of the current web where resources are described using logic-based languages in order to allow automatic processing” [27]. The aim of SW is

to overcome certain drawbacks of the current Web by providing mechanisms for the automation of document processing and the simplification of effective information recovery processes. Relying on metadata annotating resources, software agents are expected to search, locate, discover, or link documents better than today's lexical-search engines. Accordingly, SW researchers are very interested in formalisms for creating metadata to be associated to Web resources. Hence, ontologies play a fundamental role, since they are the main representation formalism in the core of the layered architecture of the SW.

The SW has contributed to Knowledge Engineering with languages (such as the standard Ontology Web Language OWL [176]); methodologies (for instance, METHONTOLOGY [71]); tools (parsers, editors, reasoners, APIs) [76]; or domain-specific large-scale ontologies (e.g. UMLS ontology [45]) (see Appendix A). These make up a suitable framework for building not only SW applications, but distributed KBS of any kind. Semantic Web Activity[4] within the World Wide Consortium (W3C) congregates several groups that are making a great effort to develop standards for the SW.

Fully-fledged semantic web pages are still far away from being common in the world outside research labs, but some of their contributions are valuable and stable enough to be incorporated in KMob systems. For our purpose, resources in KMob systems can be given semantics by using ontologies, and can be managed with Semantic Web technologies. A survey of work which shares this objective is presented in Ranganathan et al. [217]. This also presents some challenges which can be addressed using SW technologies, like the automatic coordination of actors in mobile interaction. Masuoka et al. tackle this problem, and suggests attaching descriptions to services in order to discover, communicate, and integrate customers and providers in pervasive environments [173]. More recently, Lassila also has underlined this difficulty, and has proposed the use of OWL ontologies to represent policies (roughly, contexts) and Semantic Web Services (an in-progress specification from the W3C) to achieve serendipitous device coalitions [157] (Semantic

[4] http://www.w3.org/2001/sw/

Web Services are studied in [43]). Examples of implementations that use Semantic Web technologies and agents to ensure interoperability in context-aware pervasive environments are given in Chen, Finin, and Joshi [66], who define an ontology (SOUPA, Standard Ontology for Ubiquitous and Pervasive Applications) for representing agent BDIs (beliefs, desires and intentions), user profiles, time, etc. and Soldatos et al. [236], who create a semantic directory of entities.

Applications

Applications are implemented on top of the enabling technologies of KMob, and will be supported by a KMob framework. As far as possible, these applications must keep the user unaware of the underlying complexity. Intuitive and user-friendly interfaces must be provided (the more adapted to user context, the better). There are various methodologies for improving multimodal/mobile user interface design [164], and more are expected to be investigated [187], given the fact that new input and output procedures (e.g. auditory, tactile, or motion-based) are being offered by mobile devices. Thus, best practices and proper techniques should be considered in order to make the use of KMob systems easier and to reduce the number of interaction problems and the length of the learning period. Usability methodologies allow the evaluation of user experiences, and anticipate the conditions of the interaction in future scenarios.

Regarding application domains, as presented in Section 2.3, health care applications are one of the most interesting domains where KMob can be applied since both the technical challenges and the social implications are relevant. KMob can be applied to decentralized and personalized health services, care of the disabled, hazardous drug control, food traceability, logistics, etc. Chapter 5 presents a system that provides nomadic physicians with summaries of a patient's previous diseases that directly affect diagnosis. It also provides advice regarding further clinical tests to be carried out.

2.5 Related work

In the previous section we discussed various contributions pertaining to different areas of Information Technology that will participate in KMob applications, remarking some useful references about each subject. In this section we describe integral approaches to the problem of knowledge management and delivery in mobile systems, that is, we review architectures, frameworks, platforms, APIs, etc. that are intended to support ubiquitous intelligent systems by integrating some of the aforementioned technologies. These range from pure UC applications to others closer to our vision of KMob, according to the differences remarked in Section 2.2.

Most works in the literature on pervasive systems tackle the problem of representing context information, which is one of the main objectives of UC and KMob. A fundamental contribution in that area is the Context Toolkit [80], a framework for rapid prototyping of context-aware applications. Apart from the framework (both conceptual and practical), a general definition of context and categories of context are examined in that paper. Tradeoffs of this approach are examined by Hong and Landay [115], who propose a higher-level infrastructure is suggested.

Similar ideas are explained in [111], which gives a description of a hardware implementation of a context-aware system for smart rooms based on CORBA communication. Several projects tackling the problem of context-awareness in smart rooms were unveiled in the early 2000s: Interactive Workspaces (at Stanford University) [132, 206], EasyLiving (Microsoft) [55], Aura (Carnegie Mellon) [98], Cooltown (HP) [145], BlueSpace (IBM) [156], and Oxygen (MIT) [78] are some examples. All these seminal works concentrate essentially on providing software platforms and on developing physical implementations of enhanced environments where a considerable amount of different and heterogeneous devices must be coordinated. In addition, Yau et al. propose RCMS (Reconfigurable Context-Sensitive Middleware) [261], a middleware which provides development and runtime support for applications that require context-awareness and spontaneous *ad hoc* communica-

tion.

As mentioned in Section 2.4, ontologies are being intensively used for specific-domain and context knowledge representation. They can be encoded in OWL or in other languages. Gaia, a middleware for mobile applications promoting the use of ontologies in the description of context predicates, is presented in [216]. In this platform, components are modeled as agents; the are communicated with CORBA; and their context is represented with DAML+OIL (a predecessor of OWL). Gaia has been extended to incorporate fuzzy, probabilistic and Bayesian formalisms to process uncertain facts about general context data [215]. In the same way, the Context Mediated Framework defines a platform based on a fuzzy model to pre-process inputs from crisp environmental sensors [149, 168]. These works are more oriented to classical UC than KMob. In spite of the fact that they provide rich mechanisms to represent context knowledge (mostly acquired from sensors), domain information is not expected to be as complex as in knowledge-intensive KMob applications.

Gu, Pung, and Zhang follow a similar approach to Gaia in SOCAM (Service-Oriented Context-Aware Middleware), although they use OWL and consider that a SOA is more suitable to communicate modules in that kind of systems [106]. Other works which use SOAs are [163], [212] and [144]. The first defines a middleware for Ambient Intelligence implemented with Web Services called WSAMI; the second establishes a similar architecture relying on MAS paradigm and OSGi; and the third proposes another middleware for UC applications based on SOA and Web Services with OWL ontologies being the means to manage context knowledge. In contrast to these contributions, the OWL Services Framework (OWL-SF) proposes a REST architecture and a supporting framework based on OWL for context representation and reasoning, and OMG Super Distributed Objects for sensor management, which provide support for UC intelligent applications [182].

CoBrA (Context Broker Architecture) is another infrastructure which also represents context using a RDF/OWL ontology [67], based on SOUPA [66]. A more specific proposal using ontologies is described in [154], where a rec-

ommender system for mobile users is developed on a multi-agent platform. Some of these platforms (Context Toolkit, Context Mediated Framework, CoBrA, and SOCAM) are reviewed in [233], where the STU21 framework, a proposal by the authors, is also described. It is interesting to mention the work in [7], which presents a fuzzy methodology to measure partial equivalences between situations (expressed using OWL ontologies) and to determine suitable action rules to be fired in pervasive applications.

Regarding multi-agent platforms, we have cited JADE (Java Agent DEvelopment), a framework for implementing FIPA-compliant multi-agent systems based on a peer to peer communication architecture [23]. JADE platform was adapted to mobile devices with the LEAP (Light Extensible Agent Platform) extension [25]. JADE/LEAP provides a functional but overly general infrastructure for working with high-level semantics and context information. Thus, it is not generally used directly when implementing such distributed KBSs.

Instead, new middlewares have been built over JADE/LEAP primitives. For example, that is the underlying technology of the approaches by Khedr and Karmouch [139] and Soldatos et al. [235]. The first proposes an infrastructure for developing context-aware applications (ACAI, Agent-based Context Aware Infrastructure) based on OWL ontologies for representing context and agents (implemented in JADE) for coordination and communication within the system. The second also defines a new middleware for pervasive and context-aware which relies on JADE, but concentrates more on system implementation and evaluation rather than establishing a model of the infrastructure. Nevertheless, a formal description of these authors' middleware has been recently sketched in the CHIL Reference Model Architecture for Multi-modal Perceptual Systems [251]. Agents are also used to communicate components in the ECORA (Extensible Context Oriented Reasoning Architecture) framework [198], which uses the ELVIN4 framework instead of JADE. This work claims to resolve issues resulting from the use of logic and sensor-based formalisms in context representation by providing the so called Context Spaces model, based on state-space models. Addition-

ally, this approach includes *ad hoc* mechanisms to facilitate reasoning with uncertain contexts.

The rise of both SW and MAS technologies has favored the appearance on the horizon of recent infrastructures built from a combination of them, which is very close to our view of KMob. For instance, the proposal by Lassila and Khushraj, which will be implemented in the SwapMe (Semantic Web Application Platform for the Mobile Ecosystem) project at Nokia Research and MIT, also aims to combine the best elements from a wide range of different fields (MAS, SW, mobile computing, etc.) to implement effective smart mobile systems [9, 158]. Also interesting is that fact that CoBrA, though more oriented to Ambient Intelligence, heavily relies on MAS and SW as well as underlying data communication technologies [67]. Finally, we should like to mention the PLIMM (Product Line enabled Intelligent Mobile Middleware) middleware [263], which uses ontologies to represent knowledge in the system and, more specifically, context. This middleware deploys BDI (belief, desire, intention) agents able to reason with OWL on a supporting platform, which can either be a SOA (compliant to OSGi standard [5]) or Jadex [205].

2.6 Towards Knowledge Mobilization

The numerous approaches to the development of intelligent mobile systems seem to indicate that the implementation of a KMob system is not an easy task, and can be carried out from many different perspectives. Various new and often immature technologies need to be integrated in order to achieve communication between mobile entities. Nevertheless, this is not the greatest problem that developers have to face. As explained in Chapter 1, successful KMob systems require new methodologies, architectures and knowledge models.

The research work reviewed in the previous section is largely aimed at satisfying these requirements, either partially or completely. Although these contributions are valuable, most of them are excessively focused on

the integration of technologies. Instead of coming to grips with the most crucial problem of a knowledge-intensive system, which is the representation of knowledge, this work limits itself to establishing *communication*, between mobile peers, and has nothing to say about achieving understanding. Such work targets less specialized applications that do not require elaborated knowledge to be managed.

For this reason, our research makes a detailed study of the problems that arise when refined knowledge, retrieved from large and heterogeneous information sources, has to be provided to decision-makers. Obviously, we take into account related research, but at the same time explore new ways to offer support for Kmob systems that must deal with complex problems. Thus, we propose a software architecture and a context-dependent knowledge representation model that solve information distribution and overload problems in KMob systems. The representation formalism is explained in Chapter 4, whereas the software architecture is described in the next section.

CHAPTER 3

An Architecture for Knowledge Mobilization

This chapter presents a proposal for a new abstract architecture for KMob systems. The chapter begins by exploring the rationale behind this proposal and the features that a KMob architecture should have. It then goes on to provide an introduction to software agent-based architectures. The main content of the chapter is the description of our architecture, which uses the semi-formal language AML. Since the architecture can be implemented by using different technologies, we also offer a description of alternative development frameworks.

3.1 Rationale

In the previous chapter, we presented various frameworks, standards, specifications, etc. for building KMob systems. These technologies were classified in terms of three abstraction levels, namely, data, information, and knowledge, according to the degree of refinement of the managed values (Figure 2.1).

The implementation of a KMob system might be started from scratch, and rely on (a selection of) these tools. Although such a straightforward

approach to the problem could be time-saving in the short term, it would probably turn out to be expensive in the long run. The reason for this is that software maintenance, support, and extension costs often soar if the system is very complex, and if incompatible, immature, or undocumented technologies are used.

Since this is a likely scenario in KMob, it is crucial to rigorously follow the recommendations of Software Engineering. The software life cycle[1] selected should pay special attention to the design stage, when the developers identify the global structure of the future system. The schema describing the system organization is called its architecture, and its design is one important factor that affects the success of the system.

In this chapter, we propose a general architecture for KMob systems that can be adapted to different application domains. This reference architecture semi-formally defines a set of abstractions that are the building blocks of a KMob system. These abstractions are not very specific, and must be adapted to particular applications. Therefore, this proposal may be regarded as more of a meta-architecture, i.e., a description of the possible architectures, rather than an architecture, as the term is commonly used. Lower-level solutions regarding system implementation (programming languages and APIs, communication protocols, etc.) are not dealt with in the architecture, and the specific details must be decided by the designer.

The utility of such an architecture is inversely proportional to the experience of the system designer. Since KMob is a new perspective on corporate systems, this issue is especially relevant. Actually, mobile systems development as a whole poses several challenges to IT professionals, mainly due to the fact that long-established software engineering methodologies for desktop applications are no longer valid [224]. Therefore, a design artifact for mobile systems like this one is a very valuable contribution to the field, since

[1] The software development cycle consists of a sequence of stages that go from the informal specification of requirements to the deployment and maintenance of the final product. Each stage has associated techniques, procedures, and tools. Software life cycle processes in Software Engineering (e.g. iterative, waterfall, agile, or extreme programming) define alternative methodologies for each of these stages.

it saves time and money during the development process.

The following objectives must be achieved by the architecture. Most of them correspond to non-functional requirements of KMob systems, which are requirements that are not related to *what* the system has to, but rather to *how* it is done. Unfortunately, these features are not orthogonal and are often in conflict with each other.

First, the architecture must be adaptable to a variety of situations, namely, different domains, interaction patterns between elements, and technologies. The communication schema may range from untethered asynchronous message queues to highly-demanding real-time synchronous processing, and the architecture must be able to reflect these situations. Managing different configurations can be achieved by decoupling actors, actions, resources, and communication channels as much as possible.

Two other features that a KMob architecture must provide are extensibility and robustness. Extensibility may be required in diverse dimensions, for instance, in the number and type of system users or in the amount of resources available. A robust system guarantees that it will be operative most of the time. The more critical the task to be resolved, the greater the effort that must be made to build a reliable system. Given that system failure may be triggered by many uncontrollable facts, obtaining a robust system is usually expensive.

Security is a very important property that deserves a special comment. Security entails user authentication, integrity of the data, and encryption of the communications among other tasks. KMob systems must be secure, but our architecture does not tackle this matter in depth because a complete study of it is outside the scope of this work. As widely pointed out in the literature [123, 166], security in mobile systems and applications is an extremely complex issue that transversely affects the layers of a mobile infrastructure. Nevertheless, security mechanisms can be easily incorporated in our architecture.

Given these requirements, we propose an architecture based on the multi-

agent paradigm. The reason is that the multi-agent paradigm has proved to be capable of supporting the development of distributed systems in several domains with different communication schemas [244]. Moreover, multi-agent systems are scalable, adaptable to different requirements, and robust.

The multi-agent paradigm provides the theoretical foundations for both describing and implementing distributed systems. We will use the agent paradigm for describing the architecture of KMob systems. The components of the architecture will be modelled as agents. However, the developers will be free to choose any implementation framework for the architecture, not only multi-agent platforms. For instance, Web Services can be used to implement the architecture when the communication schema of the application is of the client-server type.

In the next section, we study multi-agents architectures as a metaphor for designing KMob systems (Section 3.2). Before presenting our KMob architecture in greater detail (which is done in Section 3.3), we provide a brief introduction to agent-based architectures as well as the modeling language AML (Section 3.2). Finally, we describe three combinations of technologies that can be applied to implement the architecture in three applications domains (Section 3.4).

3.2 Agent-based architectures

Architectural patterns

Software architectures have been studied in Software Engineering for some time now, but only recently they have been acknowledged as important artifacts in the development process. This is mainly due to the evolution of monolithic software systems as distributed networks of computational resources. The recently adopted ISO standard IEC/DIS 25961 (formerly IEEE Std 1471-2000) states than an “architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its com-

ponents, their relationships to each other and the environment, and the principles governing its design and evolution" [124].

Basically, a software architecture abstractly specifies (i.e. unnecessary details are not presented) the structure of a system (i.e. the distribution and the responsibilities of the elements), the communication between the components (i.e. the flow of information), and other additional requirements (i.e. technical, quality, or business oriented) [104]. The architecture of a software product can be described from various perspectives, which produce different views of the system. Commonly used views are the functional/logical view (focus on structure), the concurrency/process view (focus on communication), the development view (focus on software modules), and the physical/deployment view (focus on implementation).

Documenting a software architecture has always been something of a headache for IT managers. Revising or sharing a specification is often rather difficult. For this reason, semi-formal methods for modeling software systems have been proposed, such as UML (Unified Modeling Language) [69, 86].

An architecture is said to *comply* with an architectural pattern when it has the typical features of the pattern. There are several predefined architectural patterns that have obtained considerable success in specific application domains. These best patterns have been verified and studied in great detail. Thus, if a problem fits a certain pattern, design and development can be guided by the pattern specification. Three widely used patterns for distributed systems are client-server, service-oriented, and agent-based architectures.

The client-server pattern is one the most frequently used architectures in distributed computing. In its basic form, it suggests a simple intercommunication schema between two clearly-distinguished entities: clients and servers. Clients are programs that perform minimal processing and require few computational resources. Clients delegate most of the tasks to servers, which run on more powerful platforms, and are responsible for satisfying client requests. Workload can be balanced between clients and servers in

such a way that clients may need servers only to carry out certain complex tasks such as database management.

The evolution from simple (*thin*) to complex (*fat*) clients has given raise to Service Oriented architectures (SOAs), which, to some extent, may be considered an enhancement of the client-server paradigm. Functionalities in a SOA are provided by services, which are stateless facilities that can be accessed remotely. Services are usually implemented with Web Services, a standard from the World Wide Web Consortium (see Section B.5).

Web Services also separate clients and servers, which have very different roles in the architecture. Nevertheless, in some situations a client may become a server (and vice versa), in such a way that each component of the architecture can alternatively request and provide information. This pattern is known as Peer-to-peer (P2P), and depicts a scenario where groups of loosely-coupled equally responsible entities communicate directly to accomplish an objective.

Peers showing a certain degree of autonomy and awareness can be regarded as agents, in the classical sense of the term [203]. The multi-agent paradigm depicts a scenario where the agents, which are autonomous, proactive, and context-aware computational entities, communicate to achieve a goal [259]. A multi-agent architecture is a description of a system in terms of agents' capabilities, organization, and interactions [91]. Multi-agent theory proposes different kind of agents, system structures and collaboration policies, which should be adapted to the problem to be faced [172]. Multi-agent software engineering techniques [28, 29] can be used to develop a multi-agent system.

Very few general architectures tackling the specific problems of mobile applications have been created, and most are adaptations of the client-server pattern. Not surprisingly, this is a natural way of structuring mobile systems, given the computational limitations of mobile devices. Thus, several research studies focusing on the adaptation of the client-server architecture to mobile environments have been published [131, 159].

Though these approaches consider participation of both thin and fat clients, intelligent applications require more diverse interaction patterns. As a result, other architectures have been proposed in the context of Intelligent Systems. Most of them have an associated implementation framework, such as those mentioned in Section 2.5. These *ad hoc* architectures are usually inspired on existing paradigms – from pure client-server to multi-agent –, and are tailored to meet specific needs.

Our contribution in [77] provides an abstract architecture for intelligent mobile systems that overcomes some of the issues of related approaches. This architecture relies on the service-oriented and the multi-agent paradigms. Services are modelled as agents, and a convenient multi-agent architecture is proposed. With this architecture, the choice of implementation technologies is left to the developer, who can use agent and non-agent platforms. As a matter of fact, the paper showcases the PDA^2 (Psychological Disorders Assistant for PDA), a mobile knowledge-based system to retrieve information about psychological disorders. PDA^2 is designed as a multi-agent system, but is implemented as a Web application. The client, equipped with a portable device, consults the server by using a Web browser.

In Section 3.3, an improvement on the architecture in [77] is presented in detail. The new architecture, also based on the multi-agent paradigm, is described using AML, a software specification language for agent-based systems. An overview of AML is provided in the next section.

The Agent Modeling Language

The Agent Modeling Language (AML) is a semi-formal visual language for specifying, modeling, and documenting systems in terms of concepts from MAS theory [256]. AML defines a set of elements and restrictions that the software designer can combine to graphically depict the structure, relations, and procedures in an agent-based system.

Similar approaches to AML have been proposed. The survey in [63] compares AML with other agent-modeling languages, such as Gaia, AUML, Mes-

sage, Tropos, MAS-ML, and AOR. AML is less academical and more practical than the other languages. As a result, it is more applicable to real problems. We have decided to use it because it is more flexible, and is better supported by existing visual design tools. It also has the advantage of having more available documentation than other proposals.

Furthermore, AML has solid underpinnings, since it is based on the Unified Modeling Language (UML) [69, 86]. UML defines entities (with an associated graphical notation) that represent elements of the application domain, such as actors, resources, procedures, etc. These entities are included in UML diagrams corresponding to different views of the system architecture. UML representations are simple enough to be understood by both developers and clients so that feedback can be directly provided. Nevertheless, in order to interpret all the details of a software design, more knowledge regarding language particularities is required.

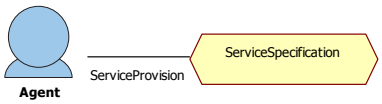
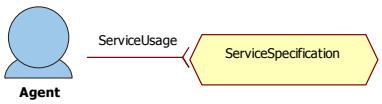
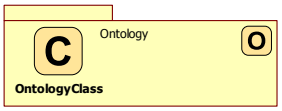
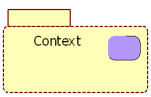
The formal semantics of UML entities and diagrams are stated in the UML metamodel. AML is defined as a layer on top of the UML metamodel, extending it with new primitives consistent with UML entities. The most important AML primitives are the following: (i) *entities*, which are social and autonomous elements with a behavior and a structure; (ii) *relationships* or associations between entities; (iii) *behaviors*, defined as procedures performed by entities; (iv) *execution environments* or platforms where entity implementations are deployed; (v) *ontological* elements or components of a knowledge model.

Table 3.1 summarizes the AML constructors employed to specify our architecture, and shows their graphical representation. Next, a brief description of each is provided[2].

AgentType. This is the type assigned to AML agents. In AML, an agent is any autonomous entity that passively or proactively interacts with the environment, independently of its implementation.

[2]Since it is out of the scope of this dissertation to provide a complete overview of AML, we refer the reader to the AML specification for further details on the language [63].

Table 3.1: AML primitives

Primitive	Iconic display
AgentType	
ResourceType	
EnvironmentType	
EntityRoleType	
ServiceSpecification	
BehaviorFragment	
SocialAssociation	
PlayAssociation	
ServiceProvision	
ServiceUsage	
ExecutionEnvironment	
Ontology	
Context	

ResourceType. Resources are passive entities with associated properties that are accessible inside the system. External resources in AML are defined as UML Actors.

EnvironmentType. Environments are the logical or physical surroundings where entities exist. The environment type defines a set of properties for defining which entities *live* in the environment (agents, resources, etc.) and some of their features (rules, laws, policies, etc.). In the same way as resources, external environments in AML are defined as UML Actors.

EntityRoleType. Generally speaking, a role denotes a set of features and capabilities that an entity may decide to acquire.

ServiceSpecification. This class is used to describe accessible services in the system. Such services are defined by their protocols (entry points that a service offers) and behaviors (activities that the service performs).

BehaviorFragment. Behavior fragments (partially) describe the dynamics of an activity performed by an entity to achieve a goal.

SocialAssociation. A social association is a bidirectional property denoting a relationship between two socialized entities.

PlayAssociation. A play association is a bidirectional property denoting that a behaviored entity can acquire a role.

ServiceProvision. Service provision stands for a dependence relation between the provider entity and the provided service.

ServiceUsage. Service usage stands for a dependence relation between the user entity and the service used.

ExecutionEnvironment. An execution environment models the physical platform where entity implementations run. It is interesting to note that new execution environments can be defined by specialization of the basic

ExecutionEnvironment. Thus, implementations with different frameworks (including non-agent technologies) can be described with AML.

Ontology. Ontology is the basic primitive to specify a knowledge base in AML. Ontologies are used in agent systems in several scenarios: to define a common language for agent understanding, to describe the features of a service, to build the knowledge models supporting a knowledge-based system, etc. Other ontological constructors are OntologyClass and OntologyUtility.

Context. Context is a primitive to describe the properties of the section of an AML model that is to be considered in a particular situation. These situations can be specified with UML constraints or states.

AML specifications are organized in diagrams, which are extensions of the standard UML diagrams. In our specification, the following diagrams are used:

Society Diagram. A society diagram presents a general view of the architecture of the multi-agent system. Entities (agents, resources, environments, and organization units) and relationships (social associations, play associations) are depicted in these diagrams. This diagram is a specialization of UML Class Diagram.

Entity Diagram. Entity diagrams are used to present in detail the structure of an entity: features, behaviors, ports, roles played, services provided and requested, etc. This diagram is a specialization of the UML Composite Structure Diagram.

Protocol Sequence and Protocol Communication Diagrams. These are two different kinds of diagram that are used to specify communication acts between entities. They are specializations of the UML Sequence and Communication Diagrams, respectively.

MAS Deployment Diagram. These diagrams specify how the multi-agent system is deployed on the execution platform. This diagram is a specialization of the UML Deployment Diagram.

3.3 Architecture description

In this section, our abstract architecture for KMob systems is presented. The full specification can be downloaded from http://decsai.ugr.es/~jgomez/thesis/. This architecture is based on the proposal in [77], which is extended to be adaptable to different KMob application domains. The new architecture is described using the AML language.

Foundations

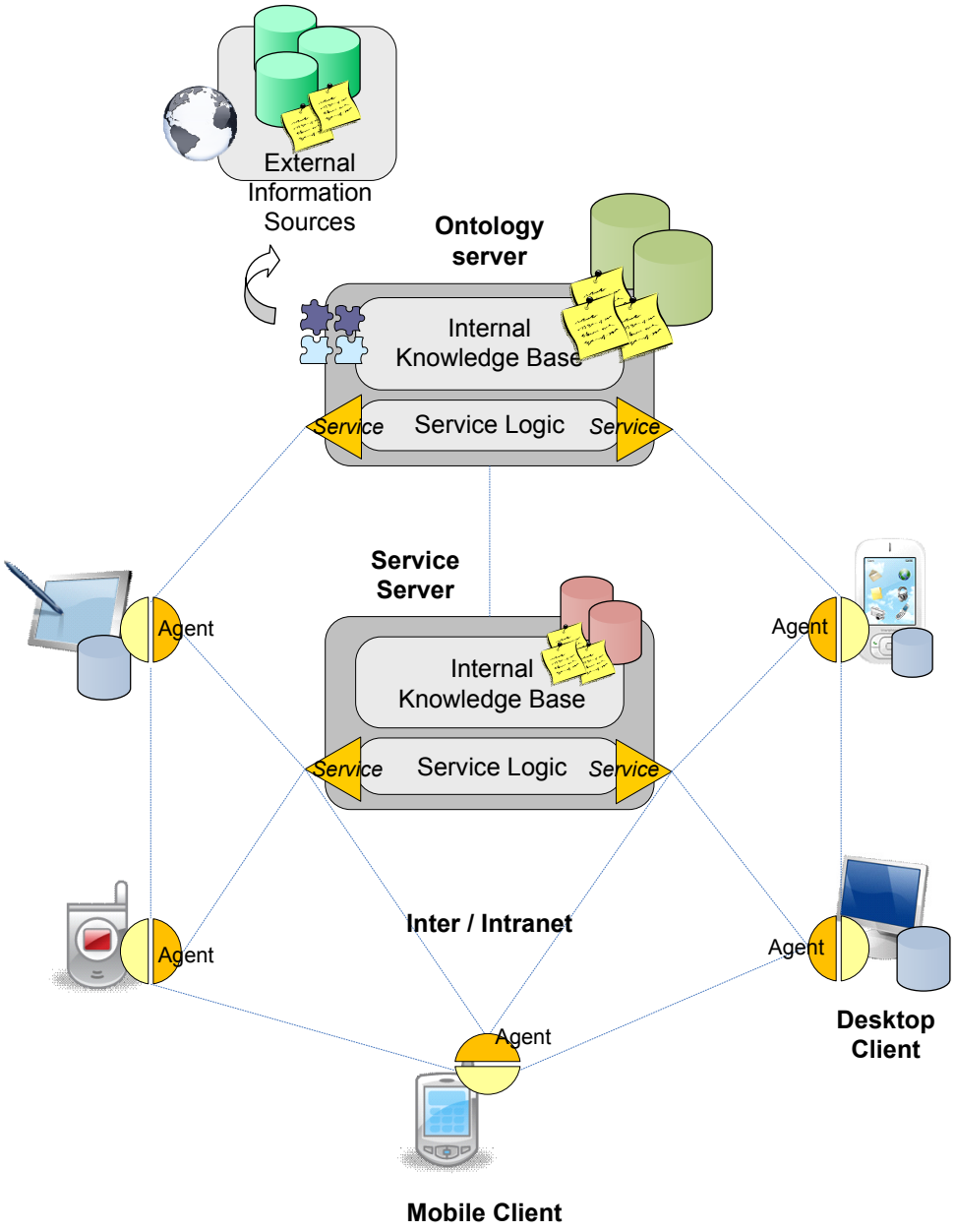
The architecture for mobile Knowledge-Based Systems in [77] has three main components, namely, clients, service servers, and ontology servers (Figure 3.1). This naive architecture reflects the following idea. Clients request services, which are provided by servers. Servers solve client requests, sometimes by consulting other services, and returns an answer to the clients. Communication between clients and servers is performed throughout the network infrastructure by using a suitable communication protocol.

The servers of the architecture implement the services provided by the system, i.e. they offer an access point to the functionalities of the application. These functionalities can be large database querying, real-time data supply, reasoning with a knowledge base, or any kind of expert decision support.

Given that knowledge management is crucial in Knowledge-Based Systems, we distinguish a special service provider, namely the ontology server. This ontology server is responsible for the management of the global knowledge base of the system, as well as the incorporation of other information sources, which may be external data repositories.

The clients are expected to run on mobile devices, such as cell phones and PDAs. If the mobile device has very limited computational capabilities, it will be forced to delegate most of the processing to the servers. Alternatively, it may happen that a client runs on a more powerful device (which, in the best case, may be a desktop device), and is able to perform heavier processing.

Figure 3.1: Components of a mobile Knowledge-Based Systems



In this case, the client can carry out more tasks, and even handle a local knowledge base. Therefore, depending on the features of the device, there will be clients with different degrees of intelligence: intelligence alludes to the ability of the client to solve queries with its own knowledge.

We have extended this architecture to better represent this variety of client abilities, which occurs in KMob systems. The extended architecture for KMob is described in the remainder of this section.

As explained below, we have completely decoupled agents and roles. Entities of the previous architecture are modelled as agents in the new architecture description. Thus, clients and servers are now agents (KMob Agents). The actions that an agent can perform have been separated from the agent description, and are represented now as roles. Thus, a role is a set of methods that an agent can execute (KMob Roles).

The difference between agents running on mobile and desktop devices is represented by defining two subtypes of agentsm namely, Desktop Agents and Nomadic Agents. The execution environment determines the features and requirements of an agent because data processing and transmission are limited in mobile devices, and they are used in dynamic situations.

Nomadic and Desktop Agents can acquire different roles. We consider that both agents running on mobile and desktop devices can eventually perform as knowledge requesters and providers. Obviously, in a P2P application, every agent will be both requester and provider, whereas in a pure client-server system, mobile clients will only be requesters and desktop servers will be mainly providers. The advantage of the architecture is that it can be adapted to these different scenarios.

Starting from these primitive elements, other entities and roles have been defined. For instance, the former ontology server would be a Desktop Agent with connections to Local and External Knowledge Models that acquires the Provider role. Next, the new agents and roles are described.

General structure

An overview of the architecture for KMob systems, in AML notation, is shown in Figure 3.2.

KMob Agents are the basic elements of the architecture. They encapsulate all the processing associated with nomadic and static entities. Two kinds of agents are distinguished: (i) Nomadic Agents, which run on mobile devices; (ii) Desktop Agents, which run on application servers. Agents may be deployed in a public, private, or mixed network, which should support communication between them. Basic KMob Agents can be specialized in concrete applications by organizing them into subclasses with additional properties.

Each agent, nomadic or desktop, manages a Local Knowledge Model. The richer the local model, the more intelligent the agent, since it will be capable of better resolving a greater number of problems. When an agent is not able to answer a question with its own knowledge, it requires the services of other agents or the contents of External Knowledge Sources. Services are offered by both Nomadic and Desktop Agents, although Desktop Agents usually provide more complex services. Integration of internal and external knowledge can either be performed on the fly, i.e. as necessary for satisfying a service request, or off-line, i.e. by creating a warehouse with a permanent mapping between internal and external knowledge.

When agents interact directly as a result of a service request, or indirectly, because they share their resources or their objectives, Social Relations are established among them. Social Relations may range from implicit associations resulting from the functioning of the system to explicit connections with a well-defined structure.

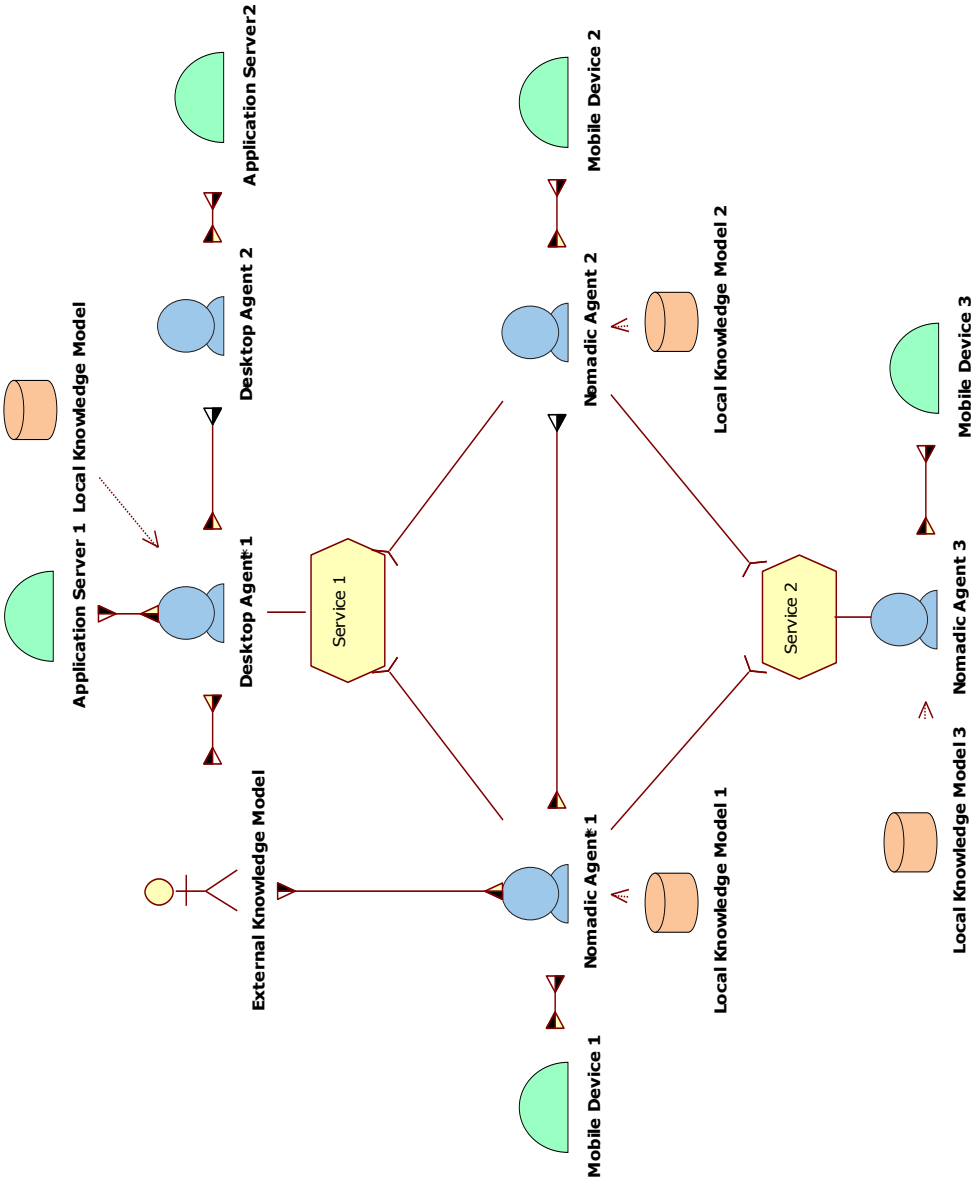


Figure 3.2: Society Diagram of the KMob architecture

Architecture components

Agent roles

The basic roles that can be adopted in the KMob architecture are Consumer, Provider, Directory, Facilitator, Integrator, and Broker (Figure 3.3). Some or all of these roles can participate in a KMob application. More interaction patterns that might be considered are those specific of the application to be implemented.

Consumer and Provider roles are self-explanatory. An agent becomes a consumer when it requests a Service. The petition is processed and eventually resolved by the Provider. In order to supply a suitable answer to the Consumer, the Provider may turn to a second Provider (for instance, an External Knowledge Model) and play the role of a Consumer (Figure 3.4). In the basic situation, a Consumer's requests are processed on-demand synchronously, although in some cases asynchronous communication may be preferred. A different situation occurs when a Provider decides to proactively supply information to a Consumer. In other words, the communication act is not started by the Consumer entity. It is the Provider who sends an information package to a Consumer, who can accept or reject it.

A Directory is a special kind of Provider that resolves queries asking about the features of the services in the system. An agent playing the Directory role can be consulted, for instance, if there is a service with a given name available (in the simplest case), or if there is a service that can fulfill a specific task. To have an effective directory, services must be registered before being offered to consumers. Service-oriented and multi-agent platforms usually provide mechanisms for elemental directories; e.g., the UDDI Web Services protocol is intended to create and consult a service directory. More recently, formal languages such as OWL-S have been proposed to attach semantics to service descriptions in order to automate location, integration, and invocation. These approaches, which are commented in Section B, can be used in the directories of the architecture. In simple applications, agents offering di-

Figure 3.3: Society Diagram of the agent roles

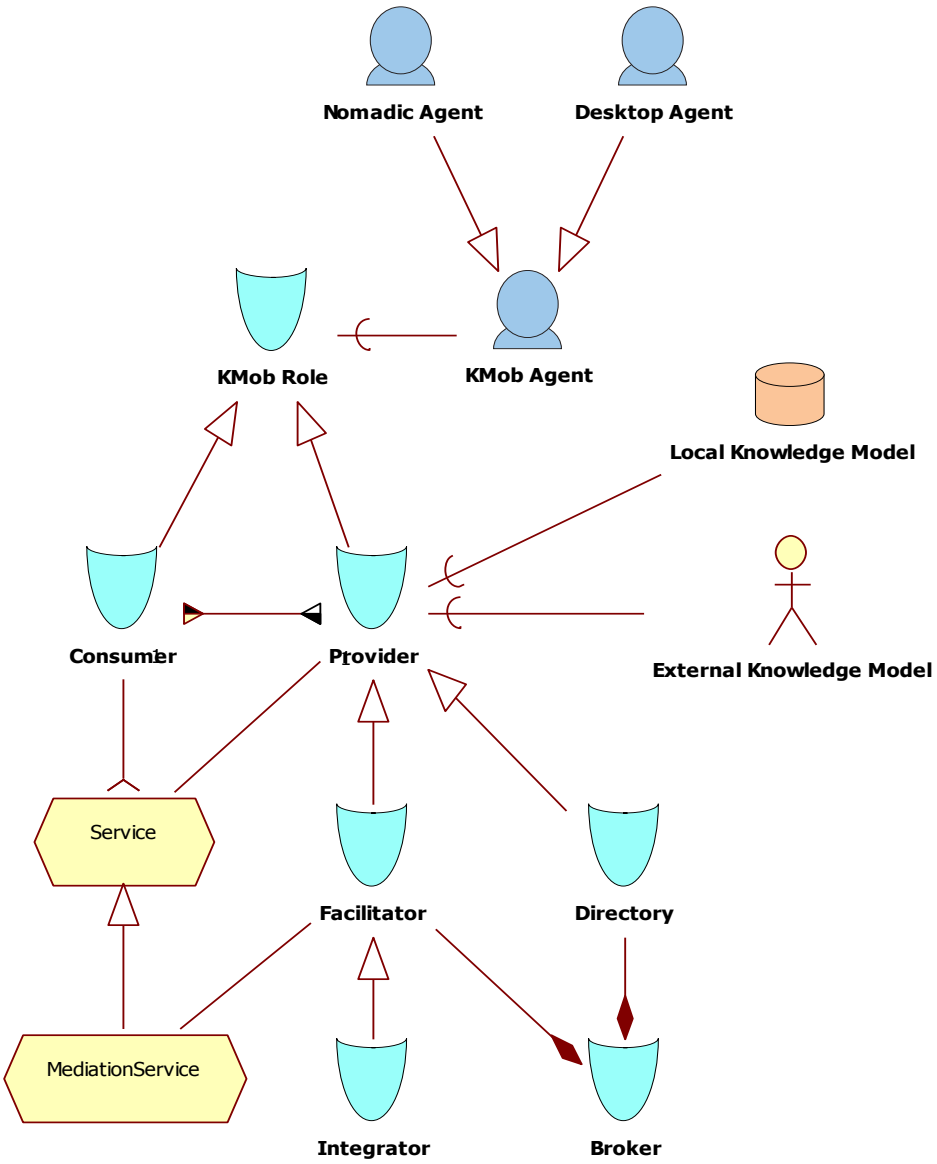
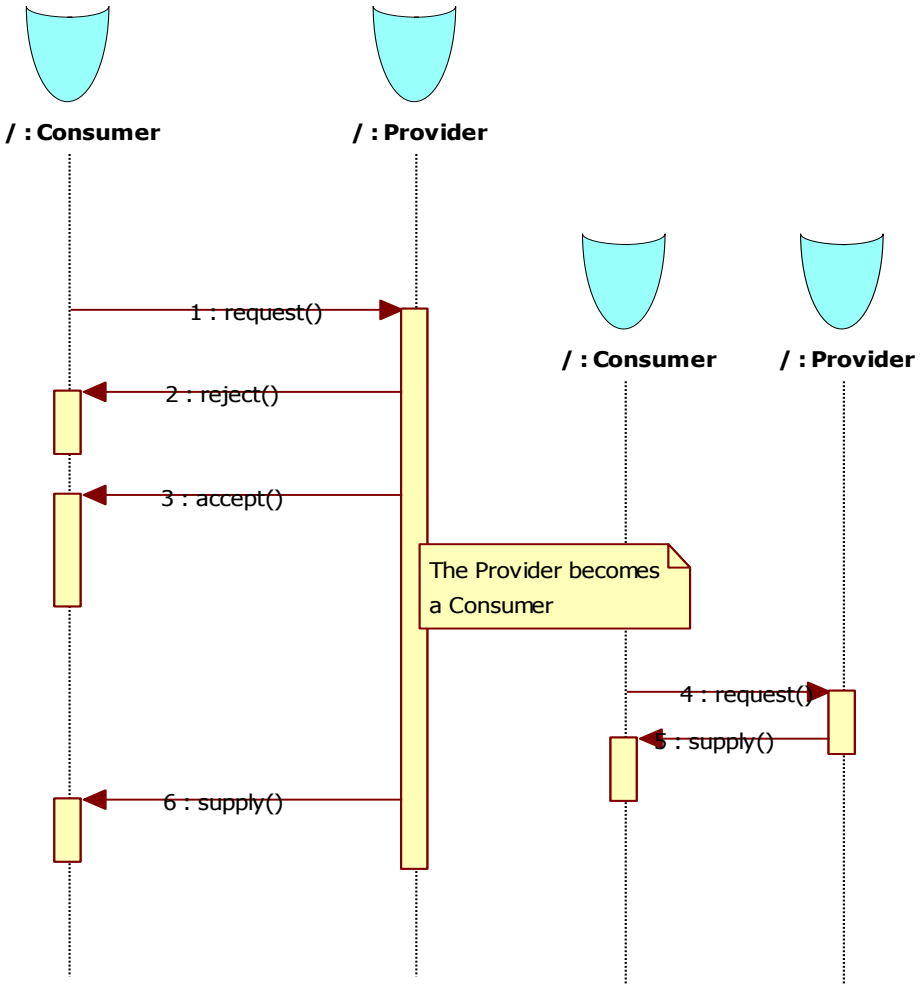


Figure 3.4: Protocol Sequence Diagram of the roles Consumer and Provider



rectory services will not be necessary since every agent can have a predefined list of services.

Facilitator is the role acquired by the agents that provide Mediation Services between entities that cannot communicate directly. This issue may arise in several circumstances; e.g., incompatibility between agents (different communication protocols or languages/ontologies are used), or security (the Facilitator is a gateway between the secured and the private network). Implementing a Facilitator role may be a difficult task, since mediation can be extraordinarily complex in some applications.

Integrators are the specialized Facilitators responsible for combining local and external knowledge sources, i.e. mediating between local and external knowledge providers. Ideally, integrators should allow owned and external knowledge to be accessed together transparently in such a way that consumers are unaware of the origin of the information.

A Broker is a role that offers both facilitation and directory services. Brokers are expected to perform several complex tasks to satisfy a request: location of services, matching of consumer preferences and provider capabilities, mediation, etc. This kind of role is frequent in applications with several providers and thin clients, who delegate most of the processing to the broker entity. Brokers are also responsible for guaranteeing quality of service in highly-demanding systems.

3.4 Frameworks and technologies

The architecture in Figure 3.2 depicts the basic components in a KMob system, but it does not state how they should be implemented. In this section, we describe three combinations of technologies (i.e., frameworks) that can be used to implement KMob systems[3]. These technologies are multi-agent

[3]Since processing in the nodes may be disparate, and will probably include knowledge management tasks, ontology tools, such as management APIs or inference engines, are generally used. These technologies are described in Appendix B.

platforms, tuplespaces, and Web-based protocols. Each framework is suitable for a particular instantiation of the architecture.

The architecture can be directly implemented using multi-agent platform, such as JADE. The multi-agent technology is especially recommended in applications with independent components that require complex coordination policies. Nevertheless, it is known that multi-agent platforms require a considerable amount of computational resources to run properly. Therefore, the multi-agent framework should be used only when the application requires all the features of a multi-agent platform.

In other cases, communication is the main issue of the KMob system. Accordingly, developers should use technologies that promote effective communication between distributed components. This is a feature provided by tuplespaces, which are the second framework suggested in this section.

On the other hand, in some applications communication is not the main problem, but rather the processing that must be carried out in certain components of the system. In these situations, a good choice is to implement a client-server application using Web technologies, such as HTTP-based interaction or Web Services.

Multi-agent technologies

The client-server and the tuplespace frameworks, described below, are simplifications of the more general communication schema supported by the architecture. Nevertheless, in some scenarios it is not possible to be absolutely certain of the structure of the conversations between consumers and providers. Moreover, entities act as a consumer or a provider depending on their needs and capabilities. Ultimately, in these situations communication may take place among equally gifted entities, corresponding to a peer-to-peer schema.

This is the case of the Ubiquitous Security Guarding application (Section 2.3). The guards are provided with mobile devices which continuously

send and receive information about the state of the areas under surveillance, both on demand and proactively. For instance, a guard can request the video signal currently captured by a camera. He can also receive a notification stating that an alert has been triggered in a sector, and automatically, the proper signal is displayed automatically on his device screen. Other links could be established between the guards in such a way that they are able to transparently communicate among themselves as well as with other nodes of the system.

An agent platform is a natural approximation to implement such a system. Entities exhibit a high degree of autonomy and, at the same time, there are social dependencies between them, which are one of the most notable features of multi-agent systems. Our architecture can be almost directly implemented using one of the MAS platforms presented in Section 2.4. In fact, certain contributions aim to automatically generate a programming code from AML specifications automatically [150]. Among these platforms, we propose the JADE/LEAP middleware as a suitable framework to implement the KMob architecture in that kind of application.

JADE (Java Agent DEvelopment Framework) is an open-source middleware oriented to the development of distributed multi-agent applications based on peer-to-peer communication [23, 22]. JADE offers three utilities to developers: (i) an API for agent programming; (ii) a runtime where agents are executed; (iii) a graphical tool to administer running agents.

JADE agents are implemented with the JADE API, which includes Java classes to manage agents, behaviors and messages, and Jave methods to send and receive messages. JADE agents are loaded into the runtime, which manages agent lifecycle and message passing. Agent state can be modified with the graphical tool, which provides an interface to check the runtime status. JADE avoids dealing with low-level localization and communication mechanisms, and different transport protocol may be used (e.g. SOAP).

JADE is compliant with the specification of the IEEE standards committee

FIPA[4] (Foundation for Intelligent Physical Agents) [94]. The FIPA standard, intended to "promote the interoperation of heterogeneous agents and the services that they can represent", is divided into five categories: agent communication, agent transport, agent management, abstract architecture, and applications. The agent communication category is the most important of the five since it defines the Agent Communication Language (ACL), a language designed to achieve understanding between diverse agents.

JADE is extended with the LEAP add-on [25], a set of additional libraries that allow JADE agent containers to run on mobile devices, such as PDAs (implementing the CDC profile of J2ME) and cell phones (CLDC profile of J2ME). A LEAP container can be executed in a stand-alone or split mode. This split mode is targeted to devices with limited computational capabilities, and thus it is the most suitable to be used in the KMob Multi-agent framework. The main drawback of LEAP, and consequently, of JADE, is that their formality makes them rather inefficient. Therefore, they are used only when all their features are required.

Tuplespace technologies

Another scenario is one with several clients, also with computational limitations, but who need to share part of their knowledge. This is the case of the KMob scenario Portable Tourism Guide (Section 2.3): information about sightseeing spots is still accessible to all of the numerous tourist community as a whole, but each tourist is able to introduce new data (tags, reviews, plans, etc.) in the system. Since little processing can be carried out by the clients, they must delegate heavy knowledge management tasks to a server. However, the knowledge base should be, at least to a certain extent, shared among the clients.

A suitable solution to these issues is that of *tuplespaces*. A tuplespace is a knowledge repository composed of n-tuples that can be accessed remotely and concurrently (a n-tuple is an ordered sequence of n items). Tuplespaces

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

were firstly proposed by Gelernter in the context of generative communication, a paradigm for asynchronous exchange of information in distributed systems [99]. Tuplespaces act as mediators in generative communication systems. In order to communicate an entity *A* and an entity *B*, *A* stores a tuple in the tuplespace and *B* reads it. Generative communication was supported by the programming language Linda, which defines three basic operations to access a tuplespace: `out(`*tuple pattern*`)`, to write in the tuplespace; `in(`*tuple pattern*`)`, to read and erase a tuple from the tuplespace; and `read(`*tuple pattern*`)`, analogous to `in` but without deleting the tuple. When a query operation fails, the calling procedure is blocked until a suitable tuple is available. In this manner, asynchronous communication is implemented from synchronous read-and-write operations.

Tuplespaces and Linda are similar to blackboard architectures defined by the multi-agent paradigm. The objective of both is to provide a central knowledge repository that can be accessed by thin clients using simple operations. More recently, other initiatives have taken a similar approach to Linda. For example, JavaSpaces are a Java technology included in Jini for implementing tuplespaces that can be distributed over the nodes of a network [31]. Tuplespaces also have received attention from researchers seeking a simple mechanism to implement interoperable mobile systems [184].

On the other hand, tuplespaces can be easily adapted to support Semantic Web applications, given the fact that the basic primitive of RDF language is the triple, i.e. the 3-tuples. In that regard, RDF-based Semantic Web Spaces [189, 255], OWL-based sTuples [140], and WSMF-based Triple Space Computing (TSC) [152], have been proposed to accomplish Semantic Web coordination by using tuplespaces.

Incorporation of a tuplespace in our KMob architecture is quite straightforward. A Tuplespace role can be defined by the specialization of Facilitator. An entity acquiring the Tuplespace role provides a Tuplespace Service, which is a specific Mediation Service offering Linda-like operations. Read and write messages will be sent to the tuplespace entity using, for example, SOAP or *RESTful* messages. The tuplespace can be implemented by adapting one of

the previously mentioned contributions, or by creating a specific solution – e.g., a Linda API wrapping a Sesame RDF repository [54].

Client-server technologies

The simplest scenario is that involving a pure client-server interaction pattern. Clients are not able to execute complicated procedures. As a result, queries are totally pre-programmed and processed almost completely in the server. Adding a new functionality implies implementing a new service. Communications are one-to-one, with a client asking for a service, and a service replying to a client. In this case, the Consumer and Provider agents are clearly identifiable, and the communication schema is totally predefined.

The Nomadic Healthcare use case (Section 2.3) is a good match for the client-server schema. Doctor Greg, equipped with a mobile device requires information from a centralized server where the bulk data is stored. The client application, running on the doctor's mobile device, carries out very little processing. It is only used to acquire the clinical description of the patient, and to display the results retrieved from the HIS. The server application accepts client requests and translates them to the HIS, acting as a gateway between them. The client may additionally include some knowledge in his local model to work out simple or previously solved queries. The greater or lesser complexity of the reasoning will naturally depend on the mobile device capabilities.

Client-server communication between a mobile device and an application server can be implemented by using various technologies. The lowest-level communication issues are handled by means of a wireless or cellular communication technology, e.g. Wi-Fi or UMTS (Section 2.4). Nevertheless, the developer can remain relatively unconcerned about the data communication layer, since middleware systems or, alternatively, Web technologies are available. We consider that a Web-based implementation is appropriate for the Nomadic Healthcare system, and in general, for client-server KMob systems.

Web applications are systems accessible throughout the Internet using the HTTP protocol. Server logic can be implemented with J2EE [133] or .NET [65], two current development platforms for corporative systems. Web applications are usually structured in two or three tiers, decoupling the back-end storage (e.g. databases) and the presentation layer. The interface of a web application can be oriented to human users or automatic procedures. For human users, HTML pages with a suitable format are created in the server and sent to the client; for automatic procedures, there is an entry point managed by a callback procedure, i.e. a Web Service, enabled to accept XML requests. Since both J2EE and .NET platforms supports HTML generation and Web Services, either can be selected. We focus on J2EE because it is multi-platform and its specifications are open. However, all considerations can be easily extended to .NET.

Human-readable web interfaces with J2EE can be created by using different technologies. For instance, *servlets* and JSPs (Java Server Pages) are server components that dynamically generate HTML content [183, 262]. These components are deployed in a special web server, such as Tomcat[5]. Servlets and JSPs have been recently complemented with JSF (Java Server Faces), which simplifies the creation of HTML interfaces in Java Server applications. Interaction between clients and servers with these technologies is described as "click and wait". In other words, the contents are downloaded from the server and presented to the client after a link in the document is clicked. This behavior is problematic in certain scenarios since user experience is downgraded when communication latency is high. The user is idle while he is waiting for the document to be transmitted, whereas the network is idle while the user is reading the document.

Ajax is a recent technology aimed at overcoming these problems [72]. Ajax defines a set of recommendations for the implementation of web applications in order to accomplish: (i) partial execution on the web browser (with Javascript); (ii) asynchronous message sending (with the `XMLHttpRequest` object); (iii) presentation of new contents by modifying only the

[5] http://tomcat.apache.org/

sections of the document that have changed (with the Document Object Model). These properties are very interesting in KMob applications because richer clients could be implemented in Ajax-enabled mobile web browsers (e.g. Opera Mobile[6]). New steps towards a better balance of the processing load are RIAs (Rich Internet Applications). RIAs are a set of competing platforms that split execution between web servers and clients. To date, these platforms are in an embryonic state though they will be incorporated in commercial mobile phones in the near future, e.g. Silverlight in Windows Mobile 6 and Nokia S60 Series.

Alternatively, Web Services can be used. Web Services are used in machine-to-machine communications, i.e. communication acts where human participation is reduced. Web Services have succeeded as means to implement corporative distributed systems. J2EE and .NET platforms offers extensive support for them. An alternative approach is the REST model (REpresentational State Transfer), which proposes a simpler architecture based on *ad hoc* XML request-response messages and APIs. This model is preferred when the power and the formality of the Web Services protocol stack are not required.

The client-server framework was used to develop the prototype of the IASO system, which is one of the contributions of this thesis. IASO clients, equipped with mobile devices able to browse the web, can access IASO HTTP servers, which provide the query-resolving services. The IASO system is described in Chapter 5.

[6] http://www.opera.com/products/mobile/

CHAPTER 4

A Context-Dependent Knowledge Representation Model for Knowledge Mobilization

This chapter proposes a solution for the problem of *information overload* in KMob systems based on using knowledge about the context of the user: a novel design pattern aimed to develop OWL ontologies explicitly representing which information from the application domain is significant in a given situation. We begin the chapter by introducing the incidence of information overload in KMob and some work related to our proposal. Then, we present the formal specification of the ontology design pattern, and describe the knowledge base that results from its application. The following section examines the properties of the pattern, including modularity of the resulting ontology. A complete algorithm to infer significant knowledge from the description of a context is also presented and discussed. Next, a software application which supports the creation of ontologies using this pattern is described. Finally, as an improvement to this pattern, we propose extending it by using fuzzy Description Logics, which allow the management of imprecise context and domain knowledge as well as the ranking of significance relations.

4.1 Rationale

Two main features of current enterprise information systems are connectivity and massiveness. Storages populated with Gbytes or even Tbytes of data are available across corporate networks, and allow users to access to complete, precise, and up-to-date information. The situation becomes more complicated when the Internet is considered because of the huge amount of valuable data that can be harvested from it. Corporative KBSs are expected to incorporate these several and probably heterogeneous information sources to provide valuable advice to decision makers.

As a result, functional KBSs usually manage so many resources when it comes to solving most requests that it is common for users, who access large-scale systems, to receive "excessive" information. Excessive means that either the time to filter it manually is extremely long or that simply it cannot be processed. In the literature this state is designated by the term *information overload*. It has also been called *data smog* [231], *analysis paralysis* [240], or *information fatigue syndrome* [196]. Information overload is described as a situation in which a user is provided with more data than he can digest, either because sifting through the information received would take too much time or simply because interesting facts cannot be separated from irrelevant data with the knowledge available [83, 85]. This results in unproductive decision processes and knowledge management failure. Solving information overload poses a great challenge for KBS, who must find the means to support the summarizing and customizing of information collected from massive, heterogeneous, and distributed sources depending on user needs. This objective can be achieved by adding metadata to information sources in order to delegate filtering tasks to automatic procedures [87].

Though few scientific studies have specifically addressed information overload issues in mobile systems (as argued in [4]), such issues are crucially important, and have been identified as a key factor for the acceptance of (knowledge-intensive) mobile services (as pointed out in [24]). Any KBS is expected to carry out tasks in consonance with user needs, and consequently,

to present only significant data. However, this requirement clearly becomes even more critical in KMob.

On the one hand, the scenario where a nomadic user requests system answers is completely different from the scenario where the user is stationary. KMob has to be able to handle dynamical decision processes. Such processes must often take place in real time and at the actual work site, where circumstances are extremely changeable, whereas classical support systems are usually exploited in more controlled situations. Therefore, the user cannot be completely focused on the device, and must not be required to search at length for the desired information. On the other hand, despite the fact that the computational power of mobile devices has continued to increase and the fact that wireless technologies provide Mb/s transfer speeds, device dimensions remain small. This is mandatory because of weight and battery life constraints. Reduced screens and adapted input interfaces make it difficult to cope with large datasets, but make it very easy to downgrade user experience. This means that succinctness is essential.

Thus, it naturally follows that mobile users cannot be overwhelmed with all the available data of the system nor can they be expected to manually review these results. On the contrary, it is crucial to provide nomadic users with data summaries that include only the most relevant fraction of the information generated. Consequently, the KMob architecture described in Chapter 3 must offer reasoning services able to compute summaries automatically.

These services must rely on suitable knowledge bases that represent *what* is *significant*. We strongly believe that what is significant (or relevant) depends on certain factors other than the query to be resolved. Such factors include user environment, preferences, previous actions, etc. All of these variables, referring to various circumstances of the query[1], embody what can be regarded as the *context* of the query. According to Dey and Abowd, context is any information (either implicit or explicit) that can be used to character-

[1]Although he was certainly not considering mobile knowledge-based systems, Spanish philosopher Ortega y Gasset (1883-1955) summarized this idea in his maxim “I am myself and my circumstances” (*Meditaciones del Quijote*, 1914).

ize the situation of an entity [79]. More details on context representation in UC are included in Section 4.2.

Consequently, a context-aware system must include two kinds of knowledge in its supporting knowledge bases, which are ontologies in KMob: (i) knowledge specific of the domain of the application; (ii) knowledge describing contextual situations. The significance of a piece of domain-specific information in a given context is represented by a relation between an ontological description of the situation and an ontological definition of the domain knowledge. Having a scenario description and a domain-specific expression connected with this type of significance relation means that this domain information must be considered because it is important in that situation.

In this chapter, we describe an innovative proposal of a design pattern aimed at representing in OWL ontologies this notion of context-dependent significance. The Context-Domain Significance (CDS, read as *Kodos*) pattern defines a set of rules to build a new ontology where context descriptions and domain expressions are connected through constrained relations. The main reasoning task within a CDS ontology is to retrieve the pieces of domain-specific knowledge that are significant in a given context. This task can be carried out with a corresponding algorithm.

A CDS ontology permits the description of which domain information is interesting in a scenario, but it does not measure how important the information is. This is very convenient in certain applications. Accordingly, we have developed an extension of the CDS design pattern which, relying on Fuzzy Description Logics, allows significance relations to be ranked. The extended pattern creates a fuzzy ontology where contexts descriptions, domain-specific knowledge expressions, and significance relations may be imprecise. This also makes it possible to rank the importance of a significance relation.

4.2 Related work

Ontology design patterns are simple recipes which help ontologists to capture and represent aspects of the application domain. In this section, we review some contributions on ontology design patterns that are related to our proposal. Since our design pattern is aimed at creating context-aware ontologies, two additional topics are addressed as well: context representation and contextualization of ontologies. In general, contextualization of ontologies consists in determining how additional knowledge influences the interpretation of an ontology (consistency, validity, partitioning, etc.). From this perspective, contextualization is related to context representation: context representation deals with environmental data management, whereas contextualization studies deal with how these context models affect the satisfiability of domain models. In this section, we discuss the literature on representation of context by means of ontologies in UC.

Ontology design patterns

Design patterns are concise guidelines which identify common design problems, and suggest how to resolve them. Patterns have been recognized as a valuable tool since the very beginning of design sciences from architecture [3] to software developmente [96]. Analysis and design patterns are important meta-artifacts that support the design process of software systems, as stressed in [126].

Acquiring, reusing, representing and eliciting knowledge to build an ontology is frequently an exhausting, time-consuming and frustrating experience even when collaborative experts, proper tools, and sound methodologies are used. Consequently, simple recipes which enable ontologists to better capture aspects of their application domain are greatly appreciated. Ontology design patterns are the extension of software design patterns. Their objective is to describe, more or less formally, recurrent modeling scenarios as well as to provide guidelines for correctly incorporating this knowledge

into ontologies. By *correctly*, we mean obtaining accurate, transparent, and reasonable representations [252].

Since the earliest work on the Semantic Web, ontology design patterns have been acknowledged as important contributions, which permit developers to save time when coping with the knowledge acquisition bottleneck. Templates to build knowledge bases have been proposed in several papers, some of which are specific for a concrete application domain (e.g. [219]), some of which are more general and (even) language-independent (e.g. [239]). The task force Ontology Engineering and Patterns [195] was created inside the W3C Semantic Web Best Practices and Deployment Working Group in order to elaborate best practices and patterns for OWL. The work of this task force was partially inspired by [191], a classical ontology-development guide which includes tips on how to build ontologies properly.

Research studies [252, 97, 38] provide a useful introduction to the use of design patterns during the ontology lifecycle. In the first study, Svátek analyzes several ontologies publicly available at Semantic Web repositories, and briefly presents some examples of patterns targeted at solving common representation mistakes. In the second paper, Gangemi describes the differences between software and ontology design patterns from a Semantic Web perspective, remarking that greater formality is required in the presentation of the CODePs (Conceptual Ontology Design Patterns), and provides some examples. In the third study, Blomqvist and Sandkuhl set out a classification of the patterns used in ontology development, and establish that ontology design patterns are those applied to create certain sections of the models. More recent contributions from these authors have proposed techniques for the automatic selection of suitable patterns to support ontology development [36, 37].

Context representation in UC

Broadly speaking, context is any information (implicit or explicit) that can be used to characterize the situation of an entity [79]. More specifically, context

is usually considered a mix of geo-spatial data, ambient sensor inputs, user profiles (preferences, intentions, etc.), and service descriptions [229].

Accordingly, context awareness is commonly defined as the ability to acquire, represent, and process environmental information in order to automatically adapt the behaviour of an application to user activity. Context-Aware Computing entails two activities: (i) interpreting the current user situation; (ii) using contextual knowledge to improve the performance of the system, for instance, helping to filter and tailor system responses. Context representation has been considered to play a key role in UC [258, 226], as explained in Section 2.4. The previously mentioned Context Toolkit is one of the first systematic approaches towards a generic framework for context-aware systems [80].

We use ontologies as the conceptual model to represent context knowledge in KMob. Not surprisingly, ontologies have been proposed as a design tool for modeling context in current pervasive systems since they offer several advantages over other formalisms: reusability, sharing, reasoning, standardization, supporting tools, etc. [57, 248]. In fact, significant progress was initially made in representing situational knowledge by means of formal theories, thanks to Levesque (inter alia) [162], one of the fathers of Description Logics. In the same regard, Situational Computing is claimed to be an extension of Context-Aware Computing which considers series of situations and past actions in the adjustment of system responses.

Recent proposals on UC which use ontologies, written in OWL or in one of its predecessors, are cited in Section 2.5 [7, 149, 66, 106, 139, 154, 158, 182, 263].

Contextualization of ontologies

Most of the work on contextualization concerns non-monotonicity in knowledge bases, in other words, reasoning with models which are satisfiable or not depending on the available knowledge [180]. Some early approaches

to the use of contexts in real Expert and Knowledge-Based Systems are described in [53]. For our purposes, contextualization is important because it determines how a knowledge base is interpreted depending on a concrete model, namely, the context knowledge model.

Interestingly enough, Description Logics and, extensively, ontologies, include a feature that makes them pretty much a monotonic formalism [49]. This feature is the open world assumption (see Section A.4). Some of the extensions proposed to allow non-monotonicity in ontology languages are the following: epistemic queries [136], default reasoning [148], circumscription [46], belief revision [95], and integration with logic Programming [181].

Other research works are focused on contextualization in the Semantic Web, and extending OWL with new operators to explicitly allow contextualization. For instance, Guha, McCool, and Fikes [107] examine seminal contributions on contexts and microtheories in Artificial Intelligence, and use some of these ideas to solve context-dependent aggregation problems in the Semantic Web. These authors use an ontology to define contexts which model implicit knowledge with a view to interpreting the right semantics of data sources. Essentially, a context associated to a source means that these data are true in that context.

Similarly, Bouquet et al. propose C-OWL [50], an extension of OWL to define mappings (via *bridge rules*) between locally-interpreted and globally-valid ontologies. Multi-viewpoint reasoning in [249] is a related approach which resembles to a certain extent our idea of significance. However, it concentrates on the conditional interpretation of a model (i.e. how to classify an ontology depending on the viewpoint submodel), whereas we focus on the relevance of the model (i.e. in which circumstances a submodel should be considered).

In the same regard, a review of different perspectives on the implementation of context-sensitivity is presented in [109]: C-OWL, ε-connections, Bayesian networks, probabilistic and possibilistic logics, etc. This work also cites context-based selection functions, which are very similar to our pro-

posal. These functions retrieve the submodel $K' \subset K$ that ought to be considered when performing some task. This report is a deliverable of the NeOn project, an on-going initiative that targets, among other problems, the handling of contexts with certain degree of uncertainty [211].

4.3 Definition

The Context-Domain Significance (CDS) design pattern is our proposal to represent relevance depending on context in KMob applications. The CDS pattern constructively states a set of rules to develop a new OWL ontology, the significance or CDS ontology. This ontology explicitly relates context descriptions, created by using a context vocabulary, with domain-specific expressions, which represent knowledge specific of the application domain. Given a CDS ontology, it is possible to infer which domain knowledge ought to be considered in a given situation by performing various subsumption tests.

The CDS ontology is the instantiation of the rules stated in the CDS pattern. Following [38], the formulation of the pattern in Description Logics notation can be interpreted as the semantic definition of the pattern, whereas the final OWL encoding is the syntactic expression of the pattern. We refer the reader to Appendix A for a review of ontology-related topics and Description Logics (DLs), upon which the remainder of this chapter is based.

To illustrate the definition of the CDS pattern, in this section we use the case study of Nomadic Healthcare presented in Section 2.3. We partially describe the ontology that supports a KMob system which delivers summaries of patients' electronic health records to a mobile doctor. The content of these summaries depends on the clinical situation of the patient who is being treated and is selected automatically.

Formulation

A CDS ontology is built from two basic subontologies, one representing domain-specific knowledge and another defining a vocabulary to describe context situations. These two subontologies may be related or even included in a more general ontology, but for the sake simplicity, we will operate on the assumption that they are disjoint. Both context and domain ontology should exist before applying the CDS pattern.

The domain-specific ontology contains the knowledge required to solve the concrete problem that the system is facing. In the KMob system for Nomadic Healthcare, this would be an ontology that abstractly represents which records are managed by the Hospital Information System, and which values have been stored for a patient. This ontology can be built by adapting some of the current proposals for a standard to represent the semantics of the contents of an HIS [84]. For instance, an OWL translation of openEHR could be adapted to our requirements (see Section 5.2).

Example 4.1 A domain-specific ontology in Nomadic Healthcare..
Concepts of the domain ontology may be `Patient`, `EHR` (Electronic Health Record[2]), `HDR` (Health Data Register), and `HDRDrugIntolerance` (Health Data Register of Drug Intolerance). Roles of this ontology can be `relatedToPatient`, from `Patient` to `EHR`, and `composedOf`, from `EHR` to `HDR`. Instances of this ontology are the actual values of patients' electronic health records.

This is an excerpt of the (simplified) ontology in DL syntax. It reflects that patient `juan` has related an `EHR` with a `HDR` stating that he has been diagnosed as having allergic reactions to Procaine. Moreover, other additional concepts describing different pieces of data stored in the HIS are defined.

$$\texttt{InformationUnit} \sqsubseteq \top$$

[2]An Electronic Health Record (EHR) is defined by the Health Information Management Systems Society's (HIMSS) as "a longitudinal electronic record of patient health information generated by one or more encounters in any care delivery setting. Included in this information are patient demographics, progress notes, problems, medications, vital signs, past medical history, immunizations, laboratory data, and radiology reports" [185].

$\texttt{Specialty} \sqsubseteq \top$
$\texttt{Patient} \sqsubseteq \top$
$\texttt{HDR} \sqsubseteq \texttt{InformationUnit}$
$\texttt{EHR} \sqsubseteq \forall\ \texttt{composedOf}.\texttt{HDR}$
$\texttt{EHR} \sqsubseteq \exists\ \texttt{relatedToPatient}.\texttt{Patient}$
$\texttt{HDRDrugIntolerance} \sqsubseteq \texttt{HDR}$
$\texttt{HDRProcaineIntolerance} \sqsubseteq \texttt{HDRDrugIntollerance}$
$\texttt{HDRPenicillinIntolerance} \sqsubseteq \texttt{HDRDrugIntollerance}$
$\texttt{HDRCoagulationDisorder} \sqsubseteq \texttt{HDR}$
$\texttt{HDRBloodPressureDisorder} \sqsubseteq \texttt{HDR}$

(jgomez : Patient)
(jgomezEHR1 : EHR)
((jgomezEHR1, juan) : relatedToPatient)
(positiveProcaine : HDRProcaineIntollerance)
((jgomezEHR1, positiveProcaine) : composedOf)

New concept definitions built from elements of the domain-specific ontology are called complex domains.

Definition 1. *Let $\mathcal{K}^D = \langle \mathcal{T}^D, \mathcal{R}^D, \mathcal{A}^D \rangle$ be a domain-specific ontology. A complex domain D_j is a concept such as $\mathbf{S}(D_j) \subseteq \mathbf{S}(\mathcal{K}^D)$.*

The context ontology contains knowledge suitable for the representation of circumstances or surroundings in which the domain knowledge is used. The context ontology can be regarded as a formal vocabulary or terminology to depict these situations. In the example, the context ontology contains terms describing the clinical situation of a patient (e.g. if he is unconscious; if he has a wound and where it is located; if his situation is serious; etc.). Thus, this is a medical ontology which can be built by specializing a general medical ontology, e.g. Galen [221].

Example 4.2 Context ontology in Nomadic Healthcare.
Concepts of the domain ontology for Nomadic Healthcare are `Unconsciousness`, `Laceration`, or `IntensityQualifier`. An important role is `hasIntensity`, which is used to express the degree of severity of a pathological process.

The following is an excerpt of the domain ontology. As can be observed, the knowledge base states some equivalences with the base ontology Galen.

$$\texttt{Unconsciousness} \sqsubseteq \texttt{galen:DisorderOfConsciousness}$$
$$\texttt{Hemorrhage} \equiv \texttt{galen:HaemorrhagingProcess}$$
$$\texttt{IrregularPenetrationWound} \equiv \texttt{galen:laceration}$$
$$\texttt{High} \sqsubseteq \texttt{IntensityQualifier}$$

New concept definitions built from elements of the context ontology are called complex contexts.

Definition 2. *Let $\mathcal{K}^C = \langle \mathcal{T}^C, \mathcal{R}^C, \mathcal{A}^C \rangle$ be a context ontology. A complex context C_i is a concept such as $\mathbf{S}(C_i) \subseteq \mathbf{S}(\mathcal{K}^C)$.*

It is important to notice that D_j and C_i are not part of the domain and the context ontology.

Actually, they are defined in the CDS ontology ($\mathcal{K}^S$), which is a new ontology where C_i, D_j, and *links* between them are defined. These links, called σ-connections (connections representing *sig*nificance), state that the domain-specific knowledge D_j should be considered in situation C_i. A σ-connection concept is a new concept representing a σ-connection, and is defined with existential restrictions on the complex context and the complex domain that it links (via properties R_c and R_d). When there is no possibility of confusion, we use the term σ-connection instead of σ-connection concept.

Definition 3. *Let $\mathcal{K}^D$ and $\mathcal{K}^C$ be, respectively, the domain and context ontologies, C_i a complex context, and D_j a complex domain. The significance or CDS*

ontology that relates the set of pairs (C_i, D_j) *(i.e., states that* D_j *is interesting when* C_i *occurs) is a consistent ontology* $\mathcal{K}^S = \langle \mathcal{T}^S, \mathcal{R}^S, \mathcal{A}^S \rangle$ *such that* $\mathcal{T}^S$ *(non-exclusively) includes definitions for the concepts* $P_\top, C_\top, D_\top, C_i, D_j, P_{i,j}$*, which satisfy:*

1. $P_\top$*,* $C_\top$*,* $D_\top$ *are the superclasses "σ-connection concePt', "complex Context" and "complex Domain":*
   - $P_{i,j} \sqsubseteq P_\top$*,* $C_i \sqsubseteq C_\top$*,* $D_j \sqsubseteq D_\top$
2. R_c *is the bridge property linking σ-connections and complex contexts:*
   - $P_\top \sqsubseteq \forall R_c.C_\top$
3. R_d *is the bridge property linking σ-connections and complex domains:*
   - $P_\top \sqsubseteq \forall R_d.D_\top$
4. $P_{i,j}$ *is the σ-connection linking the complex context* C_i *and the complex domain* D_j*:*
   - $P_{i,j} \equiv \exists R_c.C_i \sqcap \exists R_d.D_j$

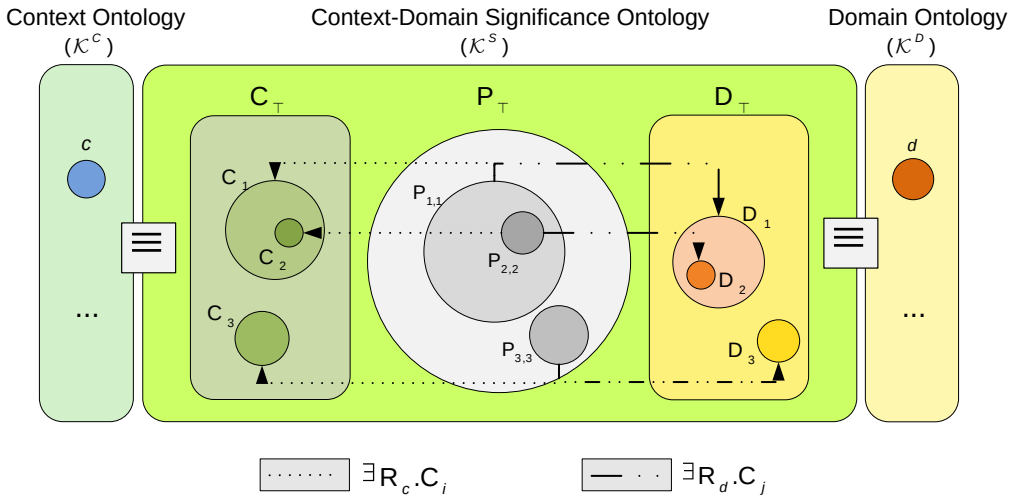
Figure 4.1 depicts the structure of a simple CDS ontology.

It can be observed that $\mathcal{K}^C$ and $\mathcal{K}^D$ are imported by $\mathcal{K}^S$. Since C_i and D_j are built from concepts defined in $\mathcal{K}^C$ and $\mathcal{K}^D$, respectively, these two ontologies must be incorporated into $\mathcal{K}^S$ to preserve their semantics. The consequences of this fact are discussed in Section 4.3.

Next, we present an example of a CDS ontology.

Example 4.3 A significance ontology for Nomadic Healthcare.
The CDS ontology $\mathcal{K}^S$ in this example must reflect which information from the HIS ought to be considered when treating a patient in a specific clinical situation. Let us suppose that `hasClinicalFact` and `hasElectronicRegister` are the bridge properties $\mathtt{R}_c$ and $\mathtt{R}_d$. Then, complex contexts $\mathtt{C}_i$, complex domains $\mathtt{D}_j$, and σ-connections between them can be defined as follows.

Figure 4.1: Schema of a CDS ontology



Top normative concepts

$\mathtt{P}_\top \sqsubseteq \top$

$\mathtt{C}_\top \sqsubseteq \top$

$\mathtt{D}_\top \sqsubseteq \top$

σ-connections

When the patient is "unconscious" and "hemorrhagic", it is necessary to check registers for "blood pressure disorders":

$\mathtt{C}_1 \equiv \mathtt{Unconsciousness} \sqcap \mathtt{Hemorrhage}$

$\mathtt{D}_1 \equiv \mathtt{HDRBloodPressureDisorder}$

$\mathtt{P}_{1,1} \equiv \exists\, \mathtt{hasClinicalFact}.\mathtt{C}_1 \sqcap \exists\, \mathtt{hasElectronicRegister}.\mathtt{D}_1$

When the patient is "unconscious", "hemorrhagic", and has a "penetrating wound", it is necessary to check registers for "drug intolerances":

$\mathtt{C}_2 \equiv \mathtt{Unconsciousness} \sqcap \mathtt{Hemorrhage} \sqcap$
$\mathtt{IrregularPenetrationWound}$

$\mathtt{D}_2 \equiv \mathtt{HDRDrugIntollerance}$

$\mathtt{P}_{2,2} \equiv \exists\, \mathtt{hasClinicalFact}.\mathtt{C}_2 \sqcap \exists\, \mathtt{hasElectronicRegister}.\mathtt{D}_2$

When the patient is "unconscious", with an "extremely serious" "hemorrhage", and has a "penetrating wound", it is necessary to check registers for "blood pressure disorders", "drug intolerances" , and "coagulation disorders".

$$\begin{aligned}
&\mathtt{C}_3 \equiv \mathtt{Unconsciousness} \sqcap \\
&\quad (\mathtt{Hemorrhage} \sqcap \exists\, \mathtt{hasIntensity.High}) \sqcap \\
&\quad \mathtt{PenetrationWound} \\
&\mathtt{D}_3 \equiv \mathtt{HDRBloodPressureDisorder} \sqcap \\
&\quad \mathtt{HDRDrugIntollerance} \sqcap \\
&\quad \mathtt{HDRCoagulationDisorder} \\
&\mathtt{P}_{3,3} \equiv \exists\, \mathtt{hasClinicalFact.C}_3 \sqcap \exists\, \mathtt{hasElectronicRegister.D}_3
\end{aligned}$$

It can be observed that $\mathtt{C}_3 \sqsubseteq \mathtt{C}_2 \sqsubseteq \mathtt{C}_1$ and $\mathtt{D}_1 \sqsubseteq \mathtt{D}_3$, $\mathtt{D}_2 \sqsubseteq \mathtt{D}_3$.

Properties

Inclusion of σ-connections

Proposition 1. *Let $\mathcal{K}^S$ be a CDS ontology, where C_i and $C_{i'}$ complex contexts defined in $\mathcal{T}^S$, and D_j are $D_{j'}$ complex domains defined in $\mathcal{T}^S$. The ontology $\mathcal{K}^S$ satisfies the following property:*

$$C_i \sqsubseteq C_{i'} \wedge D_j \sqsubseteq D_{j'} \Rightarrow P_{i,j} \sqsubseteq P_{i',j'}$$

This proposition reflects the intuition that if a context and a domain are connected through a σ-connection, more general (i.e. subsuming) contexts and domains will be connected through a more general σ-connection.

Proof. The proof is practically immediate from the fourth condition in Definition 3. Let us examine it in greater with more detail.

From the semantics of the full existential quantifier[3], given any model $\mathcal{I}$ of $\mathcal{K}$, $x \in (\exists R.C_1)^{\mathcal{I}} \leftrightarrow (\exists y \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C_1^{\mathcal{I}})$. Using this expression and the definition of the interpretation of a GCI ($C \sqsubseteq D \leftrightarrow C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$), we obtain: $y \in C^{\mathcal{I}} \Rightarrow y \in D^{\mathcal{I}} \Rightarrow (\exists y \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}) \Rightarrow x \in (\exists R.D)^{\mathcal{I}}$. Therefore, given an ontology $\mathcal{K} \models C \sqsubseteq D$ and any model $\mathcal{I}$ of $\mathcal{K}$, $(\exists R.C)^{\mathcal{I}} \subseteq (\exists R.D)^{\mathcal{I}}$ is true, or in other words, $C \sqsubseteq D \Rightarrow \exists R.C \sqsubseteq \exists R.D$.

This result is also valid for the significance ontology $\mathcal{K}^S$, and can be expressed in terms of R_c and R_d: $C_i \sqsubseteq C_{i'} \Rightarrow (\exists R_c.C_i)^{\mathcal{I}} \subseteq (\exists R_c.C_{i'})^{\mathcal{I}}$ and $D_j \sqsubseteq D_{j'} \Rightarrow (\exists R_d.D_j)^{\mathcal{I}} \subseteq (\exists R_d.D_{j'})^{\mathcal{I}}$. The application of the property of common sets $A_1 \subseteq A_2 \wedge B_1 \subseteq B_2 \Rightarrow A_1 \cap B_1 \subseteq A_2 \cap B_2$ and the definition of the interpretation of a GCI give the following: $(\exists R_c.C_i)^{\mathcal{I}} \cap (\exists R_d.D_j)^{\mathcal{I}} \subseteq (\exists R_c.C_{i'})^{\mathcal{I}} \cap (\exists R_d.D_{j'})^{\mathcal{I}} \leftrightarrow (\exists R_c.C_i) \sqcap (\exists R_d.D_j) \sqsubseteq (\exists R_c.C_{i'}) \sqcap (\exists R_d.D_{j'})$. This finally gives: $(\exists R_c.C_i) \sqcap (\exists R_d.D_j) \equiv P_{i,j} \sqsubseteq P_{i',j'} \equiv (\exists R_c.C_{i'}) \sqcap (\exists R_d.D_{j'})$.

□

In general, the reciprocal is not true. This formulation of the pattern allows a consistent significance ontology to be created with $P_{i,j} \sqsubseteq P_{i',j'}$, but $C_i \not\sqsubseteq C_{i'}$ and/or $D_j \not\sqsubseteq D_{j'}$. As this can be useful in some applications, further requirements in the pattern can be stated to ensure this property.

Definition 4. *A hierarchical significance ontology is a CDS ontology $\mathcal{K}^S$ which additionally satisfies:* $P_{i,j} \sqsubseteq P_{i',j'} \Rightarrow C_i \sqsubseteq C_{i'} \wedge D_j \sqsubseteq D_{j'}$

This property means that in a hierarchical relevance ontology, if we define a σ-connection subsumed by another σ-connection ($P_{i,j} \sqsubseteq P_{i',j'}$), the context and the domain thus linked (C_i, D_j) are subclasses of the context and domain ($C_{i'}, D_{j'}$) related by the super σ-connection.

The procedure used to prove if a CDS ontology is hierarchical is straightforward: for each pair of σ-connections, subsumption between the linked context and domain must be checked. When developing a new CDS ontology, this test can be performed for every new profile with a view to: (i)

[3] See Section A.3 for a detailed explanation on the semantics of DL constructs and axioms

detecting possible contradictions with Definition 4; (ii) displaying advice for the ontologist; (iii) adding suitable axioms. Alternatively, it is also possible to add further restrictions to our definition of significance ontology which force the resulting CDS ontology to be hierarchical.

Proposition 2. *The following additional restrictions are sufficient to ensure that a significance ontology is hierarchical:*

1. R_c *and* R_d *are functional*
2. R_c^- *and* R_d^- *are functional*
3. *A complex context is involved in a* σ*-connection:* $C_i \sqsubseteq \exists R_c^-.P_{i,j}$
4. *A complex domain is involved in a* σ*-connection:* $D_j \sqsubseteq \exists R_d^-.P_{i,j}$

Proof. The conditions can be proved to be sufficient by contradiction. Let us assume a significance ontology $\mathcal{K}^S$ satisfying Proposition 2, a model $\mathcal{I}$ of $\mathcal{K}^S$ and an instance c such as $c \in C_i^{\mathcal{I}}$ and $c \notin C_{i'}^{\mathcal{I}}$.

Because to restriction 3 (and 2), there is (only) one $p \in P_{i,j}^{\mathcal{I}}$ such as $(c,p) \in R_c^{-\mathcal{I}} \Rightarrow (p,c) \in R_c^{\mathcal{I}}$.

By hypothesis, $P_{i,j} \sqsubseteq P_{i',j'}$, so $p \in P_{i',j'}^{\mathcal{I}}$. By definition, $P_{i',j'} \equiv \exists R_c.C_{i'} \sqcap \exists R_d.D_{j'}$, which implies that $(p,\ c') \in R_c^{\mathcal{I}}$, where $c' \in C_{i'}^{\mathcal{I}}$.

Therefore, we have $(p,c) \in R_c^{\mathcal{I}}$ and $(p,c') \in R_c^{\mathcal{I}}$. Due to restriction 1, R_c is functional, so necessarily $c = c'$. Consequently, $c \in C_{i'}^{\mathcal{I}}$, which is a contradiction of the initial assumption. No individual satisfying $c \in C_i$ and $c \notin C_{i'}$ can exist and, as a result, $C_i \sqsubseteq C_{i'}$.

The counterpart for $D_j \sqsubseteq D_{j'}$ is immediate. □

Modularity

The CDS pattern promotes ontology modularization since it clearly separates the context, domain, and significance models. Nevertheless, the significance

ontology $\mathcal{K}^S$ must completely import $\mathcal{K}^C$ and $\mathcal{K}^D$. This is necessary because $\mathcal{K}^S$ contains the definitions of complex contexts C_i and domains D_j, which are built by relying on the concepts of $\mathcal{K}^C$ and $\mathcal{K}^D$, respectively. Moreover, partial inclusion is not allowed by the `import` directive of OWL.

Arbitrary definitions of C_i and D_j in $\mathcal{K}^S$ may lead to undesired inferences in $\mathcal{K}^C$ and $\mathcal{K}^D$. For instance, for two concepts A and B defined in $\mathcal{K}^C$, the following situation is possible:

$$C_1 \equiv A$$
$$C_2 \equiv B$$
$$C_2 \sqsubseteq C_1$$
$$\models B \sqsubseteq A$$

The axiom $B \sqsubseteq A$, which may not be asserted in $\mathcal{K}^C$, is obtained as a result of the definitions of C_1 and C_2. This is not a desirable inference because the significance ontology is not expected to change the meaning of the context and domain ontologies, which are external models, and possibly managed by other organizations. In the remainder of this section, we focus on the context ontology $\mathcal{K}^C$. The following considerations can be directly extended to $\mathcal{K}^D$.

Undesired inferences frequently appear in ontology reutilization. This must be avoided, and ontologies should be safe to reuse. The basic idea underlying the safe reuse of ontologies is that the importing ontology must not add new knowledge that changes the meaning of the concepts defined in the imported ontology. In our case, the definitions of the complex contexts C_i must not imply new axioms that modify the semantics of the concepts in $\mathcal{K}^C$.

This property is formalized in [73, 74] by applying the notion of conservative extension of a TBox, which is defined in [101]. Given two TBoxes $\mathcal{T}$ and $\mathcal{T}'$, $\mathcal{T} \cup \mathcal{T}'$ is a conservative extension of $\mathcal{T}'$ if, for every axiom α with $\mathbf{S}(\alpha) \subseteq \mathbf{S}(\mathcal{T}')$ we have $\mathcal{T} \cup \mathcal{T}' \models \alpha$ iff $\mathcal{T}' \models \alpha$. In our case, for the significance ontology to be safe, $\mathcal{K}^S \cup \mathcal{K}^C$ must be a conservative extension of $\mathcal{K}^C$.

The authors improve this approach and study safety without taking into

account the whole imported ontology, but rather only the imported symbols (i.e. the external signature). Thus, an ontology $\mathcal{K}$ is safe for a signature S (w.r.t. the DL in which $\mathcal{K}$ is expressed) if for every ontology $\mathcal{K}'$ with $\mathbf{S}(\mathcal{K}) \cap \mathbf{S}(\mathcal{K}') \subseteq$ S, $\mathcal{K} \cup \mathcal{K}'$ is a conservative extension of $\mathcal{K}'$.

In [74], it is proved that detecting if an ontology $\mathcal{K}$ is a safe extension of a submodel $\mathcal{K}'$ (with $\mathcal{K}' \subseteq \mathcal{K}$) is 2NExpTime for $\mathcal{ALCIQ}$, and undecidable for $\mathcal{ALCIQO}$. Consequently, this task is also undecidable for OWL DL. However, sufficient conditions are stated to guarantee that the union of the importing ontology and the imported ontology is a safe extension of the imported ontology. These conditions, which guarantee syntactic locality, are syntactic restrictions on the possible axioms involving imported symbols. As proved in the paper, syntactic locality implies locality, and locality implies safety. In our case, we must assure the syntactic locality of C_i and D_j definitions, which will result in the significance ontology $\mathcal{K}^S$ being safe.

Recently, an methodology for ontology reuse based on these results has been proposed [130]. To avoid undesired collateral effects on the imported ontologies, the design of a CDS ontology can follow this procedure. Additionally, this methodology applies the idea of module (also described in [74]). A module is the minimum subset of axioms of the imported ontology that must be incorporated into the importing ontology to be able to infer the same knowledge that could have been inferred if the whole ontology had been added. In this way, the CDS ontology would only contain the minimal number of axioms from the context and the domain ontologies. This would improve the performance of the reasoning procedure as well.

Other features

- *Reusability*. By definition, design patterns must be applicable to different problems and domain areas. Our pattern effectively fulfills this objective since it provides a general guideline for representing significance without imposing application-dependent restrictions on the domain and the context ontologies.

- *Standardization*. One of the main prerequisites for the CDS pattern was OWL-DL compliance. In other words, the resulting ontology should not include new constructors nor be in OWL-Full. As explained, the pattern generates a new OWL-DL ontology, whose complexity is bounded by context and domain models. Thus, though the reasoning process may seem somewhat less than straightforward, current tools (i.e. DL inference engines) can be directly used, without having to extend, modify, or re-implement them.

- *Expressivity*. The pattern allows significance to be represented, and makes the most of OWL expressivity. For instance, σ-connection hierarchies can be defined to assert inclusion relations between them. Actually, the resulting model is an OWL ontology, and can be modified as needed. Further improvements may be easily incorporated, e.g. definition of several bridge properties with different semantics to qualify the connections between contexts and domain, or the addition of properties to profile classes.

4.4 Context-aware reasoning

The main reasoning task involving a significance ontology consists in finding all the concepts in the domain ontology which ought to be considered in a given context. In this section, we describe a complete and decidable algorithm that performs this operation by carrying out basic DL inference tasks.

Formulation

Definition 5. *Let $\mathcal{K}^D$ be a domain-specific ontology, $\mathcal{K}^C$ a context ontology, and $\mathcal{K}^S$ a CDS ontology, with their respective signatures $\mathbf{S}(\mathcal{K}^D)$, $\mathbf{S}(\mathcal{K}^C)$, and $\mathbf{S}(\mathcal{K}^S)$. Let scenario E be a concept $E \in \mathbf{S}(\mathcal{K}^C)$. The domain knowledge in $\mathcal{K}^D$ that is significant in the scenario E w.r.t. $\mathcal{K}^S$, denoted as $\mathcal{D}(E, \mathcal{K}^S)$, is a set which includes the concepts I such that:*

$$\mathcal{D}(E, \mathcal{K}^S) = \{I \mid I \in \mathsf{Con}(\mathcal{K}^D) \wedge \mathcal{K}^S \models \{E \sqsubseteq C_n, P_{n,m} \sqsubseteq P_\top, I \sqsubseteq D_m\}\}$$

Essentially, we are interested in the concepts of the domain ontology that:

- are similar to the complex domains
- asserted to be significant to the complex contexts
- which are similar to our query.

In this case, similarity is computed as concept inclusion[4]. Thus, I stands for the concepts I defined in the domain ontology that are more specific than the complex domains D_j which are σ-connected to the contexts C_i more general than E.

The domain significant in a scenario can be retrieved by performing several subsumption tasks:

Algorithm 1. *$\mathcal{D}(E, \mathcal{K}^S)$ can be computed in practice as follows:*

1. $\{C_n\} = \{C_n \sqsubseteq C_\top \mid E \sqsubseteq C_n\}$
2. $\{P_{k,l}\} = \{P_{k,l} \sqsubseteq P_\top \mid (P_{k,l} \sqsubseteq \exists R_c.C_k) \wedge (C_k \equiv C_n)\}$
3. $\{D_m\} = \{D_m \sqsubseteq D_\top \mid (P_{k,l} \sqsubseteq \exists R_d.D_l) \wedge (D_m \equiv D_l)\}$
4. $\mathcal{D}(E, \mathcal{K}^S) = \{I \in \mathsf{Con}(\mathcal{K}^D) \mid I \sqsubseteq D_m\}$

The output of the algorithm is the set $\{I\}$ of simple domain concepts of $\mathcal{K}^D$ which ought to be considered in the query context E.

The reason to retrieve the $I \in \mathsf{Con}(\mathcal{K}^D)$ and not simply D_m is the following. We assume that the knowledge about the domain is only in $\mathcal{K}^D$, and D_m are constructions built for the sake of convenience. Thus, it may occur that a

[4]Interestingly enough, other similarity measures could be implemented. Measurement of similarities between ontologies is a topic that has been widely studied in the literature, particularly in the biomedical domain [166].

D_m has no direct correspondence with a concept in the domain. Hence, the algorithm obtain the concepts of the domain that are similar (i.e. included) in D_m.

Since the definition of the significance ontology does not allow σ-connections involving anonymous contexts or domains to be created, steps 2 and 3 of the algorithm could be simplified as follows:

- $\{P_{k,l}\} = \{P_{k,l} \sqsubseteq P_\top \mid (P_{k,l} \sqsubseteq \exists R_1.C_k) \wedge (C_n \in \{C_k\})\}$
- $\{D_m\} = \{D_m \sqsubseteq D_\top \mid (P_{k,l} \sqsubseteq \exists R_2.D_l) \wedge (D_m \in \{D_l\})\}$

Next, we present an example of the algorithm.

Example 4.4 Domain significant to a context in Nomadic Healthcare. Given the σ-connections in Example 4.3, the question arises regarding which information of the HIS should be checked if the doctor is treating a "hemorrhagic and unconscious patient with a penetration wound". This is accomplished by using Algorithm 1 to calculate the restricted domain of this complex context concept.

$E \equiv \texttt{Hemorrhage} \sqcap \texttt{Unconsciousness} \sqcap \texttt{PenetrationWound}$

1. $\texttt{C}_n = \{\texttt{C}_1, \texttt{C}_2\}$
2. $\texttt{P}_{k,l} = \{\texttt{P}_{1,1}, \texttt{P}_{2,2}\}$
3. $\texttt{D}_m = \{\texttt{D}_1, \texttt{D}_2\}$
4. I= { HDRBloodPressureDisorder,
   HDRDrugIntollerance,
   HDRProcaineIntolerance,
   HDRPenicillinIntolerance}

Once it is known which HDRs from the HIS are interesting, the concrete values for the patient may be retrieved. In this case, only records for

HDRProcaineIntollerance are available for the patient jgomez. So, the doctor will be reminded that patient jgomez is positiveProcaine.

■

The example shows that descendants of E are not considered during the reasoning process. These concepts correspond to more specific context situations, which will probably lead to more specialized domain information. However, it may be interesting to calculate the σ-connections involving these subcontexts, and to provide them as feedback information to the user, who may be recommended to describe further details of the current scenario.

Example 4.5 Additional results in Nomadic Healthcare.
In this example, $\mathtt{C}_3$ in $\mathtt{P}_{3,3}$ is subsumed by E:

$$\begin{aligned}
&\mathtt{C}_3 \equiv \mathtt{Unconsciousness} \sqcap (\mathtt{Hemorrhage} \sqcap \exists\, \mathtt{hasIntensity}.\mathtt{High}) \sqcap \\
&\quad \mathtt{IrregularPenetrationWound} \\
&\sqsubseteq \\
&\quad \mathtt{Unconsciousness} \sqcap \mathtt{Hemorrhage} \sqcap \\
&\quad \mathtt{IrregularPenetrationWound} \\
&\equiv E
\end{aligned}$$

Consequently, the doctor could be advised to carry out other clinical trials to see if the specific part of this restriction ($\exists$ hasIntensity.High qualifier of Hemorrhage) is present, but has not been diagnosed as yet. If this knowledge is subsequently supplied, more information about the patient (HDRCoagulationDisorder) will be considered significant.

■

Properties

Decidability and completeness

Proposition 3. *Algorithm 1 is decidable and complete, i.e. it finds all the concepts I related to the query concept E through σ-connections.*

Proof. From the expressions in steps 1-4 of Algorithm 1, trivially every $P_{k,l}$ subsuming the (hypothetical) σ-connection $P_{E,I}$ linking E and I is retrieved. By definition, $E \sqsubseteq C_n \sqsubseteq C_k \ \forall n,k$ and $I \sqsubseteq D_m \sqsubseteq D_l \ \forall m,l$. According to Proposition 1, we directly obtain that $P_{E,I} \sqsubseteq P_{n,m} \sqsubseteq P_{k,l}$. □

Computational complexity

The definition of the CDS ontology entails that $P_{i,j}$ definitions are (at most) in the DL $\mathcal{ALC}$ (in fact, they are in $\mathcal{ALE}$)[5]. Therefore, the computational complexity of the CDS model is mainly conditioned by the complexity of C_i and D_j expressions, the other concepts defined in the CDS ontology. The complexity of these concepts subsequently depends on the complexity of the ontologies $\mathcal{K}^C$ and $\mathcal{K}^D$, which contains the definitions of the terms used to build C_i and D_j, and must be imported by $\mathcal{K}^S$.

In the simplest case (i.e. $\mathcal{K}^C$, $\mathcal{K}^D$ and $\mathcal{K}^S$ ontologies are in $\mathcal{ALC}$), reasoning within the CDS ontology is asymptotically bounded by concept subsumption complexity, which is ExpTime-complete for $\mathcal{ALC}$ with GCIs, according to Table A.4.

If we suppose that $\mathcal{K}^C$ and $\mathcal{K}^D$ do not add further complexity, it is possible to reduce the complexity of $\mathcal{K}^D$ by restricting the allowed expressions for C_i and D_j, moving consequently to a less expressive logic (Table 4.6 [58]). Restricting negation to atomic concepts and disallowing union concepts would enclose the CDS ontology to $\mathcal{ALE}$, which has PSpace complexity for general

[5]Notice that if the conditions for hierarchical CDS ontologies are assumed (Proposition 2), the complexity increases to $\mathcal{ALCIF}$

reasoning. Another alternative consists on using only acyclic TBoxes, which would give complexities of PSPACE for $\mathcal{ALC}$ and coNP for $\mathcal{ALE}$.

Other choices are not appropriate, however. Moving from $\mathcal{ALC}$ to $\mathcal{ALU}$ does not reduce the complexity, either in the general case or with acyclic TBoxes. Moving to $\mathcal{AL}$ is not possible, because existential quantification cannot be restricted. Similarly, expressivity of $\mathcal{FL}^-$ is too limited.

Table 4.6: Complexity of reasoning with TBoxes in basic DLs

DL \ TBox	Acyclic	General
$\mathcal{FL}^-$	PTIME	PTIME
$\mathcal{AL}$	coNP	PSPACE
$\mathcal{ALE}$	coNP	PSPACE
$\mathcal{ALU}$	PSPACE	EXPTIME
$\mathcal{ALC}$	PSPACE	EXPTIME

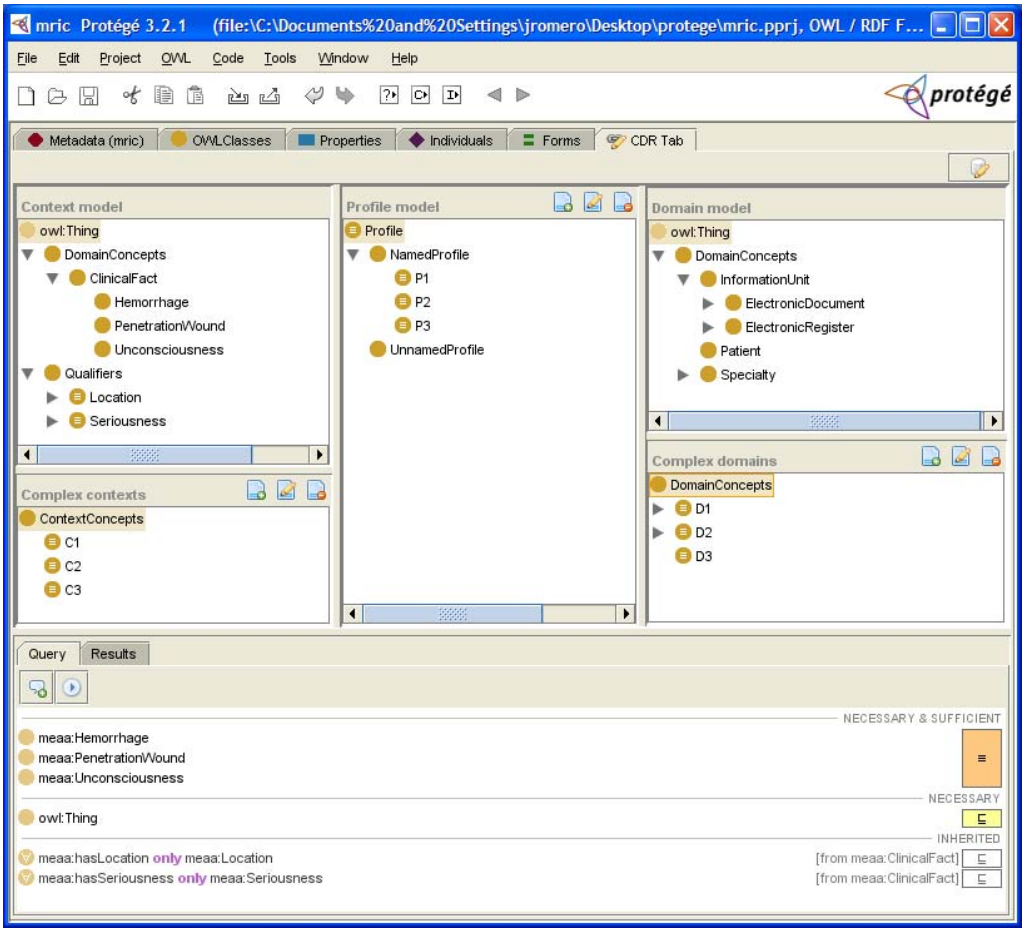
According to this formulation of the CDS pattern, role hierarchies are not necessary. Nevertheless, they may be considered for the sake of convenience, in such a way that subroles of R_c and R_d can be defined with particular semantics and handled in consequence. This increases the complexity to $\mathcal{ALCH}$, but with the advantage that reasoning for the general case still remains EXPTIME.

In any case, since $\mathcal{ALCH}$ and $\mathcal{ALCIF}$ are less expressive than $\mathcal{SHIF}(\mathcal{D})$ (equivalent to OWL-Lite), reasoning in practice with available DL engines (e.g. Pellet) will be quite efficient as they are highly optimized and worst-case inferences are infrequent. Hence, more complex logics with extended semantics could be considered as well to extend the basic formulation without significant performance impact.

4.5 CDS Plug-in for Protégé

We have developed a plug-in for the Protégé platform, which allows users to create, edit, test and reason with a CDS ontology. Our plug-in adds a new tab

Figure 4.2: CDS Tab plug-in in Protégé IDE



to the Protégé-OWL environment (Protégé enhanced with the OWL plug-in) where a simplified view of the CDS ontology is displayed and queries can be introduced. A preliminary version can be found in `http://decsai.ugr.es/~jgomez/thesis/cdsplugin`. The plug-in is based on the CDS-API, a library to programmatically manage models created with the pattern, which can also be downloaded.

We can distinguish four sections in the tab, depicted in Figure 4.2:

1. *Context section*. The left side of the tab shows the context ontology ($\mathcal{K}^C$) and the complex contexts (C_i) in the CDS ontology. The con-

text ontology can be optionally hidden. New complex contexts can be created by using the context vocabulary; existential restrictions for the new C_i are automatically added. It is also possible to edit or delete existing contexts.

2. *Domain section*. The right side mirrors the context side, but instead of the context ontology, the domain ontology ($\mathcal{K}^D$) is presented. New complex domains D_i can also be easily created, and editing and deleting are allowed as well .

3. σ*-connections*. The central section of the tab shows the σ-connections in the ontology $\mathcal{K}^S$. This is probably the most interesting part, since it simplifies the task of creating new σ-connections. To build a σ-connection, the user has only to select a complex context in the left box (C_i) and a complex domain in the right box (D_j), and then push the 'new connection' button. A new σ-connection ($P_{i,j}$) will be created as a subclass of the currently selected σ-connection, and the corresponding existential restrictions will be automatically generated.

4. *Reasoning*. The bottom section allows the retrieval of domain knowledge significant to a given context, i.e. it implements Algorithm 1. When a new complex concept for querying is created, its restrictions are shown and the 'run query' button is activated. Results are displayed in the 'Results' tab of this reasoning section and additional information about the obtained classes can be consulted.

From the formal description of the pattern, it can be deduced that some additional configuration is needed to make the CDS plug-in work correctly: it is necessary to state the URIs of the external ontologies ($\mathcal{K}^C$, $\mathcal{K}^D$); the top concepts for the σ-connections, the complex contexts, and the complex domains ($P_\top$, $C_\top$, $D_\top$); and the URL of the DIG reasoner to be used. To assist this procedure, a wizard-like window is presented to the user when the 'properties' button on the top toolbar is clicked.

The plug-in has been developed with the CDS-API and the APIs for Protégé and Protégé-OWl version 3.2.1. Installation is very simple. As any other Protégé add-on, it just has to be copied to the plug-in directory of the Protégé installation.

4.6 A fuzzy extension of the CDS pattern

Rationale

The significance ontology resulting from applying the CDS pattern has two main deficiencies.

Firstly, definitions of complex context concepts C_i (respectively for definitions of complex domain concepts D_i) are crisp. As a result, it is not possible to directly represent vague contexts, e.g. "the patient is slightly unconscious", and partial similarity between contexts, e.g. "anaphylaxis is quite similar to sepsis".

The second problem is that although the significance ontology allows the developer to assert which domain-specific knowledge is interesting in a scenario, it does not measure how important this connection is, which is desirable in some applications. For instance, in our example on Nomadic Healthcare, electronic registers about previous adverse drug events are more important than others, and should be presented firstly to the doctor because avoiding an anaphylactic shock is a major priority in healthcare. Ranking the relevance relations would allow system responses to be sorted by precedence and a threshold to be fixed in order to retrieve only the top k most relevant concepts of the domain ontology.

In this section, we propose an extension of the CDS design pattern to: (i) deal with vague complex contexts and domains; (ii) quantify the importance of σ-connections. Our approach relies on Fuzzy Description Logics (fuzzy DLs), a logical formalism which combines Fuzzy Logic theory and Description Logics, and defines a sound framework to represent and reason with

imprecise and vague knowledge in ontologies. A brief introduction to fuzzy DLs and further references are provided in Section A.5.

The extension of the CDS pattern results in a fuzzy ontology. This extension allows fuzzy context and domain descriptions to be represented, and significance relations to hold to a degree. A suitable reasoning algorithm to retrieve the domain knowledge relevant in a given context is also proposed. Interestingly enough, though the fuzzy CDS ontology is not OWL compliant, previous results can be applied to reduce it to a crisp representation in order to use existing inference engines [39, 41, 40].

Definition

The fuzzy CDS ontology extends the original proposal by allowing contexts, domains, and σ-connections to be defined by means of fuzzy GCIs. Thus, complex contexts (respectively complex domains) can be stated to be partially similar. Moreover, the degree of subsumption in a σ-connection definition represents the importance value of the link between the context and the domain.

Definition 6. *Let $\mathcal{K}^D$ and $\mathcal{K}^C$ be, respectively, the domain and the context ontologies; C_i a fuzzy complex context; and D_j a fuzzy complex domain. The fuzzy significance or fCDS ontology which relates the set of pairs (C_i, D_j) with degree $\alpha_{i,j}$ (i.e. states that D_j is interesting with rank $\alpha_{i,j}$ when C_i happens) is a consistent fuzzy ontology $f\mathcal{K}^S = \langle \mathcal{T}^S, \mathcal{R}^S, \mathcal{A}^S \rangle$ such that $\mathcal{T}^S$ (non-exclusively) includes definitions for the fuzzy concepts $P_\top, C_\top, D_\top, C_i, D_j, P_{i,j}$, which satisfy:*

1. *$P_\top, C_\top, D_\top$ are the superclasses 'top σ-connection concePt', 'top complex Context', and 'top complex domain':*
   - *$\langle P_{i,j} \sqsubseteq_{\geq 1} P_\top \rangle$, $\langle C_i \sqsubseteq_{\geq 1} C_\top \rangle$, $\langle D_j \sqsubseteq_{\geq 1} D_\top \rangle$*
2. *R_c is the (fuzzy) bridge property linking σ-connections and complex contexts:*

- $\langle P_\top \sqsubseteq_{\geq 1} \forall R_c.C_\top \rangle$

3. *R_d is the (fuzzy) bridge property linking σ-connections and complex domains:*

   - $\langle P_\top \sqsubseteq_{\geq 1} \forall R_d.D_\top \rangle$

4. *$P_{i,j}$ is the (fuzzy) σ-connection concept linking the complex context C_i and the complex domain D_j:*

   - $\langle P_{i,j} \sqsubseteq_{\geq \alpha_{i,j}} \exists R_c.C_i \sqcap \exists R_d.D_j \rangle$

It is interesting to note that the context ontology $\mathcal{K}^C$ and the domain ontology $\mathcal{K}^D$ may be fuzzy or not. However, both C_i and D_j are fuzzy concepts because they are defined with fuzzy GCIs. Next, we show a fuzzy relevance ontology built from two crisp ontologies $\mathcal{K}^C$ and $\mathcal{K}^D$.

Example 4.6.
Continuing our example, we can use the same crisp domain and context ontologies described in Examples 4.1 and 4.2. Some axioms which were not included in these examples are presented below.

Crisp axioms which extend the domain ontology $\mathcal{K}^D$:
`HDRCurrentPrescription` $\sqsubseteq$ `HDR`
`HDRAntidepressives` $\sqsubseteq$ `HDRCurrentPrescription`
`(jgomezEHR2 : EHR)`
`((jgomezEHR2, jgomez) : relatedToPatient)`
`(prozac: HDRAntidepressives)`
`((jgomezEHR2, prozac) : composedOf)`

Crisp axioms which extend the context ontology $\mathcal{K}^C$:
`Anaphylaxis` $\equiv$ `galen:Anaphylaxis`
`Shock` $\equiv$ `galen:Shock`
`Elderly` $\equiv \exists$`hasAge.trap{60, 75, 120, 120}`
`EpinephrineAdmin` $\sqsubseteq$ `galen:Treatment`

The CDS ontology $f\mathcal{K}^S$ contains definitions for C_i, D_j, and σ-connections that were created with fuzzy GCIs. In this case, fuzzy C_i denote vague descriptions of patient situations. D_j are also fuzzy concepts, but they are assigned a crisp semantics, since they correspond to the pieces of information represented in the HIS.

Additionally, some new fuzzy axioms are introduced to state similarities between context concepts. These axioms could be introduced in $\mathcal{K}^C$ to make it into a fuzzy ontology, but we have preferred to maintain the context and the domain base ontologies unaltered.

An excerpt of the (simplified) $f\mathcal{K}^S$ is shown below.

Axioms which extend the context ontology

$\langle \texttt{Anaphylaxis} \sqsubseteq_{\geq 0.7} \texttt{Shock} \rangle$

$\langle \texttt{SepticShock} \sqsubseteq_{\geq 0.5} \texttt{Anaphylaxis} \rangle$

$\langle \texttt{Shock} \sqcap \geq 1 \texttt{hasComplication} \sqsubseteq_{\geq 0.8} \texttt{EpinephrineAdmin} \rangle$

Definition of complex contexts

$\langle \texttt{C}_1 \sqsubseteq_{\geq 1} \exists\, \texttt{hasComplication.Elderly} \rangle$

$\langle \texttt{C}_2 \sqsubseteq_{\geq 1} \texttt{Anaphylaxis} \rangle$

$\langle \texttt{C}_3 \sqsubseteq_{\geq 1} \texttt{EpinephrineAdmin} \rangle$

Definition of complex domains

$\langle \texttt{D}_1 \sqsubseteq_{\geq 1} \texttt{EHRCurrentPrescription} \rangle$

$\langle \texttt{D}_2 \sqsubseteq_{\geq 1} \texttt{EHRCurrentPrescription} \sqcup \texttt{EHRDrugIntollerance} \rangle$

$\langle \texttt{D}_3 \sqsubseteq_{\geq 1} \texttt{EHRAntidepressives} \rangle$

$\langle \texttt{D}_3 \sqsubseteq_{\geq 1} \texttt{D}_1 \rangle$

Definition of relations (for the sake of convenience)

$\langle \texttt{R}_a \equiv \texttt{relSymptom} \rangle$

$\langle \texttt{R}_b \equiv \texttt{relRegister} \rangle$

Definition of σ-connections

$\langle \texttt{P}_{1,1} \sqsubseteq_{\geq 0.6} \exists\, \texttt{R}_c.\texttt{C}_1 \sqcap \exists \texttt{R}_d.\texttt{D}_1 \rangle$

$\langle \mathtt{P}_{2,2} \sqsubseteq_{\geq 0.5} \exists\ \mathtt{R}_c.\mathtt{C}_2 \sqcap \exists \mathtt{R}_d.\mathtt{D}_2 \rangle$
$\langle \mathtt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists\ \mathtt{R}_c.\mathtt{C}_3 \sqcap \exists \mathtt{R}_d.\mathtt{D}_3 \rangle$

Mandatory axioms
$\langle \mathtt{C}_1 \sqsubseteq_{\geq 1} \mathtt{C}_\top \rangle$, $\langle \mathtt{C}_2 \sqsubseteq_{\geq 1} \mathtt{C}_\top \rangle$,
$\langle \mathtt{D}_1 \sqsubseteq_{\geq 1} \mathtt{D}_\top \rangle$, $\langle \mathtt{D}_2 \sqsubseteq_{\geq 1} \mathtt{D}_\top \rangle$,
$\langle \mathtt{P}_{1,1} \sqsubseteq_{\geq 1} \mathtt{P}_\top \rangle$, $\langle \mathtt{P}_{2,2} \sqsubseteq_{\geq 1} \mathtt{P}_\top \rangle$,
$\langle \mathtt{P}_\top \sqsubseteq_{\geq 1} \forall R_c.\ \mathtt{C}_\top \rangle$, $\langle \mathtt{P}_\top \sqsubseteq_{\geq 1} \forall R_d.\ \mathtt{D}_\top \rangle$

Fuzzy context-aware reasoning

Formulation

As an extension of the crisp case, the domain knowledge significant in a context w.r.t. a fuzzy relevance ontology contains all the (fuzzy or crisp) concepts of the domain sub-ontology which are relevant in a given (fuzzy) context, and the degree of interest. It is formally defined as a set of pairs (concept of the domain ontology, degree), where "domain concepts" are the concepts of the domain ontology which are significant to the context description introduced as input of the query, and "degree" is a number computed on $\alpha_{i,j}$ values.

Definition 7. *Let $\mathcal{K}^D$ be a domain-specific ontology, $\mathcal{K}^C$ a context ontology, and $f\mathcal{K}^S$ a fuzzy CDS ontology, with their respective signatures $\mathbf{S}(\mathcal{K}^D)$, $\mathbf{S}(\mathcal{K}^C)$, and $\mathbf{S}(f\mathcal{K}^S)$. Let scenario E be a concept $E \in \mathbf{S}(\mathcal{K}^C)$. The domain knowledge in $\mathcal{K}^D$ α-significant in the scenario E w.r.t. $f\mathcal{K}^S$, denoted as $\mathcal{D}_\alpha(E, f\mathcal{K}^S)$, is a set of pairs $(I, \alpha_{i,j})$ such that:*

- $I \in \mathsf{Con}(\mathcal{K}^D)$
- $f\mathcal{K}^S \models \{\langle E \sqsubseteq_{>0} C_n \rangle, \langle P_{n,m} \sqsubseteq_{>0} \exists R_c.C_n \sqcap \exists R_d.D_m \rangle, \langle I \sqsubseteq_{>0} D_m \rangle\}$

- $\alpha_{i,j} = \mathsf{glb}(E \sqsubseteq C_n) \otimes \mathsf{glb}(P_{n,m} \sqsubseteq \exists R_1.C_n \sqcap \exists R_2.D_m) \otimes \mathsf{glb}(I \sqsubseteq D_m)$

The algorithm to calculate the α-significant domain of a scenario is a fuzzy extension of Algorithm 1.

Algorithm 2. *$\mathcal{D}_\alpha(E, f\mathcal{K}^S)$ can be computed in practice as follows:*

1. *Retrieve the complex contexts subsuming the query context (and their degree):*
   $Z_1 = \{(C_n, \beta_n) \mid (E \sqsubseteq_{>0} C_n) \wedge (\beta_n = \mathsf{glb}(E \sqsubseteq C_n))\}$
2. *Retrieve the σ-connections which involve the retrieved complex contexts (and their degree):*
   $Z_2 = \{(C_k, P_{k,l}, \beta_k) \mid$
   $(P_{k,l} \sqsubseteq_{>0} \exists R_c.C_k) \wedge (\beta_k = \mathsf{glb}(P_{k,l} \sqsubseteq \exists R_c.C_k)) \wedge (C_k \equiv C_n)\}$
3. *Retrieve the complex domains involved by the retrieved σ-connections (and their degree):*
   $Z_3 = \{(P_{k,l}, D_l, \beta_l) \mid (P_{k,l} \sqsubseteq_{>0} \exists R_d.D_l) \wedge (\beta_l = \mathsf{glb}(P_{k,l} \sqsubseteq \exists R_d.D_l))\}$
4. *Combine the partial degrees of the retrieved profiles using a $\otimes$:*
   $Z_4 = \{(C_k, D_l, \beta_{k,l}) \mid$
   $((C_k, P_{k,l}, \beta_k) \in Z_2) \wedge ((P_{k,l}, D_l, \beta_l) \in Z_3) \wedge (\beta_{k,l} = \beta_k \otimes \beta_l)\}$
5. *Aggregate, by means of a $\oplus$, all the degrees with which a domain has been retrieved :*
   $Z_5 = \{(D_m, \beta_m) \mid (\beta_m = \bigoplus_{(C_k, D_m, \beta_{k,l}) \in Z_4} (\beta_{k,l} \otimes \beta_n))\}$
6. *Retrieve the $I \in \mathsf{Con}(f\mathcal{K}^D)$ more specific than the retrieved complex domains (and their degree):*
   $\mathcal{D}_\alpha(E, f\mathcal{K}^S) = \{(I, \alpha_{i,j}) \mid (I \sqsubseteq D_m) \wedge (\alpha_{i,j} = \beta_m \otimes \mathsf{glb}(I \sqsubseteq D_m))\}$

(for simplicity's sake, we assume that $C_n, C_k \sqsubseteq C_\top, P_{k,l} \sqsubseteq P_\top, D_m, D_l \sqsubseteq D_\top$)

The output of Step 6 is a set of pairs containing all the $I \sqsubseteq D_m$ and their degree of importance. Since a concept I can be retrieved with more

than a degree through different profiles, these values should be conveniently aggregated again using a t-conorm $\oplus$, in order to provide the user with a single final relevance value. Therefore, the final output of the procedure to the user will be a set of pairs (simple domain concept, degree) with no repeated simple domain concepts.

Next, we present an example of the algorithm.

Example 4.7 Domain α-significant in a context in Nomadic Healthcare. Let us suppose the query context $\texttt{Anaphylaxis} \sqcap \exists\, \texttt{hasComplication}.\texttt{Elderly}$. Using Algorithm 2, we can retrieve the domains asserted to be interesting in this context, that is, $\mathcal{D}_\alpha(\texttt{Anaphylaxis} \sqcap \exists\, \texttt{hasComplication}.\texttt{Elderly}, f\mathcal{K}^S)$. In this example, the Gödel implication is used to interpret the semantics of GCIs.

- ***Step 1***

  $\texttt{Anaphylaxis} \sqcap \exists \texttt{hasComplication}.\texttt{Elderly} \sqsubseteq_{\geq 1} \texttt{Anaphylaxis} \sqsubseteq_{\geq 0.7} \texttt{Shock}$,

  $\texttt{Anaphylaxis} \sqcap \exists \texttt{hasComplication}.\texttt{Elderly} \sqsubseteq_{\geq 1} \geq 1\,\texttt{hasComplication}$

  $\Rightarrow \texttt{Anaphylaxis} \sqcap \exists \texttt{hasComplication}.\texttt{Elderly} \sqsubseteq_{\geq 0.7} \texttt{Shock} \sqcap \geq 1\,\texttt{hasComplication}$

  $\texttt{Anaphylaxis} \sqcap \exists \texttt{hasComplication}.\texttt{Elderly} \sqsubseteq_{\geq 0.7} \texttt{Shock} \sqcap \geq 1\,\texttt{hasComplication}$,

  $\texttt{Shock} \sqcap \geq 1\,\texttt{hasComplication} \sqsubseteq_{\geq 0.8} \texttt{EpinephrineAdmin}$

  $\Rightarrow \texttt{Anaphylaxis} \sqcap \exists \texttt{hasComplication}.\texttt{Elderly} \sqsubseteq_{\geq 0.7} \texttt{EpinephrineAdmin}$

  $Z_1 = \{(\texttt{C}_1, 1), (\texttt{C}_2, 1), (\texttt{C}_3, 0.7)\}$

- ***Step 2***

  $\texttt{P}_{1,1} \sqsubseteq_{\geq 0.6} \exists \texttt{R}_c.\texttt{C}_1 \sqcap \exists \texttt{R}_d.\texttt{D}_1 \Rightarrow \texttt{P}_{1,1} \sqsubseteq_{\geq 0.6} \exists \texttt{R}_c.\texttt{C}_1$

  $\texttt{P}_{2,2} \sqsubseteq_{\geq 0.5} \exists \texttt{R}_c.\texttt{C}_2 \sqcap \exists \texttt{R}_d.\texttt{D}_2 \Rightarrow \texttt{P}_{2,2} \sqsubseteq_{\geq 0.5} \exists \texttt{R}_c.\texttt{C}_2$

  $\texttt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists \texttt{R}_c.\texttt{C}_3 \sqcap \exists \texttt{R}_d.\texttt{D}_3 \Rightarrow \texttt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists \texttt{R}_c.\texttt{C}_3$

$Z_2 = \{(\mathtt{C}_1,\mathtt{P}_{1,1},0.6),(\mathtt{C}_2,\mathtt{P}_{2,2},0.5),(\mathtt{C}_3,\mathtt{P}_{3,3},0.9)\}$

- ***Step 3***

  $\mathtt{P}_{1,1} \sqsubseteq_{\geq 0.6} \exists \mathtt{R}_c.\mathtt{C}_1 \sqcap \exists \mathtt{R}_d.\mathtt{D}_1 \Rightarrow \mathtt{P}_{1,1} \sqsubseteq_{\geq 0.6} \exists \mathtt{R}_d.\mathtt{D}_1$
  $\mathtt{P}_{1,1} \sqsubseteq_{\geq 0.6} \exists \mathtt{R}_d.\mathtt{D}_1 \sqsubseteq_{\geq 1} \exists \mathtt{R}_d.\mathtt{D}_2 \Rightarrow \mathtt{P}_{1,1} \sqsubseteq_{\geq 0.6} \exists \mathtt{R}_d.\mathtt{D}_2$
  $\mathtt{P}_{2,2} \sqsubseteq_{\geq 0.5} \exists \mathtt{R}_c.\mathtt{C}_2 \sqcap \exists \mathtt{R}_d.\mathtt{D}_2 \Rightarrow \mathtt{P}_{2,2} \sqsubseteq_{\geq 0.5} \exists \mathtt{R}_d.\mathtt{D}_2$
  $\mathtt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists \mathtt{R}_c.\mathtt{C}_3 \sqcap \exists \mathtt{R}_d.\mathtt{D}_3 \Rightarrow \mathtt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists \mathtt{R}_d.\mathtt{D}_3$
  $\mathtt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists \mathtt{R}_d.\mathtt{D}_3 \sqsubseteq_{\geq 1} \exists \mathtt{R}_d.\mathtt{D}_1 \Rightarrow \mathtt{P}_{1,1} \sqsubseteq_{\geq 0.9} \exists \mathtt{R}_d.\mathtt{D}_1$
  $\mathtt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists \mathtt{R}_d.\mathtt{D}_1 \sqsubseteq_{\geq 1} \exists \mathtt{R}_d.\mathtt{D}_2 \Rightarrow \mathtt{P}_{3,3} \sqsubseteq_{\geq 0.9} \exists \mathtt{R}_d.\mathtt{D}_2$

  $Z_3 = \{(\mathtt{P}_{1,1},\mathtt{D}_1,0.6),(\mathtt{P}_{1,1},\mathtt{D}_2,0.6),(\mathtt{P}_{2,2},\mathtt{D}_2,0.5),$
  $(\mathtt{P}_{3,3},\mathtt{D}_3,0.9),(\mathtt{P}_{3,3},\mathtt{D}_1,0.9),(\mathtt{P}_{3,3},\mathtt{D}_2,0.9)\}$

- ***Step 4***

  $Z_4 = \{(\mathtt{C}_1,\mathtt{D}_1,min(0.6,0.6)=0.6),(\mathtt{C}_1,\mathtt{D}_2,min(0.6,0.6)=0.6),$
  $(\mathtt{C}_2,\mathtt{D}_2,min(0.5,0.5)=0.5),(\mathtt{C}_3,\mathtt{D}_1,min(0.9,0.9)=0.9),$
  $(\mathtt{C}_3,\mathtt{D}_2,min(0.9,\ 0.9)=0.9),(\mathtt{C}_3,\mathtt{D}_3,min(0.9,0.9)=0.9)\}$

- ***Step 5***

  $Z_5 = \{(\mathtt{D}_1,max(min(0.6,1),min(0.9,0.7))=0.7),$
  $(\mathtt{D}_2,max(min(0.6,1),min(0.5,1),min(0.9,0.7))=0.7),$
  $(\mathtt{D}_3,min(0.9,0.7)=0.7)\}$

- ***Step 6***

  $\mathcal{D}_\alpha(E,f\mathcal{K}^S) = \{\ \ (\mathtt{EHRCurrentPrescription},min(0.7,1)=0.7),$
  $(\mathtt{EHRCurrentPrescription},min(0.7,1)=0.7),$
  $(\mathtt{EHRAntidepressives},min(0.7,1)=0.7),$
  $(\mathtt{EHRDrugIntolerance},min(0.7,1)=0.7),$
  $(\mathtt{EHRAntidepressives},min(0.7,1)=0.7),$
  $(\mathtt{EHRAntidepressives},min(0.7,1)=0.7)\}$

If the outputs of the algorithm are aggregated, the final results provided to the user are:

(`EHRCurrentPrescription`, $max(0.7, 0.7) = 0.7$),
(`EHRDrugIntollerance`, 0.7),
(`EHRAntidepressives`, $max(0.7, 0.7, 0.7) = 0.7$).

These results mean that the system alerts the doctor, and tells him to check the patient information about current prescriptions, especially those concerning antidepressive drugs, and past diagnoses of drug intolerance. All the recommendations are equally important with degree 0.7.

Once the patient is identified, the concrete instances of these classes of the ontology can be obtained from the ABox $\mathcal{A}^D$. In this case, for the patient `jgomez`, the doctor is advised that the patient has a prescription of `prozac` and an intolerance to `procaine`.

Properties

Proposition 4. *Algorithm 2 is complete, i.e. it finds all the concepts I related with E through σ-connections and the degree of this connection.*

Proof. The expressions of Algorithm 2 show that the retrieved $P_{k,l}, C_n, D_m$ are the same as in the crisp case. The only difference in respect to the previous algorithm is the computation of β values. We assume in this proof that the Gödel implication is used to define the semantics of GCIs and the t-norm $\otimes$ is min, though it can be easily extended to other families of fuzzy operators.

Therefore, based on proof of Algorithm 1 and Definition 7, we have merely to prove that $\beta_{k,l} = \beta_k \otimes \beta_l$ is equal to $\mathrm{glb}(P_{k,l} \sqsubseteq \exists R_c.C_k \sqcap \exists R_d.D_l)$, the degree of the relevance relation between C_k (a superclass of E) and D_l (a superclass of I).

Using the properties of fuzzy sets, we know that $(A \Rightarrow B \otimes C) \geq \alpha$ implies $(A \Rightarrow B) \geq \alpha$ and $(A \Rightarrow C) \geq \alpha$, for some t-norm and its residuum-based im-

plication (for example, for *min* t-norm and the Gödel implication). Applying this expression to our GCI, $P_{k,l} \sqsubseteq_{\geq \beta_{k,l}} \exists R_c.C_k \sqcap \exists R_d.D_l \Rightarrow P_{k,l} \sqsubseteq_{\geq \gamma_1} \exists R_c.C_k (\gamma_1 \geq \beta_{k,l})$ and $P_{k,l} \sqsubseteq_{\geq \gamma_2} \exists R_d.D_l (\gamma_2 \geq \beta_{k,l})$. From Algorithm 2, we have the g|b*s* β_k and β_l. Since they are the greatest lower bounds, $\beta_k \geq \gamma_1 \geq \beta_{k,l}$ and $\beta_l \geq \gamma_2 \geq \beta_{k,l}$. Consequently, $\beta_k \geq \beta_{k,l} \Rightarrow \beta_{k,l} \leq \beta_k \otimes \beta'$, for any β', and $\beta_l \geq \beta_{k,l} \Rightarrow \alpha_{k,l} \leq \beta_l \otimes \beta''$, for any β''.

On the other hand, for *min* t-norm and the Gödel implication, $(A \Rightarrow B) \geq \alpha_1$ and $(A \Rightarrow C) \geq \alpha_2$ imply $(A \Rightarrow B \otimes C) \geq \alpha_1 \otimes \alpha_2$. Applying this expression to the GCIs of Algorithm 2, $P_{k,l} \sqsubseteq_{\geq \beta_k} \exists R_1.C_k$ and $P_{k,l} \sqsubseteq_{\geq \beta_l} \exists R_2.D_l$, we have $P_{k,l} \sqsubseteq_{\geq \beta_k \otimes \beta_l} \exists R_1.C_k \sqcap \exists R_2.D_l$. By definition, $\beta_{k,l} \geq \beta_k \otimes \beta_l$.

Consequently, $\beta_{k,l} \leq \beta_k \otimes \beta_l$ and $\beta_{k,l} \geq \beta_k \otimes \beta_l$, so necessarily $\beta_{k,l} = \beta_k \otimes \beta_l$.

□

Computational complexity. An upper bound for the computational complexity of the reasoning procedure can be deduced from the research studies [39, 41, 40], where fuzzy ontologies in $f\mathcal{SHOIN}$ and $f\mathcal{SROIQ}$ have been proved to be reducible to crisp ontologies. Therefore, the overall complexity of retrieving the α-significant domain concepts w.r.t. a fuzzy significance ontology is asymptotically bounded by the complexity of the reduction plus the complexity of the reasoning within the crisp ontology.

The mentioned contributions show that the complexity of this reduction for a $f\mathcal{SROIQ}$ ontology with Zadeh operators and the Gödel implication for GCIs (the top complexity level considered in this work) is quadratic (in space) with regard to the number of degrees used in the ontology[6].

Algorithm 2 calculates a considerable number of g|bs (at least, one for each retrieved concept in Steps 1-4). The computation of each g|b needs at most $log(N)$ subsumption tests, where N is the number of degrees of the

[6]This complexity can be cut down to linear if a fixed number of degrees is assumed. Moreover, under certain conditions (i.e. new axioms do not introduce new atomic concepts, new atomic roles, or new degrees of truth), this reduction can be performed only once, and this overhead can be avoided.

fuzzy ontology [246]. In the simplest case, the CDS ontology is in $f\mathcal{ALC}$, which implies that the complex context and the domain expressions are (at most) in $\mathcal{ALC}$; the number of degrees is fixed; and the fuzzy CDS ontology does not need to be reduced. If we assume this situation, the complexity of Steps 1-4 is upper-bounded by $|\mathsf{Con}(f\mathcal{K}^S)| \times log(N)$ times the subsumption test complexity (ExpTime for $\mathcal{ALC}$).

According to this result, despite the optimizations applied, reasoning with the fuzzy relevance ontology has a high computational complexity. Nevertheless, it is generally assumed that the worst cases are very infrequent in DL reasoning. Thus, as practical experiences show, the overhead produced by the fuzzy extension of the CDS pattern is assumable, and the use of fuzzy significance relations is recommended in applications that require higher descriptive power. This is the case of the Nomadic Healthcare use case, implemented in the IASO system described in the next chapter. However, in its current version the knowledge bases of IASO were built by applying the crisp CDS pattern. The use of the fuzzy approach remains to be explored in future work.

CHAPTER 5

A Knowledge Mobilization Application: the *IASO* System

This section describes the IASO system, a mobile application based on the principles of Knowledge Mobilization. IASO solves the Nomadic Healthcare KMob use case presented in Chapter 2. It also shows the usefulness of the architecture and the knowledge representation model created as part of our doctoral thesis.

5.1 Problem description

In the first chapter of this dissertation, we presented the Nomadic Healthcare KMob use case (Section 2.3). In this section, we give a detailed description of the design and the implementation of IASO, a prototype of a KMob system that solves the Nomadic Healthcare problem. IASO is a proof-of-concept system since it demonstrates the validity and usefulness of the contributions of Chapter 3 (the KMob architecture) and Chapter 4 (the CDS model).

The example of the Nomadic Healthcare use case is the following. Dr. Greg is caring for a patient outside the hospital. He is equipped with a

portable device, and wishes to consult the Hospital Information System (HIS), where the patient's clinical history is stored, with a view to prescribing a treatment for him. For instance, Dr. Greg should know whether the patient has been previously diagnosed as having an allergy to anesthetic drugs. By no means does he wish to receive the patient's complete history because he would be unable to review all the records available in the system. In fact, most of this data would be irrelevant, given the current situation of the patient and the type of treatment needed.

The IASO system that we designed is capable of accomplishing this task and providing the doctor with precisely the information required. Thus, IASO delivers customized summaries of the most relevant clinical data in the patient's medical history to the doctor so that he can decide on the best treatment. The contents of these summaries depend on the current state and situation of the patient, which is introduced by the doctor as the input of the application. The new application provides answers to questions such as "what should I know about this patient?" and "should I perform further clinical tests?".

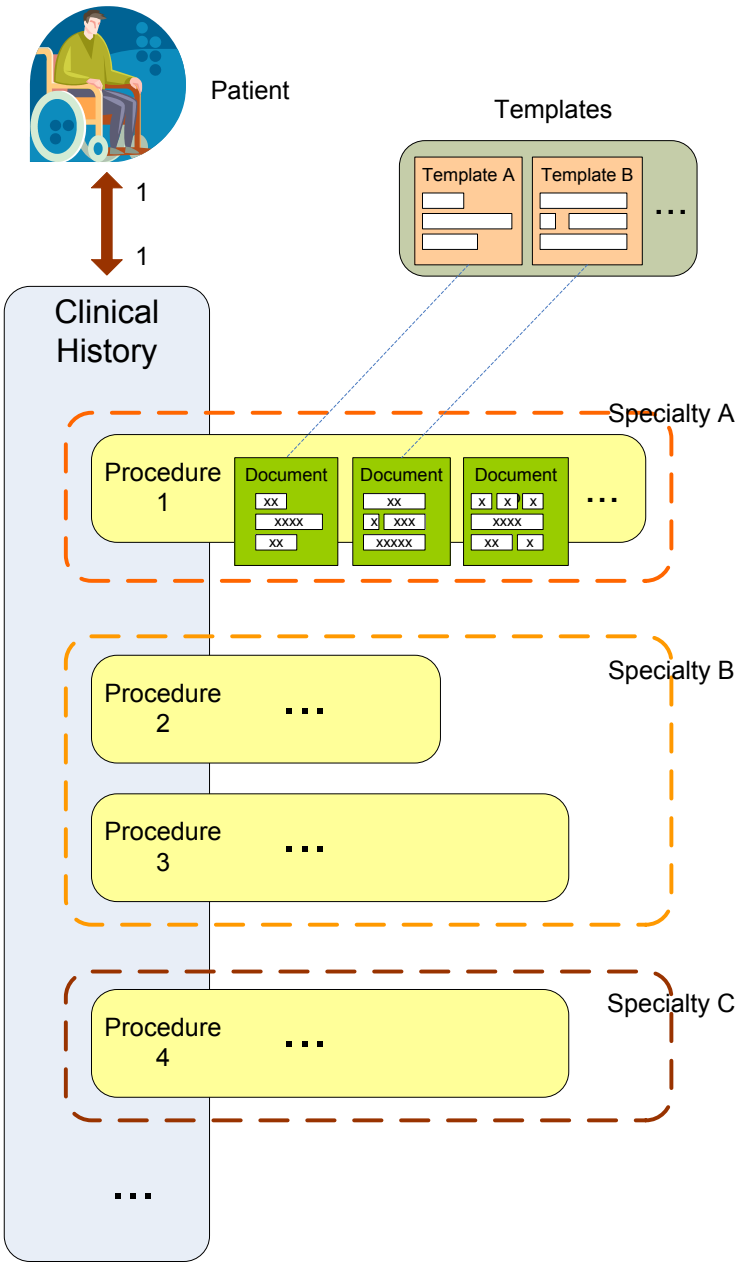
In this case, we use the data stored in the HIS of the Hospital Clínico San Cecilio of Granada. This system stores the clinical histories of the patients of the hospital. A clinical history is the sum total of all the electronic health records generated as a result of the healthcare services received by a patient. In other words, a patient's history includes all the information concerning his previous and current clinical treatments, diagnoses, and prescriptions.

The system relies on a database which links one clinical history to each patient. Information is stored in 'registers', and the registers are organized in 'electronic documents'. Documents are classified depending on the medical specialization that they are related to. The structure and contents of documents are defined with in terms of templates, which specify which data fields must be filled to generate a document. Figure 5.1 shows the logical structure of the database of the hospital.

It should be pointed out that the system implemented by the Hospital is

only used in this institution. Nevertheless, our approach could be extended to other systems, and, especially, Diraya, the clinical history system used in Primary Healthcare Centers that are part of the Andalusian Healthcare Service SAS [35].

Figure 5.1: Structure of the database of the Hospital Clínico San Cecilio



The San Cecilio HIS is a client-server application specifically developed for this hospital. Physicians can use the software to retrieve patients' histories by using the PCs in their offices. The HIS implements a security policy that prevents doctors from accessing information generated by procedures carried out in medical specializations other than their own. In order to guarantee data integrity and security, the system has a password-based identification mechanism.

The current system behaves reasonably well in situations in which the doctor has sufficient time to manually consult, filter, and extract useful information from the stored electronic documents. Nevertheless, problems arise in two situations: (i) when the system must be accessed from outside the hospital; (ii) when the doctor does not have enough time to review the information provided, or when the information cannot be processed.

The first problem can be solved by implementing a gateway to access the HIS from outside the hospital. This is relatively easy to do if security issues are not considered. For instance, a Web interface that replicates query forms would allow remote access with different tools, including mobile devices. However, the second issue is more difficult to resolve since more knowledge must be added to the system to avoid information overload (see Section 4.1). A knowledge base stating which clinical data is relevant in a given patient situation must also be created. Data summaries would be composed when the user consults this knowledge base.

IASO elegantly solves both of these problems. The IASO application has the characteristic features of KMob systems. It is a ubiquitous system, since application services can be accessed from anywhere, at anytime, and with any device. The system must behave in consonance with the context of use, and discover which information is needed by the doctors in each case. Doctors do not need to tell the system how this information can be retrieved since the system is capable of automatically inferring it. As already mentioned, in this application, it is particularly important to be concise and to summarize the available data in order to avoid information overload. Last but not least, the system has to integrate heterogeneous information sources

as well as different communication technologies.

Consequently, we applied the KMob principles, methodologies, and tools analyzed in this thesis. The system was designed by specializing of the architecture presented in Chapter 3. The implementation was carried out by using the technologies described in that chapter. The knowledge base that supports the creation of the context-dependent summaries was created by relying on the design pattern proposed in Chapter 4.

5.2 Related work

The management of the clinical data of patients has always been a principal focus of interest in Medical Informatics. However, two problems which arise in this area are data storage and data retrieval. Effectively solving both problems is crucial for providing better healthcare services.

The acquisition and storage of clinical data has been extensively studied since the appearance of the first information systems. Currently, there are various interesting proposals that target the creation of standards for electronic health records (EHRs) [84].

One of the most recent of these initiatives is OpenEHR[1], which claims to improve the previous specifications for EHR management systems. OpenEHR defines ADL (Archetype Description Language), a language for describing the medical concepts that can be represented in an HIS [10]. These concepts are called archetypes. For example, the archetype "blood pressure', which includes the values of the systolic and the diastolic blood pressure, the place where the measurements have been obtained, and the device which has been used, can be defined with OpenEHR. A list of possible ADL archetypes can be found at the OpenEHR website [2].

The retrieval of clinical data is the other problem that HIS research work is currently trying to solve. Retrieval does not only involve querying HIS

[1] http://www.openehr.org
[2] http://svn.openehr.org/knowledge/archetypes/dev/index.html

databases, which is performed with suitable consultation languages (e.g. EQL for OpenEHR). It is also necessary to *understand* the data obtained. A standard for EHRs does not completely solve this issue because the requester may not know what the values mean. If the requester is a human user, he will probably be able to decipher this data, but if the procedure is automatic, interpreting such data is almost impossible. This is especially problematic when systems using different representation models are obliged to understand each other. In order to support interoperability and exchange of data, the meaning of the terms used by each organization must be well-defined and published.

As a result, there is much research now in progress, whose objective is to endow HIS data with semantics. Adding formal metadata to describe HIS registers allows the patients' data to be automatically processed. These metadata must be incorporated at two levels: (i) the level describing EHR meaning; (ii) the level describing EHR contents.

The first of these objectives is achieved by using an EHR metamodel. For instance, OpenEHR primitives are formally characterized in the OpenEHR UML model. The OpenEHR metamodel has been translated to OWL in such a way that archetypes can be specified by using an OWL ontology [30, 143, 222].

The second objective is part of the more general problem stemming from the formal description of terms used in Medicine. As it is well-known, this is one of the most frequently studied application domains of ontologies. It is beyond of the scope of this work to provide a review of medical ontologies. Important medical ontologies such as UMLS [45], Galen [221], and SNOMED-CT [70], which were originally coded in other representation languages, have now all been translated to OWL.

There is a considerable number of contributions that offer solutions for HIS remote access. Some of them even use portable devices [82, 110]. With the rise of Web technologies, this problem has been successfully solved. The main challenge for the developers of these systems is to guarantee security

and integrity of the data, which is especially important given the high degree of privacy required for medical records.

To the best of our knowledge, there is little or no significant research on the automatic summarization of patients' clinical histories for on-line consultation. Nevertheless, this facility has been stressed by some authors as a very interesting feature for HISs to have [102]. We strongly believe that context-dependent summarization will not only be desirable, but essential in order to make the most of the large medical data warehouses that will be available in the very near future.

5.3 Design

The KMob system developed to meet the requirements explained in Section 5.1 is called IASO[3] (*Intelligent ASsistant for Outdoors healthcare*). IASO allows nomadic users to access the HIS of the Hospital Clíinico San Cecilio. Users, equipped with mobile devices, provide a description of the clinical situation of the patient and his identifier, and the system infers which information of the HIS ought to be made accessible. Thus, in order to avoid information overload, the resulting information is a summary of the available data for the patient.

To create this system, we had to successfully accomplish the following tasks:

1. The construction of a knowledge model capable of describing which subset of the data stored should be considered in a given scenario.

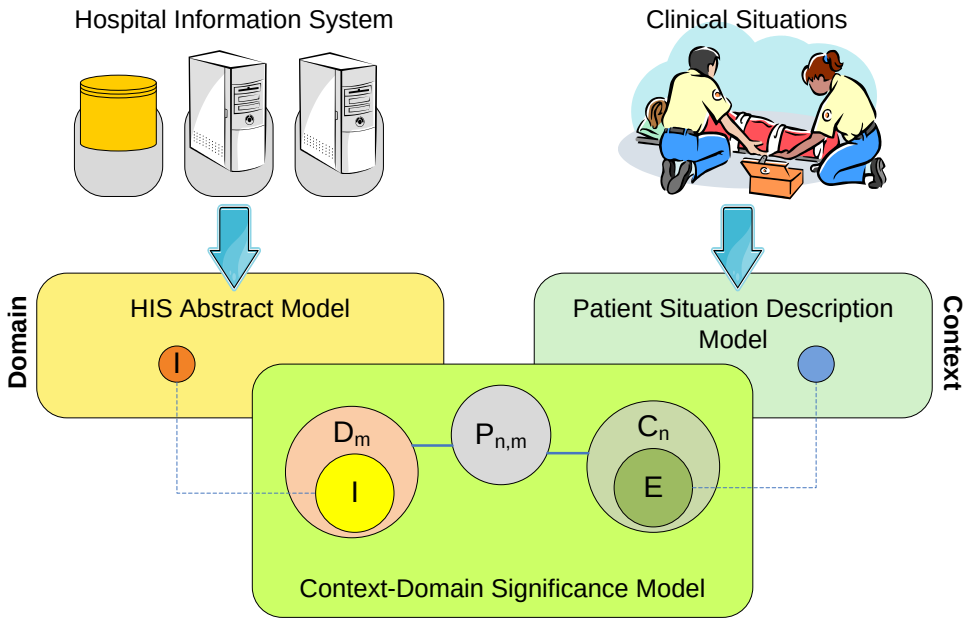
2. The implementation of the software that delivers this information to the (doctors') mobile devices.

[3]Iaso (also "Iaso Tholus" or "Jaso"; in Ionic Greek, "Ieso") was the Greek goddess of recovery from illness. Iaso was the daughter of Asclepius and had three sisters and a stepsister: Panacea, Aceso, Aglæa-Ægle and Hygeia. Iaso helped sick people together with Panacea, Aglæa and Hygeia.

IASO knowledge base

The IASO knowledge base was created by applying the crisp version of the CDS design pattern. We developed three OWL ontologies that abstractly represent: (i) the data registers of the clinical histories managed by the HIS (the *domain*); (ii) a vocabulary to describe the patients' clinical situations (the *context*); (iii) links stating that certain data registers are significant in a given situation (the σ-connections). Figure 5.2 illustrates the relations between these three ontologies. The contents and the structure of the ontologies are extensions of the (OWL translation of the) examples included in Chapter 4.

Figure 5.2: Schema of the IASO context, domain, and significance ontologies



As mentioned in Chapter 4, it is possible to reuse existing ontologies to build the context, domain, and significance models. We used the OWL translation of the Galen ontology [218], available in the Web[4], to create the Pa-

[4] http://www.co-ode.org/galen/full-galen.owl

tient Situation Description Model. A local copy of Galen was imported from this ontology.

Likewise, a standard EHR representation model could have been used to develop the domain ontology. Nevertheless, since the HIS database has a very idiosyncratic structure, we decided to build the domain ontology from scratch. The HIS Abstract Model includes concepts that model *patients*, *clinical histories*, *medical specializations*, *electronic documents*, etc. The instances of these concepts are the concrete values stored in the HIS. This signifies that it would then be necessary to import instance data from the HIS to the HIS Abstract Model. We solved this problem by implementing a *bridge* between this ontology and the HIS in such a way that HIS data is retrieved on demand as ontology instances. This solution is explained in Section 5.4.

In this version of the IASO CDS ontology, σ-connections have been created by relying on the advice of experts in the field and specialized bibliography. In the future, we plan to study if it is possible to semi-automatically build the CDS ontology by relying on formal medical guidelines.

The resolution of a query with the IASO CDS ontology consists of the three stages explained in Example 4.4. Firstly, a description of a patient situation (created in terms of the Patient Situation Description Model) and the identifier of the patient are provided by the user. Secondly, using this description as the input, the algorithm capable of inferring the domain significant to a context (Section 4.4) is executed. The result of the algorithm is a set that includes the concepts of the HIS Abstract Model which are regarded as significant. Thirdly, the instances of these concepts that correspond to the patient are retrieved. These values are the final output of the query resolution process.

IASO architecture

The IASO architecture instantiates the abstract architecture proposed in Chapter 3. We chose a client-server communication schema to organize the appli-

cation. Accordingly, client-server technologies were used to implement IASO, as explained in the next section.

We believe that the client-server pattern is the most suitable for this system. The most important reason for this is that the query resolution process is too complex to be executed with a mobile device. The mobile agent should manage a DL reasoning engine and a full version of the CDS ontology, including instances, to completely resolve any query. This is not currently viable. Therefore, the mobile agent only knows the Patient Situation Description vocabulary, and is the desktop agent who carries out the query-resolving process. Since client and server roles are clearly identified, the client-server paradigm is a natural way of structuring the system.

Figure 5.3 shows the Society Diagram of the IASO application. It depicts the Nomadic Agents, Desktop Agents, Services, and Knowledge Models of the system.

The IASO Client Agent is a Nomadic Agent that runs on the users' mobile devices. It manages a copy of the Patient Description Model, which is used to create the descriptions of the clinical situations that are sent as part of the queries to the IASO Server Agent. This client agent offers a friendly interface to introduce these descriptions. Once a query has been introduced, the client agent acquires the IASO Client role. This role encompasses all the functions aimed at: (i) requesting the IASO Query Service; (ii) processing the results of the query to present them to the user in a suitable format.

The IASO Query Service is the service that processes user queries. The input of this service is a description of a concept built with the Patient Description Model vocabulary and a patient ID. The output of the service is a set of concept descriptions corresponding to the significant registers retrieved and the instances of these concepts[5].

The IASO Server Agent is a Desktop Agent that runs on an application

[5]This ontological data is expressed with the HIS Abstract Model. Therefore, it might also be useful to manage a copy of the HIS Abstract Model in the IASO Client in order to interpret the results. However, we consider that the results should be interpreted by the user, who is an expert in the area.

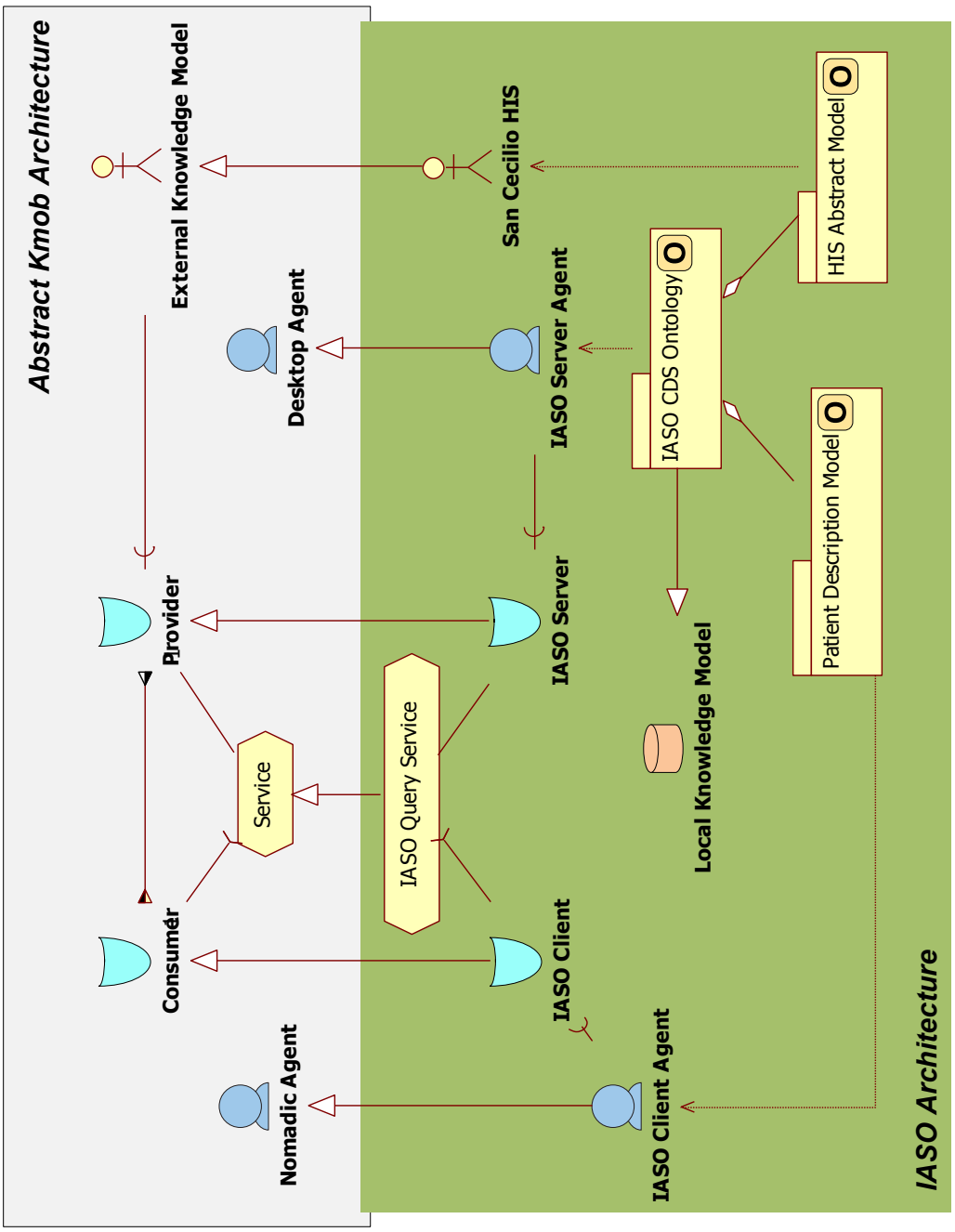


Figure 5.3: Architecture of the IASO system

server. When this agent acquires the IASO Server role, it provides the IASO Query Service. In order to solve IASO Clients queries, the IASO Server Agent manages the IASO CDS Ontology, which relates the descriptions created by means of the Patient Description Model ontology with sets of HIS registers modeled by using the HIS Abstract Model.

Worthy of mention is the participation of the San Cecilio HIS in the system. This is an External Knowledge Model, and consequently, it is modeled as an external agent. As commented in the previous sections, the instances of the HIS Abstract Model must reflect the contents of the HIS. Therefore, there is a dependence relation between the ontology and the HIS itself.

5.4 Implementation

We developed a proof-of-concept prototype of the IASO system described in the previous section [42]. This prototype is available at http://arai.ugr.es/8084/IasoTest/EntryPoint.

The prototype simplifies the design of IASO by assuming that the ontologies and databases that participate in the system are reduced versions of the complete ones that would be used in a practical implementation. Thus, the architecture of the IASO prototype is the one depicted in Figure 5.3, but the size of the context, domain, and CDS ontologies, as well as the HIS database, has been reduced.

More precisely, the simplifications in the prototype are the following. First, the CDS ontology contains only a few σ-connections. Secondly, the simplified version of the HIS database only reflects the clinical histories with one medical specialization, which is implicitly represented. This database only contains certain test values, which allowed us to sidestep security and privacy issues. Thirdly, the HIS Abstract ontology only models the information stored in this simplification of the HIS. Apart from these considerations, the prototype offers all the features that the complete IASO system would

provide. The prototype can be easily extended, and the analysis of its properties can be inductively generalized to the eventual complete system.

The IASO architecture was implemented by using the client-server technology framework proposed in Section 3.4. More specifically, we developed the IASO prototype as a Web-based application. The clients and the server communicate with HTML forms that are transmitted with the HTTP protocol. Figure 5.4 shows the technologies applied to implement the IASO prototype.

The IASO Client Agent is a Web browser that can access to the Web pages created by the IASO Server Agent. Thanks to this simplicity, this implementation of the client guarantees that most mobile devices can participate in the system, despite their computational limitations, as long as they are able to access the Web.

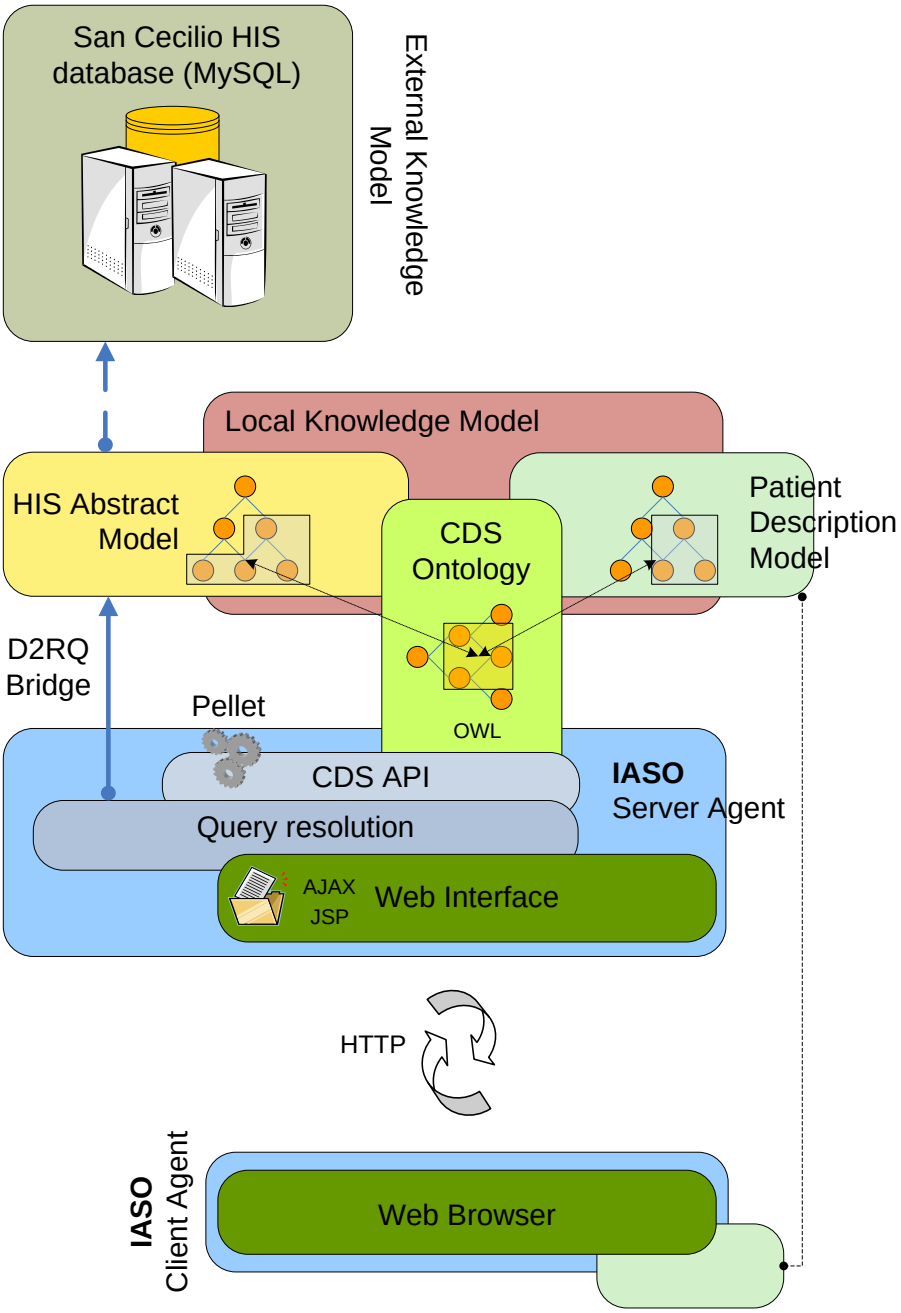
The IASO Server Agent provides an access point to the query-resolving service throughout an HTML form. This form allows the client to introduce a query and send it to the server. The query format in this implementation is an *ad hoc* ontology expression though it could be serialized to one of the possible OWL syntaxes. The IASO Server Agent receives and solves queries with the knowledge available in the IASO CDS Ontology and the HIS Database. In order to manage the CDS ontology, the server uses the CDS API, which implements Algorithm 1, and a DL inference engine (Pellet). The IASO Server Agent application runs on an Apache Tomcat Web server.

In order to minimize the drawbacks of the HTTP interaction, the IASO Server Agent was implemented with JSF and AJAX technologies. The JSF distribution used in the prototype is the open source library Apache MyFaces[6]. Since MyFaces supports AJAX, the IASO Client Agent can execute complex Javascript code. The use of JSF and AJAX in our prototype facilitates the creation of an attractive interface that clearly presents the input and the output information in the user's mobile device.

The Local Knowledge Model of the IASO Server Agent is composed of the three previously mentioned ontologies: HIS Abstract Model, Patient De-

[6] http://myfaces.apache.org/

Figure 5.4: Implementation of the IASO prototype



scription Model, and IASO CDS Ontology. These ontologies can be published in the same application server where the IASO Server Agent runs or in another Web server, since the CDS API transparently manages either local or remote ontologies. In this implementation, they were deployed in a second Apache web server[7].

An outstanding feature of the knowledge base in the prototype is the presence of the SQL Bridge. This bridge is responsible for transparently retrieving data from the HIS database as if they were instances of the ontology, but without actually importing them. In this way, the register contents of the HIS do not need to explicitly be part of the HIS Abstract Model. The SQL Bridge was implemented by using D2RQ, a toolkit to describe mappings between relational databases and OWL/RDF(S) ontologies and to manage the resulting models. D2RQ is described in Section B.4.

The SQL Bridge avoids two important problems. On the one hand, synchronization between ontology instances and database values is not required, which is a costly process. On the other hand, ontology inference procedures are more efficient, since the time and memory needed to reason with such a large ontology would be excessive.

5.5 Execution

The IASO prototype execution process is the following:

1. A user, equipped with a portable device that can access the Web throughout a wireless or cellular network, launches the Web browser (IASO Client Agent) and downloads the HTML query form from the server (IASO Server Agent).

2. On the form, the user describes the situation of the patient by using terms of the Patient Description Model. Before sending back the query form, the user also introduces the patient ID.

[7] http://httpd.apache.org/

3. On receiving the query, the server transforms the description of the clinical situation of the patient. The transformed description is used as the input of the implementation of Algorithm 1 provided by the CDS ontology. The result of this procedure is the set of concepts from the HIS Abstract Model that are relevant.

4. The server uses the SQL Bridge to retrieve the instances of the relevant concepts from the HIS database.

5. The resulting information is conveniently formatted and sent back as an HTML form to the user.

Figures 5.5 and 5.6 show two screens corresponding to the input and the output forms of the IASO prototype, respectively. These pictures were obtained by solving Example 4.4 with the IASO prototype at `http://arai.ugr.es/8084/IasoTest/EntryPoint`. The Windows Mobile 5 emulator and the Opera Mobile navigator were used to access the system.

Figure 5.5 presents the input form. On the left (A), the available terms are listed, whereas on the right (B), the introduced query is shown. The description of the clinical situation depicts a patient, named "Juan Gomez", who is unconscious, hemorrhagic, and has a penetrating wound (the terms are interpreted conjunctively).

Figure 5.6 shows the results obtained for this query. The 'clip' icons (A) represent the concepts of the HIS abstract model that are relevant. In this case, the significant registers are those related to blood pressure disorders, drug intolerance, procaine intolerance, and penicillin intolerance. The 'ok' bullets (B) represent instances of these concepts retrieved throughout the SQL Bridge, i.e. the values of the HIS database corresponding to the relevant registers associated with "Juan Gomez". For instance, the value of the register "ProcaineIntollerance" is "Procaine allergic".

The 'warning' bullets (C) provide further information about this query. More precisely, this is a list of clinical situations that are more specific than the one described in the query. These more specific clinical situations might

Figure 5.5: Input form of the IASO prototype

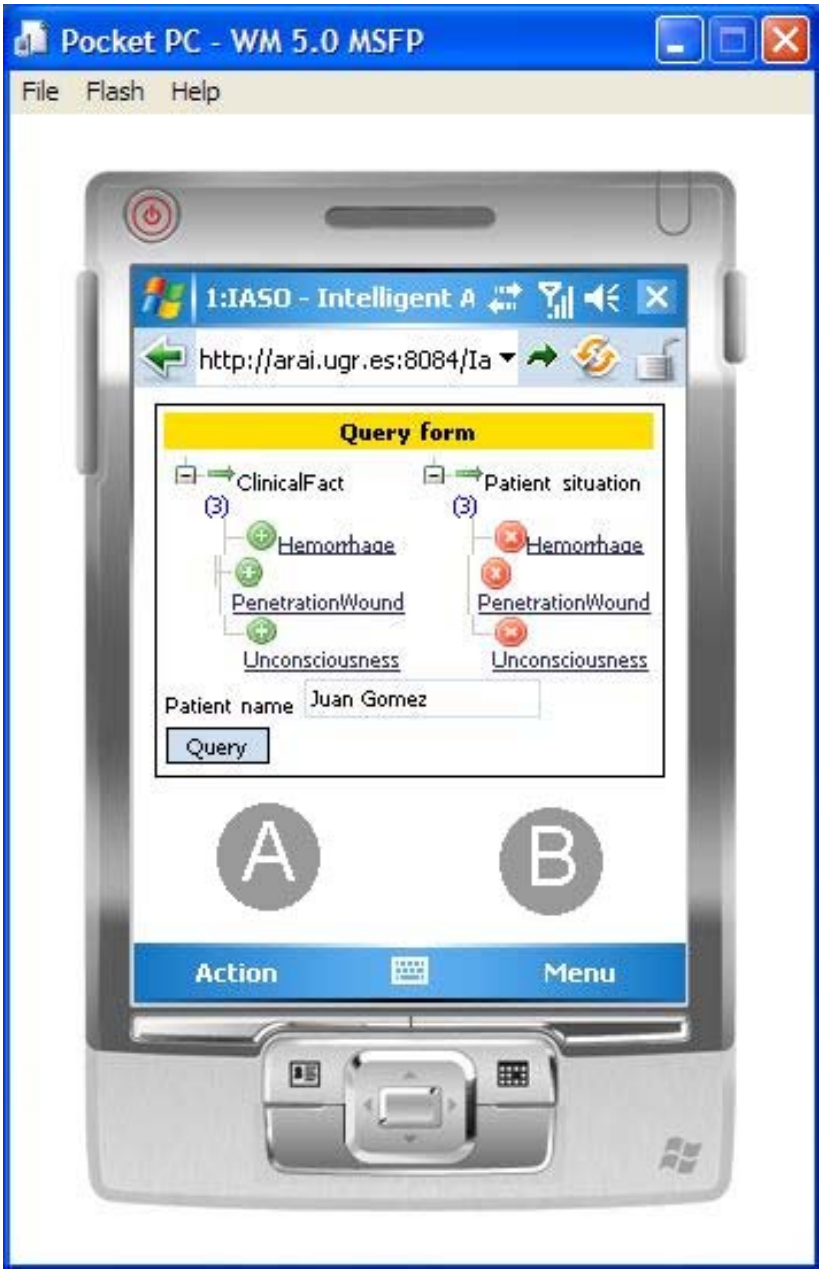


Figure 5.6: Output form of the IASO prototype

be considered in subsequent consultations. In the example, in the query the doctor may consider adding the qualifier ‘High’ to the hemorrhage symptom, because patient data is available that is relevant to this situation.

As expected, the architecture and the context-dependent knowledge representation model proposed in this thesis provide excellent support for the development of an application that solves the Nomadic Healthcare KMob use case. The IASO application and the prototype implemented show that both of them are valuable contributions to KMob.

CHAPTER 6

Conclusions and Future Work

As part of our research, we studied Knowledge Mobilization, a research area aimed at providing integral solutions for the challenges that arise when developing mobile systems for the delivery of knowledge retrieved from large information sources to nomadic users. The general objective of this research is the investigation of theories, techniques, and tools that facilitate this process. More specifically, we found potential solutions for two computational Knowledge Mobilization problems: (i) the access and distribution of heterogeneous knowledge in mobile systems; (ii) the overload of information that users experience if all the available knowledge is delivered to their mobile devices.

We have seen that contributions from areas such as Distributed Artificial Intelligence, Ubiquitous Computing, Semantic Web, Soft Computing, or Mobile Computing, can help to achieve successful Knowledge Mobilization. Chapter 2 fulfills the first of our objectives, which was to further develop the concept of Knowledge Mobilization (or *KMob*). In this chapter, we examined previous definitions of Knowledge Mobilization, and specified the features of KMob systems. Despite the many approaches for the development of intelligent mobile systems, none really provides a satisfactory solution for the problems mentioned above. Moreover, the plethora of existing technologies

and the heterogeneity of mobile platforms and devices make the development of such systems even more difficult.

On the basis of our findings, we proposed a new architecture that describes the structure, relations, and entities of a KMob system. We provided a general definition of a KMob architecture, which can support different configurations, communication schemas, etc. It goes without saying that the requirements of mobile systems can vary considerably, and strongly depend on the applicacion domain. The architecture described in Chapter 3 fulfills the second objective of this thesis, which is directly related to the problem of distribution of knowledge in mobile systems. The use of the AML language in the specification of the architecture makes it both comprehensible and reusable.

Along with the architecture, we studied various current technologies that can be applied to implement KMob systems. We classified these technologies in three frameworks, which correspond to three different system archetypes. The frameworks target the coordination of system entities (multi-agent approach), interchange of knowledge (tuplespace approach), and delegation of tasks (client-server approach), respectively.

We also dealt with the problem of information overload in knowledge-intensive systems, and especially in KMob systems. We determined that information overload can be overcome by using contextual knowledge to customize system answers to user environment. Consequently, we proposed a design pattern (the CDS model) to create significance ontologies, which are knowledge bases that state which information is relevant in a given context. Significance ontologies can be used to create summaries containing only context-relevant knowledge. This is accomplished by reasoning with an associated inference algorithm. Thus, the third objective of the thesis is attained with the CDS pattern presented in Chapter 4.

Furthermore, we implemented two tools to create and manage significance ontologies, namely the CDS Java API and the CDS plug-in for Protégé. These tools offer additional support to KMob system developers. As an im-

provement of the CDS pattern, we proposed an extension that allows fuzzy significance ontologies to be created. This extension permits the representation of fuzzy contexts and domains, which is very useful in several application domains. The fuzzy CDS pattern allows imprecise context and domains to be represented, and ranking of significance relations.

The KMob architecture and the significance ontologies are an integral approach to KMob systems development. This assertion was demonstrated in Chapter 5, where the design and the implementation of a KMob system is described. The IASO application is a system that solves the Nomadic Healthcare KMob use case. IASO is based on the abstract architecture that was specified, and relies on a knowledge base that follows the CDS design pattern. Additionally, a prototype of IASO was implemented by using the client-server technology framework. Thus, IASO is evidence that these two results are valid, which is the last (but certainly not the least) of our objectives.

Since the sub-objectives of this thesis were fulfilled we can affirm that the general objective was also achieved. We studied two important problems that arise in knowledge-intensive mobile systems, namely knowledge distribution and information overload, and we proposed valid solutions for them.

The solutions presented in this thesis have certain limitations though in our future research we already envision ways to improve and significant enhance them. Likewise, new research lines have come to light as the result of this work. Actually, another important contribution of this research is that it opens the door to an extremely promising field of study.

From a general point of view, our proposals do not target a specific domain though they have been tested in the Nomadic Healthcare use case. Without any loss of generality, the abstract architecture, the examples of the CDS model, and the knowledge base were specifically implemented in the IASO system. However, these results could just as usefully been applied to other application areas.

The KMob architecture could be improved by specifying in greater detail the *orchestration* and *choreography* of agents and services, which at the

moment is mainly left to the application designer. Orchestration defines the external services that are required by services to fulfill a task, whereas choreography establishes how messages are created and exchanged to request a service. These two features could be specified by using AML Protocol Sequence and Protocol Communication Diagrams.

In addition, the description of service features with a semantic language should be considered. Semantic Web Services are a recent technology that proposes adding formal metadata to Web Services in such a way that services can be automatically located, invoked, and integrated (see Section B.5). Semantic Web Services can be incorporated into our architecture with relative ease, and this would be another interesting direction for future work. Nowadays Semantic Web Services is a very active topic of Semantic Web research since the W3C has not as yet published a specification.

Further studies concerning the knowledge representation model are also needed, and would be a very enriching research goal. For instance, as already mentioned, the context-dependent reasoning performed with significance ontologies is similar to the inference procedures of certain variants of classic Logic. Thus, the relation of context representation models and other formalisms, such as temporal and non-monotonic logics, should be explored with a focus on whether they can be mutually reduced. Independently of the CDS pattern approach, the creation, publication and promotion of best practices for Semantic Web ontologies is essential for the advancement of Semantic applications. In this regard, it is worth mentioning that ontology patterns could be described with a formal ontology language and published in a publicly available repository.

Further research on the fuzzy extension of the CDS design pattern should also be carried out. We strongly believe that the ability to reason with fuzzy context descriptions is an important contribution of this model, and it could be exploited in different scenarios. Other minor aspects to be developed are: (i) the extension of the CDS Protégé plug-in to facilitate the creation of fuzzy significance ontologies, which are currently not supported; (ii) the refinement of the reasoning process with fuzzy significance ontologies, which

is computationally complex.

In the near future, we expect to continue working on the IASO system and to continue improving the current prototype. It will thus be necessary to extend the domain, context, and significance ontologies in order to better represent the data of the San Cecilio HIS, the eventual patient situations, and the medical protocols that determine which procedures should be used in each situation. A major challenge will be to guarantee the security and integrity of communications between components of the system, given that medical information should be private.

In conclusion, there has been very little research on intelligent mobile systems. Mobile technologies have obviously changed the way that information is delivered in the same way that network technologies did in the previous decades. Nevertheless, mobility is much more than a set of new technologies for the transmission of information. Mobility entails a paradigm shift that is closely related to today's necessities of nowadays (it both increases them and resolves them). Immediacy, massiveness, location-independence, or context-awareness are problems that Computer Science can solve, or at least mitigate, with *intelligent* software. Thus, in order for system users and developers not to be overwhelmed in the mobile age, suitable knowledge models, methodologies, architectures, supporting tools, etc. for intelligent mobile systems need to be developed. This is precisely what we promised to do in our doctoral thesis, and it is what we have accomplished as shown by our proposals and the results obtained in our research.

APPENDIX A

Ontologies and Description Logics

This appendix focuses on ontologies, a widely-used knowledge representation formalism, and its close relation with Description Logics, a family of logics for representing structured knowledge. First, ontology fundamentals are presented, followed by a brief introduction to the representation of knowledge and reasoning with Description Logics. We then describe $\mathcal{ALC}$, the Description Logic used in our research. The appendix concludes with a short note on fuzzy Descriptions Logics.

A.1 Background

Artificial Intelligence investigates Intelligent Systems (IS), defined as computational systems aimed at solving complex problems by using algorithms inspired in intelligent human problem-solving methods when classical techniques fail [223]. Knowledge Engineering is a subtopic of Artificial Intelligence that is concerned with Knowledge Representation (KR), the study of how the agents of an IS manage *what they know* before deciding *what to do* [51]. In this context, intelligence is achieved *reasoning*, namely, the ability to automatically infer implicit conclusions from explicit knowledge.

Representing knowledge basically consists of writing descriptions of the entities of a domain using the symbols of a language. Since computational KR requires these descriptions to be interpretable by a computer, representation languages must be formal, i.e., they must have a well-defined specification.

First Order Logic (or Predicate Logic) was the language initially used in KR. Unfortunately, representation of knowledge using First Order Logic poses certain problems. For example, its verboseness makes it tedious and unintuitive. Considerably more important is the fact that according to Church and Turing theorems, reasoning about general First Order Logic formulas is a semi-decidable process: it cannot be known if the algorithm which finds if a predicate is true will finish for one that cannot be satisfied.

Verbosity has been overcome by creating other representation languages. Generally speaking, even though these languages are not more expressive than First Order Logic and may not even have a logical substratum, they allow knowledge to be acquired and managed more easily. This group includes cognitive models, and more specifically, network-based models [160].

The second limitation, undecidability, was handled by only considering decidable and complete subsets of First Order Logic. *Decidable* means that all inferences will finish in a finite time period, and *complete* means that all entailments are guaranteed to be computed. The primary motivation of Description Logics is to characterize different families of logics which, depending on their expressivity (i.e., the primitive constructors allowed), encompass certain computational properties.

Two additional difficulties in current IS are the following: (i) the necessity of having a huge amount of distributed and heterogeneous information sources that must be incorporated to the knowledge model of the system in order to have the most complete, up-to-date information; (ii) the convenience of reusing previous knowledge bases in order to minimize the effort of developing a new application.

Ontologies are a knowledge representation formalism aimed at dealing with all these problems, since, as shall be explained, they possess the intu-

itiveness of cognitive models; provide a formal semantics based on Description Logics; and promote information integration and knowledge reuse.

A.2 Definition

The term *Ontology* comes from Philosophy, which defines it as the study of part-of relationships and entity dependencies [56]. Ontology as a science analyzes the features of possible things, and the categories in which they can be included.

In Knowledge Engineering, the concept of Ontology has evolved over the past decades. There is now a consensus of opinion that *an ontology* is a rigorous and exhaustive conceptual schema, focused on a certain domain and designed to facilitate information communication and reuse among different computational systems. Ontologies are considered to be a proper formalism for the representation of knowledge in modern IS [64], and are one of the most frequently used models nowadays. Actually, they have been proposed for the support of metadata management in the Semantic Web [27], and not surprisingly, we use ontologies in this work for representing knowledge in KMob applications.

One of the most widely cited definitions of *ontology* comes from Studer, Benjamins, and Fensel: "an ontology is a formal, explicit specification of a shared conceptualization" (in [250], based on [105] and [48]). The authors state that an ontology is a knowledge model which describes from a common perspective the objects in a common domain using a language that can be processed automatically. This language is usually[1] a Description Logic-based representation. Since Description Logics have proved to be suitable ontology languages [14, 120], for all practical purposes, the term ontology can be regarded as a Description Logic knowledge base. Hence, we concentrate on Description Logics in the remaining sections of this Appendix.

[1] KIF (Knowledge Interchange Format) [100] or F-Logic [142, 141] are other formalisms to represent ontologies which are not based on Description Logics.

An ontology is developed from the following primitive elements:

- *Concepts*. Concepts or classes represent the basic ideas of the domain which must be understood, and they determine sets which classify domain objects. Concepts are arranged in a hierarchy. Accordingly, a concept of an upper level is more general than a concept of a lower level.
- *Instances*. Instances or individuals are concrete occurrences of a concept.
- *Relations*. Relations or roles represent binary connections between individuals or individuals and typed values (integers, strings, etc.).
- *Axioms*. Axioms establish restrictions over concepts, instances, and relations, describing their attributes by delimiting their possible interpretation.

Ontology features have several advantages over other formalisms. These advantages are the following: information sharing among different people or software agents, knowledge reuse, separation between declarative and procedural knowledge, and acquisition and analysis of knowledge.

Sharing knowledge representations is one of the main objectives of ontologies. For instance, let us examine the case of two web sites, one supplying information about symptoms and medical diseases, and another displaying a catalogue of pharmaceutical products. If a common ontology defining the semantics of the clinical terms is used, a software agent will be able to integrate information from both sites, thus providing value-added services. For example, drug prescriptions in the diagnosis site could be automatically linked to drug contraindications in the pharmaceutical site. In this way, sharing knowledge allows information to be more automatically located and integrated.

Knowledge reuse is another fundamental ontology benefit. Ontologies are especially designed to save time and effort in the knowledge acquisition process by promoting the combination of previous models that describe

specific parts of the domain, and refining more general models to represent particular details of the domain.

Moreover, ontologies clearly distinguish between declarative and procedural knowledge: ontologies are composed of explicit definitions of domain entities, which guarantees that if the available information changes, the assertions in the model can be modified without having to re-implement the software which uses it.

Once an ontology is developed, a formal specification of the domain will be available. This makes it possible to both manually and automatically validate and verify the knowledge represented, and to incorporate it into a reliable repository.

A.3 Description Logics

Basics

Baader, Horrocks, and Sattler define Description Logics (DLs) as "a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way" [15].

DLs have features of cognitive models, such as Minsky's frames [92], Sowa's conceptual graphs [237], or Quillian semantic networks [213], but they have the advantage of providing a formal substratum that these formalisms lack. It was Brachman and Levesque who showed that most cognitive models can be endowed with formal semantics, expressed with fragments of First Order Logic, and furthermore, that the fragment considered determines the complexity of reasoning procedures with the models [52, 161]. In this way, DLs were born.

Research on DLs is focused on: (i) studying the theoretical foundations of DLs (e.g. semantics, reasoning, and complexity of the various DLs); (ii) developing knowledge representation frameworks to support DL management

(e.g. representation languages and inference procedures); (iii) implementing practical applications that rely on them (e.g. Semantic Web). An extensive introduction to these three areas is provided in [17].

DLs are ontology languages that represent domain knowledge by asserting axioms built from concept, role, and instance expressions. Complex concept and role expressions are defined by using the logic-based constructors provided by the concrete DL. The expressivity of a DL, i.e. the semantics that can be represented with valid expressions, determines the complexity of the resulting model, more precisely, the complexity of the reasoning procedures within the model. Hence, DLs are structured in levels, each named with a string of capital letters that denote the allowed expressions. Having more constructors in a logic means that it is more expressive, and consequently, the computational complexity is greater.

The minimal DL usually considered is $\mathcal{AL}$, which stands for attributive concept description language. $\mathcal{AL}$ allows complex concepts to be: the top ($\top$) or the bottom ($\bot$) concept, a negation of an atomic concept ($\neg A$), a concept intersection ($C_1 \sqcap C_2$), a value restriction ($\forall R.C$), or an existential quantification ($\exists R.\top$). Complex roles cannot be defined in $\mathcal{AL}$. These complex concepts and atomic roles can be used in concept inclusion axioms ($C_1 \sqsubseteq C_2$), instance membership axioms ($a:C$), and role membership axioms ($(a,b):R$). In the following section, we describe $\mathcal{AL}$, an immediate extension of $\mathcal{AL}$, whereas other interesting DLs are introduced in Section A.3.

The Description Logic $\mathcal{ALC}$

In this section, we summarize the formal features of $\mathcal{ALC}$ [228], the DL mainly considered in this dissertation. We present $\mathcal{ALC}$ syntax and semantics as a particular case of general DLs in such a way that its definition can be easily extended to more expressive logics.

Signature. The symbols used in a DL are its signature or vocabulary. Formally, the signature is the disjoint union $\mathbf{S} = \mathbf{C} \uplus \mathbf{R} \uplus \mathbf{I}$, where $\mathbf{C} = \{A\}$ is

the set of atomic concepts (or classes); $\mathbf{R} = \{R_A\}$ the set of atomic roles (or properties); and $\mathbf{I} = \{a, b, \ldots\}$ the set of individuals (or instances). From the atomic elements in $\mathbf{S}$, new complex concepts $\mathsf{Con}(\mathbf{S}) = \{C_{(i)}, D_{(i)}, \ldots\}$, roles $\mathsf{Rol}(\mathbf{S}) = \{R_{(i)}\}$, and axioms $\mathsf{Ax}(\mathbf{S}) = \{O_{(i)}\}$ can be composed (subscripts are not used when disambiguation is not needed). By extension, the signature $\mathbf{S}(O)$ of an axiom (respectively for roles and concepts) is the set of atomic elements of $\mathbf{S}$ which are included in O (respectively R and C).

Concept and role constructors. $\mathcal{ALC}$ extends $\mathcal{AL}$ with the complete concept negation constructor. Complex concepts and roles in $\mathcal{ALC}$ are built according to the syntax rule in Table A.1. It can be observed that only atomic roles are allowed in $\mathcal{ALC}$. Given that De Morgan's laws hold, $C \sqcup D$ is a shorthand for $\neg(\neg C \sqcap \neg D)$ and $\exists R.C$ for $\neg(\forall(\neg R.C))$. Therefore, concept union and complete existential restrictions can be represented in $\mathcal{ALC}$, which is at least as expressive as $\mathcal{AL}$ plus $\mathcal{UE}$.

Table A.1: Syntax and semantics of concepts and roles in $\mathcal{ALC}$

Constructor	Syntax	Semantics
Concept constructors		
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom concept	$\bot$	$\emptyset$
Atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Concept negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Concept intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap C^{\mathcal{I}}$
Concept union ($\mathcal{U}$)	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Universal quantification	$\forall R.C$	$\{x : \forall x, (x,y) \notin R^{\mathcal{I}}$ or $y \in C^{\mathcal{I}}\}$
Existential quantification ($\mathcal{E}$)	$\exists R.C$	$\{x : \exists y, (x,y) \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$
Role constructors		
Atomic role	R_A	$R_A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Axioms. A DL ontology is a triple $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ (the TBox) and $\mathcal{R}$ (the RBox) contain, respectively, axioms about concepts and roles (terminological axioms), and $\mathcal{A}$ (the ABox) contains axioms about individuals (asserts). The signature of an ontology $\mathbf{S}(\mathcal{K})$ is the union of all the signatures $\mathbf{S}(O)$ of the axioms in $\mathcal{K}$. Accordingly, an $\mathcal{ALC}$ ontology is a DL

ontology where $\mathcal{A}$ is an $\mathcal{ALC}$-valid TBox; $\mathcal{R}$ is an $\mathcal{ALC}$-valid RBox; and $\mathcal{A}$ is an $\mathcal{ALC}$-valid ABox.

A DL TBox and in particular, an $\mathcal{ALC}$ TBox $\mathcal{T}$ consists of a finite set of general concept inclusion (GCI) axioms of the form $C \sqsubseteq D$, which means that concept C is more specific than D, or that D subsumes C. A concept definition $C \equiv D$ (C and D are equivalent) is an abbreviation of the pair of axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. Concept expressions for C and D can be derived inductively from atomic primitives using the previously mentioned $\mathcal{ALC}$ concept constructors.

In general, a DL RBox $\mathcal{R}$ consists of a finite set of role axioms stating role properties such as inclusion, transitivity, functionality, etc. (Table A.3). However, in $\mathcal{ALC}$ the RBox is assumed to be empty.

A DL ABox $\mathcal{A}$ consists of a finite set of axioms about individuals. In $\mathcal{ALC}$, these axioms can describe an individual with respect to a concept ($a : C$, which means that a is an instance of C) or a pair of individuals with respect to a role ($(a, b):R$, which means that (a, b) is an instance of R).

The set of concepts defined in an ontology is denoted as $\mathsf{Con}(\mathcal{K})$. The set of roles defined in an ontology is denoted as $\mathsf{Rol}(\mathcal{K})$.

Interpretation. An interpretation $\mathcal{I}$ of a DL ontology $\mathcal{K}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$, the domain of the interpretation, is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function which maps:

1. each individual a in $\mathcal{K}$ with an element $a^{\mathcal{I}}$,
2. each concept C in $\mathcal{K}$ with a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
3. each role R in $\mathcal{K}$ with a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

This interpretation is conveniently extended for complex concepts and roles. In $\mathcal{ALC}$, this extension is given by the inductive definitions in Table A.1.

An interpretation $\mathcal{I}$ is a model of (i.e. satisfies) the axiom:

- $a:C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
- $(a,b):R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$,
- $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$,
- an $\mathcal{ALC}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ iff it is a model for each element in $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

Other Description Logics

There are a handful of logics with different levels of expressivity in the literature about DLs. This section offers an overview of the syntax and semantics of some selected extensions. Different combinations of the extensions may lead to the same language since some of them can be mutually reducible. The formal description of the DLs already mentioned can be consulted in the corresponding references.

Concept and role constructors. Table A.2 shows the syntax and semantics of some additional role and concept constructors. In the table, S denotes a simple role. A simple role is an atomic role, the inverse of a simple role, or a role that only subsumes simple roles. The operator $\circ$ denotes the composition of binary relations.

Axioms. Table A.3 presents further role and instance axioms that can be used in expressive DLs. Observe that a DL TBox, irrespectively of the expressivity of the logic, usually only contains GCIs. Δ_D denotes a predefined datatype such as `integer`, `real`, etc.

Interpretation. Table A.2 shows the interpretation of the additional constructors presented in this section.

Extending the $\mathcal{ALC}$ case, an interpretation $\mathcal{I}$ of an expressive DL is a model of (i.e. satisfies) the axiom:

- $(a,b):\neg R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$,

Table A.2: Constructors for some extensions of $\mathcal{ALC}$

Constructor	Syntax	Semantics
Concept constructors		
Nominals ($\mathcal{O}$)	$\{a, b, \dots\}$	$\{a^{\mathcal{I}}, b^{\mathcal{I}}, \dots\}$
Unqualified number restriction ($\mathcal{N}$)	$\geq n\, R$	$\{x : \lvert\{y : (x,y) \in R^{\mathcal{I}}\}\rvert \geq n\}$
	$\leq m\, R$	$\{x : \lvert\{y : (x,y) \in R^{\mathcal{I}}\}\rvert \leq m\}$
Qualified number restriction ($\mathcal{Q}$)	$\geq n\, S.C$	$\{x : \lvert\{y : (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}\rvert \geq n\}$
	$\leq m\, S.C$	$\{x : \lvert\{y : (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}\rvert \leq m\}$
Reflexivity restriction (s)	$\exists S.Self$	$\{x \ : \ (x,x) \in S^{\mathcal{I}}\}$
Role constructors		
Inverse role ($\mathcal{I}$)	R^{-}	$\{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\}$
Role chain ($(\circ)$)	$R_1 R_2 \dots R_n$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}} \circ \dots R_n^{\mathcal{I}}$
Concrete role ((D))	T	$T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_D$
Universal role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Table A.3: Axioms for some extensions of $\mathcal{ALC}$

Axiom	Syntax
Role axioms	
Functional roles ($\mathcal{F}$)	$\mathsf{Func}(R)$
Transitive roles ($\mathcal{S}$)	$\mathsf{Trans}(R)$
Disjoint roles (s)	$\mathsf{Dis}(S, S')$
Reflexive roles (s)	$\mathsf{Ref}(R)$
Irreflexive roles (s)	$\mathsf{Irr}(S)$
Symmetric roles	$\mathsf{Sym}(R)$
Asymmetric roles	$\mathsf{Asy}(S)$
Role hierarchies ($\mathcal{H}$)	$R_1 \subseteq R_2$
Complex role inclusion ($\mathcal{R}$)	$RS \subseteq R$ $RS \subseteq S$
Instance axioms	
Negated role assertions	$(a, b) : \neg R$
Inequality assertions	$a \neq b$
Equality assertions	$a = b$

- $a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$,
- $a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$,
- $R_1 \sqsubseteq R_2$ iff $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$,
- $\mathsf{Func}(R)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ and $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in R^{\mathcal{I}}$ imply $b^{\mathcal{I}} = c^{\mathcal{I}}$,
- $R_1 \ldots R_n \sqsubseteq R$ iff $R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$,
- $\mathsf{Trans}(R)$ iff $R^{\mathcal{I}}$ is transitive,
- $\mathsf{Dis}(S, S')$ iff $S^{\mathcal{I}} \cap S'^{\mathcal{I}} = \emptyset$,
- $\mathsf{Ref}(R)$ iff $(x, x) \in R^{\mathcal{I}}, \forall x \in \Delta^{\mathcal{I}}$,
- $\mathsf{Irr}(S)$ iff $(x, x) \notin S^{\mathcal{I}}, \forall x \in \Delta^{\mathcal{I}}$,
- $\mathsf{Sym}(R)$ iff $(x, y) \in R^{\mathcal{I}}$ implies $(y, x) \in R^{\mathcal{I}}$,
- $\mathsf{Asy}(S)$ iff $(x, y) \in S^{\mathcal{I}}$ implies $(y, x) \notin S^{\mathcal{I}}$,
- a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ iff it is a model for each element in $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

Relevant Description Logics

Some commonly used DL families are:

- $\mathcal{FL}^-$, which stands for $\mathcal{AL}$ without concept negation, top, and bottom concept. Its immediate extension $\mathcal{FL}$ corresponds to $\mathcal{FL}^-$ plus a domain restriction constructor for roles [52].
- $\mathcal{EL}$, which takes a different perspective and disallows value restrictions. Thus, it provides as concept constructors only the top concept, conjunction, and (complete) existential restriction, besides concept equivalences as the only axiom [12]. $\mathcal{EL}++$, an extension of $\mathcal{EL}$ which adds the bottom concept, nominals, concrete domains, and GCIs [13], is the logic underlying the 'EL++' profile of OWL 2 (see Appendix Semantic Web).
- $\mathcal{SH}$, which extends $\mathcal{ALC}$ with transitive roles and role hierarchies [122].
- $\mathcal{SHIF}$, which extends $\mathcal{SH}$ with inverse and functional roles [121]. This logic is almost equivalent to the 'Lite' level of the standard ontology language OWL (see Appendix Semantic Web).
- $\mathcal{SHOIN}$(D), which extends $\mathcal{SH}$ with nominals, inverse roles, cardinality restrictions, and datatypes [121]. This logic is almost equivalent to the 'DL' level of the standard ontology language OWL (see Appendix Semantic Web).
- $\mathcal{SROIQ}$(D), which extends $\mathcal{SHOIN}$(D) with qualified number restrictions, disjoint roles, reflexive and irreflexive roles, role chains, complex role inclusions, universal role, local reflexivity of concepts, negated role assertions, and (in)equality assertions [119]. This logic is almost equivalent to the most-expressive and decidable level of OWL 1.1.

A.4 Reasoning in Description Logics

Basics

Reasoning within a knowledge base is the automatic procedure aimed at inferring new axioms which have not been represented and are logical consequences of the axioms represented. Usually, reasoning in DLs can be carried out with concepts in the TBox, individuals in the ABox, or TBox concepts and ABox individuals together.

In DLs, the basic reasoning task regarding concepts is concept satisfiability. Intuitively, a concept is satisfiable if it is not contradictory of the rest of the knowledge in the ontology. Another important task is concept subsumption, which infers if a concept is more general than another concept. The concept equivalence test, which determines whether two concepts are the same, and concept disjointness, which determines whether two concepts include any common individuals, are immediate extensions of the concept subsumption check.

Formally, these concept inference tasks are defined as follows:

- A concept C is *satisfiable* or *consistent* w.r.t. a knowledge base $\mathcal{K}$ if there exists some model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}}$ is not empty. Extensively, a TBox $\mathcal{T}$ is satisfiable if every axiom in $\mathcal{T}$ is satisfiable.

- A concept C is *subsumed* by a concept D w.r.t. a knowledge base $\mathcal{K}$ if every model $\mathcal{I}$ of $\mathcal{K}$ is a model of $C \sqsubseteq D$. This is denoted as $\mathcal{K} \models C \sqsubseteq D$ ($\mathcal{K}$ entails $C \sqsubseteq D$).

- Two concepts C and D are *equivalent* w.r.t. a knowledge base $\mathcal{K}$ if C is subsumed by D w.r.t. $\mathcal{K}$, and D is subsumed by C w.r.t. $\mathcal{K}$. This is denoted as $\mathcal{K} \models C_1 \equiv C_2$ ($\mathcal{K}$ entails C is equivalent to D).

- Two concepts C and D are *disjoint* w.r.t. a knowledge base $\mathcal{K}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ holds for every model $\mathcal{I}$ of $\mathcal{K}$.

Trivially, satisfiability, equivalence, and disjointness of concepts can be reduced to concept subsumption. For instance, C is unsatisfiable iff C is subsumed by $\bot$. On the other hand, if a DL allows complete negation and intersection of concepts, subsumption, equivalence, and disjointness of concepts can be reduced to the satisfiability problem [227]. For instance, C is subsumed by D iff $C \sqcap \neg D$ is unsatisfiable.

Insofar as reasoning with individuals is concerned, inferences in the ABox can be performed with respect to the whole knowledge base or by only considering the axioms in the ABox (the TBox and RBox are assumed to be empty). The basic inference task is to test if an assert of the ABox is not contradictory with the other axioms in the ontology, or particularly in the ABox. Other possible queries are to check if certain relationships between concepts, roles, and individuals hold.

Formally, the basic inference tasks with instances are defined as follows:

- An instance axiom O is *satisfiable* or *consistent* w.r.t. a knowledge base $\mathcal{K}$ if there exists at least one interpretation $\mathcal{I}$ that is a model of both O and $\mathcal{K}$. The ABox $\mathcal{A}$ is said to be consistent w.r.t. $\mathcal{K}$ if every axiom in $\mathcal{A}$ is consistent w.r.t. $\mathcal{K}$. An assert is simply consistent if the TBox and the RBox are supposed to be empty.

- An instance axiom O is said to be *entailed* by the ABox $\mathcal{A}$ if every model $\mathcal{I}$ of $\mathcal{A}$ is also a model of O. This is denoted as $\mathcal{A} \models O$ ($\mathcal{A}$ entails O). This test can be extended to be performed w.r.t. to a knowledge base $\mathcal{K}$. If O is a membership axiom $(a:C)$, this test is called instance checking.

Similarly to concept inferences, instance checking can be reduced to ABox consistency checking, given that $\mathcal{A} \models (a:C)$ iff $\mathcal{A} \cup \{\neg(a:C)\}$ is inconsistent. It has been proved that ABox consistency can be reduced to concept consistency in languages with the nominal constructor ($\mathcal{O}$) and the *fills* constructor (a concept is defined as the set of individuals which are related to an instance) [227].

It should be underlined that, when reasoning with DL ABoxes, the open world assumption stands. The open world assumption supposes that the set of axioms in a knowledge base is not complete, and consequently, new knowledge cannot be inferred inductively. In contrast, the closed world assumption supposes that the set of axioms is complete, and inductions can be safely made. For example, let us suppose an ABox containing two axioms, $hasPrescription(juan, diazepam)$ and $hasPrescription(juan, omeprazole)$ with no other knowledge about them in the TBox. With the open world assumption, the response to the query 'how many prescriptions does Juan have' would be 'unknown', whereas with the closed world assumption, the response would be 'two'.

Based on these basic inference tasks, more complex reasoning services can be offered. Usually, DL reasoners implement a *classification* procedure, which finds the place of a concept in the hierarchy, i.e., its direct subclasses and superclasses. Other non-standard inferences in DLs are described in [16]: the least common subsumer(s) of a collection of concepts, the most specific concept(s) that include(s) an instance, the rewriting of concept descriptions, and the matching of concept expressions using concept variables.

The most common algorithms for reasoning with DLs are the so-called tableau-based algorithms. The first tableau-based algorithm, proposed by Schmidt-Schauss and Smolka [228], solved the satisfiability problem for $\mathcal{ALC}$ concepts. This proposal has been adapted to more expressive extensions of $\mathcal{ALC}$, and extended to deal with ABox consistency queries, as widely explained in [17]. The complexity of the reasoning procedures using tableau algorithms depends on the complexity of the language considered, which is high even for the basic DLs (see next section). Fortunately, worst-case inferences are infrequent, and the procedures have been highly optimized to offer good execution times in the typical cases.

Complexity of reasoning

To be concise, the computational complexity of an algorithm measures the number of computational resources required to solve a certain problem. A problem P is in complexity class C if there exists an algorithm in C that solves P (then C is the upper boundary for P). A problem P is C-hard if all the problems in class C can be reduced to P polynomially (then C is a lower boundary for P). P is C-complete if it is C-hard and in C. A formal definition of complexity and an introduction to complexity classes can be found in [199].

As explained in the previous sections, DLs research focuses on the relation between the computational complexity of reasoning procedures and the expressivity of the DL used in the representation.

Reasoning in DLs has a high computational complexity. As a matter of fact, concept satisfiability with general TBoxes in the (not very expressive) DL $\mathcal{ALC}$ is an ExpTime-complete problem. Fortunately, the worst-case scenarios do not occur very often in practical applications, and thus, the reasoning algorithms behave fairly well. Basic fragments of DLs, such as DL-Lite family [60], are being studied to guarantee the tractability of the reasoning (i.e. that it can be performed in polynomial time).

The seminal work of Brachman and Levesque for $\mathcal{FL}$ [52], which formally proved the intuition that the more expressive the logic, the more complex the reasoning, has given rise to further research on complexity. An extensive review of the work dealing with complexity analysis of $\mathcal{ALC}$-based DLs can be found in Donini et al., who study the complexity of the concept satisfiability and instance check problems in $\mathcal{AL}[\mathcal{E}][\mathcal{U}][\mathcal{C}][\mathcal{R}][\mathcal{N}]$ as well as the source of this complexity [81]. Calvanese focuses on non-expressive DLs with restricted TBoxes [58], a contribution that has been completed with additional results for $\mathcal{ALCQI}$ in [59].

More expressive logics, such as $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$, have been recently studied in [254] and [119]. The Description Logic Complexity Naviga-

tor is a useful and well documented utility capable of exploring the complexity of reasoning with TBoxes and ABoxes in several extensions of $\mathcal{ALC}$ [264].

Table A.4 summarizes the complexity of reasoning tasks for some of the DLs presented in Section A.3.

Table A.4: Complexity of reasoning different DLs w.r.t. general TBoxes

Concept satisfiability / ABox consistency		
DL	**Acyclic TBoxes**	**General TBoxes**
$\mathcal{EL}++$	PTIME	PTIME
$\mathcal{ALC}$	PSPACE-complete	EXPTIME-complete
$\mathcal{SHIF}$	EXPTIME-complete	EXPTIME-complete
$\mathcal{SHOIN}$	NEXPTIME-complete	NEXPTIME-complete
$\mathcal{SROIQ}^a$	NEXPTIME-hard	NEXPTIME-hard

[a] Acyclic RBoxes are required to keep the logic decidable.

A.5 Fuzzy Description Logics

Basics

Although DLs have proved to be a very powerful formalism for knowledge representation, it is also true that ontologies cannot deal with imprecise and vague information, which is inherent to most real world domains[2]. Consequently, further extensions of DLs have been proposed in order to represent this kind of knowledge. Since Fuzzy sets and Fuzzy logic are suitable formalisms for imprecise knowledge, a promising direction is to enhance DLs with fuzzy representation mechanisms, which generate fuzzy ontologies.

A fuzzy set is a generalization of the classical notion of set. With classical (or crisp) sets, an element can either belong to a set or not, whereas with fuzzy sets, membership in a set is a question of degree. More formally, let

[2] Interestingly enough, one of them is the Web. The W3C Incubator Group on Uncertainty Reasoning for the World Wide Web (http://www.w3.org/2005/Incubator/urw3/) studies how uncertainty can be managed in the context of the Semantic Web.

X be a set of elements called the reference set. A fuzzy subset A of X is defined by a membership function $\mu_A(x)$, or simply $A(x)$, which assigns any $x \in X$ to a value in the interval of real numbers between 0 and 1. 0 means no-membership and 1 full membership (as in classic logics), but now a value between 0 and 1 represents the extent to which x can be considered to be an element of X.

Likewise, all crisp set operations are extended to fuzzy sets. The intersection, union, complement, and implication set operations are performed by a t-norm function $\otimes$, a t-conorm function $\oplus$, a negation function $\ominus$, and an implication function $\Rightarrow$, respectively. Analogously, fuzzy relations define a partial association between two or more elements. In this work, we use Zadeh's family of functions and Gödel's implication, which are defined as follows:

$$
\begin{array}{rrcl}
\text{Zadeh t-norm} & \alpha \otimes \beta & = & \min\{\alpha, \beta\} \\
\text{Zadeh t-conorm} & \alpha \oplus \beta & = & \max\{\alpha, \beta\} \\
\text{Łukasiewicz negation} & \ominus\alpha & = & 1 - \alpha \\
\text{Kleene-Dienes implication} & \alpha \Rightarrow \beta & = & \max\{1 - \alpha, \beta\} \\
\text{Gödel implication} & \alpha \Rightarrow \beta & = & \begin{cases} 1, & \text{if } \alpha \leq \beta \\ \beta, & \text{if } \alpha > \beta \end{cases}
\end{array}
$$

A comprehensive introduction to Fuzzy Logic, including a detailed explanation of these operations, can be found in [146].

Several fuzzy extensions to DLs are described in [165]. They can be differentiated by the fuzzy semantics added to the DL constructors, the expressivity of the base logic, and the family of fuzzy set operators considered. For instance, FuzzyOWL [243] is a proposal for extending the OWL language (i.e., $\mathcal{SHOIN}(\mathrm{D})$) so that it has fuzzy capabilities. Since it is beyond of the scope of this appendix to provide a full introduction to Fuzzy DLs, we refer the reader to [225] for further studies on the topic.

Briefly, fuzzy DLs (fDLs) extend DLs by allowing concepts to denote fuzzy sets of individuals and roles to denote fuzzy binary relations. The notion of interpretation is extended to the fuzzy case in such a way that: (i) an

individual of the domain may belong to a concept to a certain degree in $[0,1]$; (ii) a pair of individuals of the domain may belong to a role to a certain degree in $[0,1]$. The semantics of the constructors used to build non-atomic concepts and roles are conveniently adapted; e.g. the semantics of the concept intersection are given by a t-norm function. Axioms are also extended to the fuzzy case, holding to a degree. For example, given two fuzzy concepts, a terminological axiom may be asserted to define a fuzzy inclusion relation between them.

In a fuzzy DL we can define, for instance, $BloodBorneDiseases$ as the set of diseases that can be spread by contamination of blood. $hepatitisC$ is a full member of this set (degree equals to 1), whereas $nephropathiaEpidemica$ also belongs to the set, but to a somewhat lesser degree (equal to 0.7). Similarly, two individuals can be partially related through a role: *causes(hepatitisC, liverCancer)* with degree 0.6. Other axioms may also be fuzzified, e.g. GCIs: *BloodBorneDiseases* is a subset of *InfectiousDiseases* with degree 0.7 because some bloodborne diseases are infectious, whereas others are not.

The fuzzy DL $f\mathcal{ALC}$

In this section, we shall consider the fuzzy DL $f\mathcal{ALC}$, used in Chapter 4 of this dissertation. This logic was firstly described in [246]. We shall follow the formulation in [39, 41, 40], which is based on the former, restricting to $f\mathcal{ALC}$ the more expressive fDLs $f\mathcal{SHOIN}$ and $f\mathcal{SROIQ}$ proposed in them.

Complex concept and role expressions in $f\mathcal{ALC}$ have the same generation grammar as in $\mathcal{ALC}$ (Table A.5). Axioms in $f\mathcal{ALC}$ are the fuzzy extension of the crisp asserts in $\mathcal{ALC}$. Let $\rhd = \{\geq, >\}$, $\lhd = \{\leq, <\}$, and $\alpha \in [0,1]$. Thus:

- A $f\mathcal{ALC}$ TBox consists of *fuzzy GCIs*, which constrain the truth value of a GCI. Fuzzy GCIs are expressions of the form $\langle C \sqsubseteq_{\rhd\alpha} D\rangle$, denoting that C is included in D with degree α.
- A $f\mathcal{ALC}$ RBox is empty.

- A $f\mathcal{ALC}$ ABox consists of a finite set of *fuzzy assertions*, which constrain the truth value of an assert. A fuzzy assertion can be an expression of the form $\langle a : C \rhd \alpha \rangle$ (a is a member of C with degree $\rhd \alpha$), $\langle a : C \lhd \alpha \rangle$ (a is a member of C with degree $\lhd \alpha$), or $\langle (a,b) : R \rhd \alpha \rangle$ ((a,b) are related through R with degree α).

It can be observed that negative GCIs or negative role membership axioms are not allowed.

Obviously, concept and role interpretations have fuzzy semantics. Concepts are fuzzy sets of individuals and roles are fuzzy relations between pairs of individuals. A $f\mathcal{ALC}$ interpretation maps every individual a onto an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$; every concept C onto a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$; and every role R onto a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$. For a t-norm $\otimes$, a t-conorm $\oplus$, a negation function $\ominus$ and an implication function $\Rightarrow$, Table A.5 depicts the semantics of the interpretation of concept and roles in $f\mathcal{ALC}$.

Table A.5: Syntax and semantics of concepts and roles in $f\mathcal{ALC}$

Constructor	Syntax	Semantics
Concept constructors		
Top concept	$\top$	1
Bottom concept	$\bot$	0
Atomic concept	A	$A^{\mathcal{I}}(x)$
Concept negation	$\neg C$	$\ominus C^{\mathcal{I}}(x)$
Concept intersection	$C \sqcap D$	$C^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x)$
Concept union	$C \sqcup D$	$C^{\mathcal{I}}(x) \oplus D^{\mathcal{I}}(x)$
Universal quantification	$\forall R.C$	$\inf_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\}$
Existential quantification	$\exists R.C$	$\sup_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)\}$
Role constructors		
Atomic role	R_A	$R_A^{\mathcal{I}}(x,y)$

A fuzzy interpretation $\mathcal{I}$ is a model of (satisfies):

- $\langle a : C \rhd \alpha \rangle$ iff $C^{\mathcal{I}}(a^{\mathcal{I}}) \rhd \alpha$,
- $\langle a : C \lhd \alpha \rangle$ iff $C^{\mathcal{I}}(a^{\mathcal{I}}) \lhd \alpha$,

- $\langle (a,b):R \rhd \alpha \rangle$ iff $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \rhd \alpha$,
- $\langle C \sqsubseteq_{\rhd \alpha} D \rangle$ iff $\inf_{x \in \Delta^{\mathcal{I}}} \{C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)\} \rhd \alpha$,
- a fKB $f\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ iff it satisfies each element in $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

We assume that there are no fuzzy axioms of the form $\tau \geq 0$, $\tau \leq 1$ (which are tautologies) or $\tau > 1$, $\tau < 0$ (which are obvious inconsistencies).

An axiom τ is a *logical consequence* of a knowledge base $\mathcal{K}$, denoted $\mathcal{K} \models \tau$ iff every model of $\mathcal{K}$ satisfies τ. The greatest lower bound (g|b) of a fuzzy axiom τ is defined as the $\sup\{\alpha : \mathcal{K} \models \langle \tau \geq \alpha \rangle\}$.

Due to the standard properties of the fuzzy operators, the following concept equivalences hold [246]: $\neg\top \equiv \bot$, $\neg\bot \equiv \top$, $C \sqcap \top \equiv C$, $C \sqcup \bot \equiv C$, $C \sqcap \bot \equiv \bot$, $C \sqcup \top \equiv \top$, $\exists R.\bot = \bot$, $\forall R.\top = \top$. Moreover, $f\mathcal{ALC}$ allow some sort of *modus ponens* and *chaining* of GCIs:

Proposition 5. *For $\alpha, \beta \in [0,1]$ and $\rhd = \{\geq, >\}$, the following properties are verified:*

(i) $\langle a : C \rhd \alpha \rangle$ *and* $\langle C \sqsubseteq D \rhd \beta \rangle$ *imply* $\langle a : D \rhd \alpha \otimes \beta \rangle$.

(ii) $\langle C \sqsubseteq_{\rhd \alpha} D \rangle$ *and* $\langle D \sqsubseteq_{\rhd \beta} E \rangle$ *imply* $\langle C \sqsubseteq_{\rhd \alpha \otimes \beta} E \rangle$.

Reasoning in fuzzy Description Logics

Fuzzy DLs cause crisp reasoning procedures (like tableau-based algorithms, Section A.4) to no longer be valid. Consequently, in order to perform reasoning tasks with them, new inference algorithms must be developed. This is the approach taken is most research in the area. The first contribution in this sense was fuzzyDL [44], a reasoner for $f\mathcal{SHOIN}$ which additionally supports GCIs, Zadeh and Łukasiewicz semantics, and explicit definition of fuzzy concepts with triangular and trapezoidal membership functions. A related approach is presented in [242], where a reasoning algorithm for the

expressive fuzzy DLs $f\mathcal{SI}$ and $f\mathcal{SHIN}$ (with Kleene-Dienes semantics) is described.

Nevertheless, an alternative, which has been explored by fewer authors, is to define reduction procedures with a view to transforming fuzzy representations to equivalent crisp ones. This would be done in such a way that existing algorithms and inference engines can be used to carry out reasoning tasks. The first research in this direction was done by Straccia, who developed a reasoning preserving procedure for fuzzy $\mathcal{ALC}$ [247, 245]. This approach was extended by Bobillo, Delgado, and Gómez-Romero, who successively consider the more expressive fuzzy DLs $f\mathcal{SHOIN}$ [39], $f\mathcal{SROIQ}$ [41] and $f\mathcal{SROIQ}$(D) [40].

APPENDIX B

The Semantic Web

This appendix is an introduction to the Semantic Web. The first section describes the main objectives of the Semantic Web initiative, as well as the overall structure of the Semantic Web in relation to the current Web. The second and third sections offer an overview of RDF and OWL languages, with a special focus on OWL. The fourth section describes other Semantic Web technologies such as ontology editors, inference engines, APIs, and database publishing tools. After an explanation of Semantic Web Services, the semantic extension of Web Services, the appendix concludes with a reflection on the future of the Semantic Web.

B.1 Basics

The Semantic Web is an extension of the current Web, whose aim is the automation of document processing and information retrieval. The Semantic Web appears to solve certain problems of the current web stemming from the large quantity of available resources and the limitations of search engines. The overabundance of documents on the Web makes it difficult to locate the most relevant ones for a specific user query. Web languages are only able

to describe the syntax of the documents. This results in manual information integration processes.

In their seminal paper on the Semantic Web, Berners-Lee, Hendler, and Lassila propose improving the Web by giving information a "well-defined meaning, better enabling computers and people to work in cooperation" [27]. The *meaning* of web resources is represented by means of formal metadata describing their semantics. Consequently, the Semantic Web requires new standards and technologies to create and publish such metadata, which are managed by software agents to find, discover, locate, and integrate information better than today's lexical search engines.

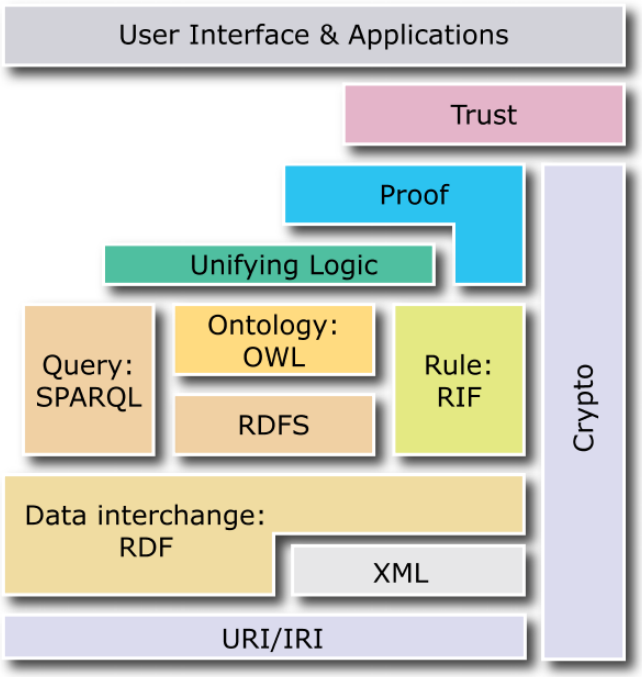
Semantic Web research is coordinated by the World Wide Web Consortium (W3C)[1], an international organization devoted to the creation of standards and guidelines for the Web. Under the direction of T. Berners-Lee, various academic institutions and corporations participate in the elaboration of these standards. Semantic Web Activity[2] embodies the W3C groups working on the Semantic Web: languages for metadata (RDF, OWL), rule-based representations, applications, etc. *Recommendations* (i.e. the final specifications endorsed by W3C members), are the final deliverables produced by the groups as the result of their discussions.

The Semantic Web is implemented as a successive set of layers of various levels of abstraction, which have been developed by relying on the Web standards. Figure B.1 shows the most recent "layercake" diagram[3], which is explained in [230].

1. The URI/IRI layer offers the most basic support for the Semantic Web, and pertains to the identification of resources. URIs (Universal Resource Identifiers) and IRIs (Internationalized Resource Identifiers) are used to name entities in the Semantic Web, which can be documents, people, places, or anything else worth representing.

[1] http://www.w3.org/
[2] http://www.w3.org/2001/sw/
[3] (http://www.w3.org/2007/03/layerCake.png)

Figure B.1: Semantic Web layers



2. The XML layer provides a metalanguage to define the syntax of Semantic Web languages. The XML grounding facilitates the interoperability of languages.

3. The RDF layer provides the basic means to associate semantics to Web entities. RDF is a simple language to state assertions regarding entities in the form of 'object-attribute-value' triples (see Section B.2).

4. The next layer contains languages that improve RDF expressivity. The query language SPARQL allows complex consultations of RDF repositories. RDFS adds certain constructs to RDF in order to represent basic ontological semantics, for example, class-subclass relations. OWL is the W3C language used for ontological knowledge representation. RIF is a specification (still under elaboration) of a rule interchange language that will ultimately provide a common framework for rule management in the Semantic Web.

The development of the uppers layers of the Semantic Web layercake has not progressed very far for the moment. In the future, the Unifying Logic layer will be responsible for creating a unified view of the Web, described in terms of the lower-level languages. The Trust layer, relying on the Proof and the Crypto layers, will evaluate the reliability of the results obtained for specific knowledge sources.

Semantic Web applications are gaining momentum as standards are being published and proof-of-concept systems are showcasing their benefits. Understandably, the most interested parties in Semantic Web technologies are application domains in which the integration of very different information is extremely difficult. This is the case of Health and Life Sciences, which are considered as one of the early adopters of the Semantic Web [90]. They even have their own special interest group[4] within the W3C Semantic Web Activity.

The Semantic Web has solid Artificial Intelligence underpinnings, mainly based on Knowledge Representation and Description Logics. This has been a source of both agreement and dissension. Certain authors have argued that it is simply not feasible to build generals ontologies to support the Semantic Web, because people cannot be forced to use these ontologies and logical syllogisms are not sufficient in themselves to capture and represent knowledge (see for instance [232]). These arguments have been refuted by Semantic Web supporters, who claim that this is a misunderstanding of the Semantic Web proposal [113].

We believe that both claims have pros and cons, but independently of the philosophical differences between supporters and detractors, the Semantic Web is a source of valuable technologies for Knowledge Mobilization. Representation formalisms, especially RDF and OWL language (Sections B.2 and B.3), as well as methodologies and tools (Section B.4), can be used to deal with KMob issues.

[4] http://www.w3.org/2001/sw/hcls/

B.2 The Resource Description Framework (RDF)

RDF (Resource Description Framework) is the W3C standard language to describe resources in the Semantic Web [167]. RDF allows metadata to be asserted in the form of triples, i.e. statements relating an object, a property, and a value. For instance, it can be stated that `[Juan]` (subject) `[has email address]` (predicate) `[jgomez@decsai.ugr.es]` (object). Subjects, predicates and objects are identified by their URI.

A set of triples describing the same subject or subjects connected with predicates takes the form of an RDF graph. RDF statements can be reified, which means that an assertion can be assigned a URI and participate as the subject or object of another assertion. Likewise, a complete RDF graph can be reified and assigned a URI.

RDF graphs can be serialized (i.e., represented as plain text) in several formats. One of them is the standard XML-based syntax, which is often criticized for its verbosity. Other RDF serialization formats are N3, proposed by the W3C itself; Turtle, a subset of N3; and N-Triples, a simplification of N3. Most Semantic Web APIs support the reading and writing of RDF documents in these formats.

RDF data are stored in RDF repositories, which can be persistent or volatile. A persistent repository is an RDF storage that, instead of being maintained only in volatile memory, is saved on the hard disk, or more precisely, in a database. The contents of the in-memory model and the persistent model are linked in such a way that changes in one of them are automatically reflected in the other. This feature notably improves the performance of an application when large models are involved.

RDF repositories can be consulted with SPARQL, the standard query language and protocol for RDF [210]. SPARQL is similar to the database query language SQL, although its syntax is more logic-oriented since free variables can be included in the queries. SPARQL does not make inferences on the RDF repository. This means that only explicit knowledge is retrieved in SPARQL

queries.

RDF has been used to define vocabularies that can be used as metadata templates. One example is the FOAF (Friend of a Friend) terminology, which includes terms such as *person* or *knows*, and can describe social networks with RDF. The Dublin Core, a vocabulary for the description of documents with terms like *author* or *title*, also has an RDF version.

Notwithstanding, the extreme simplicity of RDF representation primitives makes them insufficient in certain scenarios. RDF Schema (RDFS) is an extension of RDF with additional constructs: range, domain, subproperty, subclass, etc. [175]. RDFS lies halfway between RDF, which is too simple for representing complex knowledge, and OWL, whose logical complexity makes it too difficult for non-specialized users.

B.3 The Web Ontology Language (OWL)

Definition

Ontologies are formal specifications. This means that ontological knowledge must be encoded with a well-defined representation language. Since the ontologies first appeared in Knowledge Engineering, various languages have been proposed [103].

Currently, the most widely used ontology language is the Web Ontology Language (OWL), a standard created by the W3C Semantic Web Activity. OWL provides the syntax and semantics needed to represent the information in Web documents, and to process it automatically. OWL is a W3C recommendation since 2004 [176]. OWL has been influenced by other formalisms [120], including Description Logics (DLs). Thus, an OWL knowledge base contains descriptions of classes, roles and individuals.

The OWL specification has three species or dialects. Each is equivalent to a different type of DL, and has a different level of computational complexity.

OWL Lite is the least expressive level of OWL. It corresponds to the DL $\mathcal{SHIF}(\mathrm{D})$. OWL Lite is used for the creation of thesauri and simple taxonomies. Reasoning with OWL Lite has ExpTime complexity, which makes it the most efficient of the three levels.

OWL DL is more expressive than OWL Lite yet complete (all the inferences are computable) and decidable (all the inferences end). OWL DL includes all the constructors of OWL, but they have certain restrictions. For instance, a class cannot be an instance of another class, and new datatypes cannot be created. Since OWL DL is almost equivalent to $\mathcal{SHOIN}(\mathrm{D})$, its complexity is NexpTime.

OWL Full is the most expressive dialect, but it is not decidable. OWL Full embodies the Lite and DL levels, and allows the free mixing of OWL and RDF. Therefore, any RDF document is a valid OWL Full document.

Syntax

OWL normative syntax is based on the XML syntax of RDFS. An OWL document is a set of XML tags with the structure and semantics stated by the specification. Usually, an OWL file has three parts: a heading, classes and properties definitions, and asserts about individuals. OWL syntax is detailed in [21].

An OWL heading includes previous statements about the ontology, for instance, definitions of the namespaces used in the document or the additional models that are imported.

An OWL class definition specifies which individuals are members of the class. In other words, class definitions are axioms about the classes. These axioms are constructed using class descriptions, which is the name that the OWL specification uses for complex class expressions built with the OWL constructors. The syntax of class descriptions and axioms is shown in Table B.1, which also includes the level at which the expression is allowed.

Table B.1: OWL class descriptions and axioms

	Syntax	**Level**
Class descriptions		
Class identifier	`rdf:ID`	Lite
Exhaustive enumeration	`owl:oneOf`	DL
Restrictions on properties	(`owl:Restriction`)	
	`owl:someValuesFrom`	Lite, with restrictions (w.r.)
	`owl:allValuesFrom`	Lite, w.r.
	`owl:hasValue`	DL
	`owl:cardinality`	Lite, w.r.
	`owl:minCardinality`	Lite, w.r.
	`owl:maxCardinality`	Lite, w.r.
Intersection	`owl:intersectionOf`	Lite, w.r.
Union	`owl:unionOf`	DL
Complement	`owl:complementOf`	DL
Class axioms		
Inheritance	`rdfs:subClassOf`	Lite
Equivalence	`owl:equivalentClass`	Lite
Disjointness	`owl:disjointWith`	DL

OWL properties define relations between individuals of the ontology (object properties: `owl:ObjectProperty`) or between individuals and values of an XML datatype (datatype properties: `owl:DatatypeProperty`). There are two additional types of properties, annotation properties (`owl:AnnotationProperty`) and ontology properties (`owl:OntologyProperty`), which are used to specify certain metaproperties of the knowledge base.

OWL does not allow complex property descriptions to be created, and property axioms are built using only simple property names. The syntax of property axioms is shown in Table B.2.

Axioms about individuals are called facts. Facts include a name assignment (if the individual is not anonymous), class membership information, and role values. OWL does not assume that two individuals with different names are different. This assumption is known as UNA (Unique Name Assumption). Therefore, facts can also state information about the identity of an individual: `owl:sameAs`, `owl:differentFrom`, `owl:AllDifferent`.

XML-based syntax is the standard OWL syntax, but other codifications have been proposed. Of these codifications, the Manchester syntax is one of the most widely used. It has the advantage of not being as verbose of XML, and its notation is simpler than that of Description Logics [117]. It is based on the OWL abstract syntax, described in the OWL recommendation [201], and the Description Logics syntax, but without the mathematical symbols.

B.4 Semantic Web technologies

In this section, selected Semantic Web technologies are studied. These technologies are very helpful to manage ontologies in KMob systems. Extensive reviews of Semantic Web technologies can be found in [76, 103]. A list of links on Semantic Web technologies is maintained at the SemanticWeb.org wiki page[5].

[5] http://www.semanticweb.org/wiki/Tools

Table B.2: OWL property axioms

	Syntax	Level
Property axioms		
Inheritance	`rdfs:subPropertyOf`	Lite/DL, w.r.
Domain and range	`rdfs:domain`	Lite, w.r.
	`rdfs:range`	Lite, w.r.
Relations to other properties	`owl:inverseOf`	Lite
	`owl:equivalentProperty`	Lite
Global cardinality constraints	`owl:FunctionalProperty`	Lite
	`owl:InverseFunctionalProperty`	Lite
Logical property characteristics	`owl:SymmetricProperty`	Lite
	`owl:TransitiveProperty`	Lite/DL w.r.

Ontology editors

Ontology editors facilitate the creation of Semantic Web ontologies. Ontology editors provide a friendly interface that saves ontology developers from being forced to edit the raw RDF or OWL code of the knowledge base.

One of the most extended environments is Protégé [192], an open-source platform for knowledge management created at the University of Stanford[6]. Protégé is an integrated development environment with an associated methodology that supports the implementation of ontology-based applications. Protégé plug-ins widen the scope of editor capabilities, allowing ontology visualization or exporting.

Originally, Protégé had its own knowledge metamodel, which was a frame-based representation formalism compliant with the OKBC (Open Knowledge-Base Connectivity) protocol [190]. This metamodel could be adapted to other ontology languages such as RDF or OWL. Thus, Protégé did not natively support OWL an additional complement, namely the OWL plug-in [147], was required.

As of version 4, Protégé has been completely reimplemented, and OWL is natively supported by means of the OWLAPI [116] (formerly WonderWeb OWL API). Reasoning in Protégé 4 is performed with the bundled Pellet reasoner, though the DIG interface [20], which was used in the previous versions to communicate the IDE with external reasoners, is also supported.

Protégé can be used as a stand-alone application or as a package inside another application. In the stand-alone mode, Protégé is an ontology editor capable of creating, editing, or reasoning with ontologies. On the other hand, Protégé Java libraries can be included in any application, and the functionalities of core Protégé and Protégé plug-ins can be accessed from the application.

Protégé is a very valuable tool, because it can be used by non-expert users to create ontologies without obliging them to deal with OWL syntax. Never-

[6] http://protege.stanford.edu

theless, it has been often criticized because it can be unstable and resource-consuming, despite the fact that version 4 solves some of these problems.

Other OWL editors have been created to overcome the problems in Protégé. Of these editors, TopBraid Composer is one of the most complete. TopBraid Composer is a visual modeling tool for developing ontologies commercially distributed by TopQuadrant Inc[7]. TopBraid Composer provides support for W3C's Semantic Web standards (RDFS, OWL) and different data backends (Oracle 11g, Sesame). One of its main features is that it simplifies the integration of information sources external to the models created with the editor. This is the case of DBPedia [11], an RDF repository based on Wikipedia, or Google Maps data, which can be easily imported from the editor to enrich an ontology. These combinations of data sources are called mash-ups.

Inference engines

Reasoning with ontologies is performed by reasoning engines, which are software programs that implement optimized versions of tableau-based algorithms. Currently, there are several engines able to deal with $\mathcal{SHOIN}$(D), the DL underlying OWL. Some of these engines are listed on in U. Sattler's web page[8]. This section presents three widely used tools capable of managing the OWL language, namely RacerPro, FaCT and Pellet.

RacerPro. RACER (*Renamed ABox and Concept Expression Reasoner*) [108] was originally developed at the University of Hamburg, and is now commercially available from Racer Systems GmbH & Co with the name of RacerPro[9]. In its version 1.9.2, RacerPro implements the DL $\mathcal{ALCQHI}_{R+}$ with datatypes, equivalent to $\mathcal{SHIN}$(D), and consequently, to OWL without the `one-of` constructor.

[7] http://www.topquadrant.com/topbraid/composer/index.html
[8] http://www.cs.man.ac.uk/~sattler/reasoners.html
[9] http://www.racer-systems.com/products/racerpro/index.phtml

RacerPro allows several kinds of TBox and ABox queries to be executed. Such queries can be both standard and non-standard DL queries. Additionally, Racer provides a consulting language called nRQL, which supports algebraic (in)equations, string expressions, numerical constraints, or negation as failure (i.e. a limited version of the closed world assumption) among other features. Rules can also be managed in RacerPro. The engine can be remotely invoked through a DIG interface, a standard for communicating with DL reasoners through HTTP message-passing [20].

FaCT. FaCT (*Fast Classification of Terminologies*) [118] is an inference engine developed by Ian Horrocks at the University of Manchester, and is available online[10]. FaCT implements two reasoners, one for the logic $\mathcal{SHF}$ and another for the logic $\mathcal{SHIQ}$.

FaCT offers features similar to those in RACER: (i) an expressive language for the formulation queries; (ii) reasoning with multiple knowledge bases; (iii) optimizations that improve the tableau-based algorithms implemented. FaCT also provides a client-server architecture based on CORBA to access the ontologies. A new version of FaCT is now available, i.e. FaCT++[11], which incorporates further improvements and full support for $\mathcal{SHOIQ}$(D).

Pellet. Pellet [234] is an open-source OWL reasoner, originally implemented by the research group Mindswap at the University of Maryland and currently distributed by Clark & Parsia LLC[12].

Pellet was the first wide-scale reasoner to implement complete reasoning procedures for $\mathcal{SHOIN}$(D), and consequently, for OWL DL. The current version 1.4 includes all the features of $\mathcal{SROIQ}$(D), the logic underlying OWL 1.1. Furthermore, it supports DIG communication, partial management of SWRL rules, and SPARQL queries. The authors have recently included an extension to Pellet called Pronto[13], capable of reasoning with probabilistic ontologies.

[10] http://www.cs.man.ac.uk/~horrocks/FaCT/
[11] http://owl.man.ac.uk/factplusplus/
[12] http://pellet.owldl.com/
[13] http://clarkparsia.com/weblog/2007/09/27/introducing-pronto/

APIs

Semantic Web programs need to manage ontologies. Applications have to read, write, update, and query ontologies, as well as be able to carry out different tasks depending on concepts, relations, axioms, etc. Semantic Web APIs are programming libraries that implement these functions with a view to dealing with Web ontologies.

As previously mentioned, Protégé provides a graphical editor plus a Java library to manage ontologies that can be used in stand-alone applications. The most recent versions of Protégé are based on the open-source library OWLAPI, which can be included in our own software as well.

The OWLAPI offers different functionalities to Semantic Web application developers. It allows efficient reading and parsing of OWL ontologies expressed in any of the possible OWL syntaxes: XML-based, Manchester, Functional, etc. It works properly with large in-memory models, which can be edited and exported conveniently. Moreover, the OWLAPI includes interfaces for accessing reasoning services provided by the DL reasoners Pellet and FaCT++, as well as other engines compliant with the DIG specification.

The OWLAPI is a relatively recent initiative, but it is not the only API that can be used to build Semantic Web applications. Actually, up until now the most widely-used Semantic Web API has been Jena [62], which is an open-source library developed at the HP Laboratories[14] that focuses on RDF management.

Jena supports reading, parsing, and writing RDF models. It also allows the consultation of RDF ontologies using the SPARQL language with its query engine ARQ. Jena provides a mechanism to add 'if-then' rules to RDF models, which makes possible certain kind of logical inferences. Another notable feature of Jena is that persistent RDF repositories can be used transparently.

Besides RDF management features, Jena also offers support for OWL ontologies. Although it is more complex to use than the OWLAPI, OWL mod-

[14] http://jena.sourceforge.net/

els can be managed with Jena. Jena interfaces for OWL ontologies permit logic inference with the models via the built-in reasoners or external DIG-compliant inference engines.

Database publishing

The Semantic Web promises better integration of information sources thanks to the use of semantic metadata. A scenario which would greatly benefit from Semantic Web integration features is corporative database management. Frequently, corporative databases that store large volumes of data have to be combined with other internal or external information sources, which is a hard task. In fact, publishing relational data in the Semantic Web was considered a design issue in the early research in this area [26].

The RDB2RDF Incubator Group [15] was created within the W3C Semantic Web Activity to study the synergies between databases and the Semantic Web. The objectives of this group are: (i) to review existing approaches for mapping relational data onto RDF; (ii) to review existing approaches for mapping OWL classes onto relational data.

D2RQ [32, 33] and Virtuoso RDF views [34] are two proposals for mapping relational data onto RDF. They resemble each other in that both provide the means to define a declarative mapping between a database and an ontology. They allow the database to be accessed transparently as though it were an RDF/OWL ontology. This has several advantages: (i) data can be stored in a relational database, which is more efficient; (ii) new features are implemented as a layer on top of the existing applications, which do not need to be modified; (iii) data is not duplicated, which spares the user the necessity of implementing procedures to guarantee data integrity and consistence.

The main difficulty of D2RQ and Virtuoso RDF Views is that mappings must be created manually, which is often costly in certain cases. Hence, other authors have proposed unsupervised methods to establish these kinds

[15] http://www.w3.org/2005/Incubator/rdb2rdf/

of mapping. Worth mentioning are RDQuery [155], SquirrelRDF [241] and SPASQL [209], which automatically translate SPARQL queries to SQL using simple heuristics. A complete list of related works is presented in the ESW Wiki[16], maintained by the W3C staff.

B.5 Semantic Web Services

Semantic Web Services are the extension of Web Services in the context of the Semantic Web [177]. This section provides a brief overview on Web Services standards and Semantic Web Services proposals is provided.

Web Services

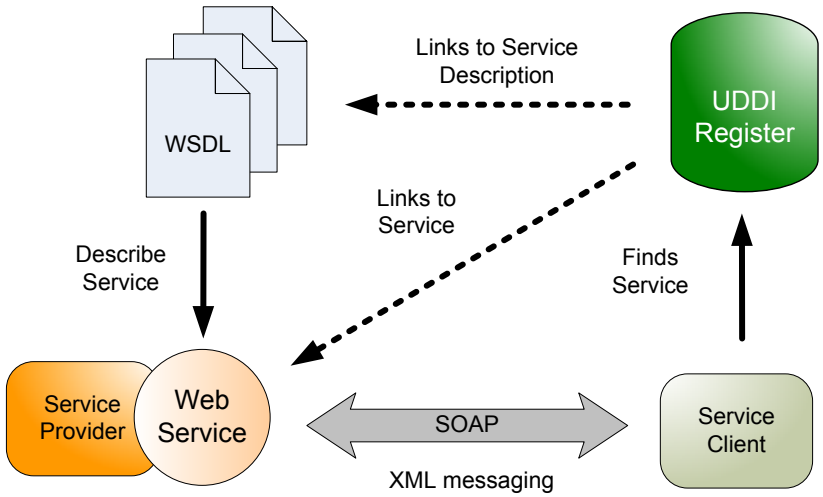
According to the definition of the W3C, a Web Service is an abstraction that allows access to remote applications using XML and HTTP-based standards [47]. Some benefits of using Web Services are availability (because web protocols are omnipresent) and extensibility (because implementation details are kept separate from the external interface, and the incorporation of new functionalities is thus easier).

Web Services are implemented with a combination of three standard protocols [75, 186]: SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery, and Integration).

Communication between Web Services is performed throughout XML messages. Actually, processing in a service may be regarded as the transformation of an input XML document into an output XML document. SOAP is the most widely used protocol for Web Services message-passing. Clients and servers create SOAP messages that wrap the data to be sent, and then transmit them using a web protocol (usually HTTP). The receiver of the message is responsible for interpreting the SOAP message and answering it.

[16] http://esw.w3.org/topic/RdfAndSql

Figure B.2: Web Services protocols



The description of Web Services is performed with the XML-based language WSDL. A Web Service description is a set of annotations that explain the features of the service with a well-defined language. WSDL abstractly describes the operations that a service can carry out and how they should be invoked. It also specifies the meaning of these descriptions in terms of concrete message formats and communication protocols.

Web Service descriptions may include additional information about service cooperation, for instance, choreography (which messages are created when the service is requested) or orchestration (which external services are required to complete the task). Some languages have been proposed to describe the interaction between services such as WSBPEL [134] (formerly BPEL4WS) and WS-CDL [137].

The location of Web Services is achieved with Services Registries, which are repositories where providers publish the descriptions of their services to make them accessible to clients. UDDI is a standard created by OASIS[17], and which specifically targets this objective. UDDI defines the structure of a well-formed service description and a set of programming interfaces to access the registry. The main disadvantage of UDDI is its poor search mechanism since

[17] http://www.oasis-open.org/

service matching is done by lexical matching of textual service descriptions and client preferences.

Semantic Web Services proposals

Semantic Web Services propose the enhancement of syntactic Web Service descriptions in WSDL with additional metadata. These metadata can be an explanation of the service capabilities, a detailed view of its processing flow, quality parameters, information about its creator, etc.

The objective of Semantic Web Services is to automate location, execution and composition of Web Services [177]:

- Automatic service discovery. A service satisfying a description of desirable features can be located by a search agent.
- Automatic service execution. Formal annotations describe the interface to access a service, and therefore a broker agent can invoke it.
- Automatic composition of services. Preconditions and effects of a service can be described semantically and then used by a planning agent to integrate different services.

Metadata in Semantic Web Services must be encoded with a formal language, which should be sufficiently expressive to represent assorted knowledge domains. Several approaches have been proposed for Semantic Web Services [43]. The most outstanding are OWL-S, WSMO, SWSF and WSDL-S, which have been submitted to W3C to be considered in the development of a standard for Semantic Web Services [43].

These approaches are conceptually similar. They propose the use of an ontology to semantically describe the features of a service. Semantic Web Services ontologies contain concepts such as *service*, *interface*, *precondition*, *effect*, *communication protocol*, etc. The ontological description of a service is

published in a registry where matching between client requirements and service capabilities is carried out. Interoperability between services is achieved by transforming service outputs in inputs interpretable by another service. This translation can be performed automatically, since mediation services *understand* the meaning of the parameters.

OWL-S (OWL for Services) is defined as an OWL ontology for annotating Web Services [171]. It has been designed to be compliant with Web Services standards. OWL-S vocabulary represents three aspects of Web Services: (i) the *profile*, which is a basic description of the capabilities of the service; (ii) the *process model*, which explains in more detail how the service works; (iii) the *grounding*, which determines how the service is implemented.

WSMO (Web Service Modeling Ontology) is another model to semantically describe Web Services [88]. It is based on the WSMF (Web Services Modeling Framework), a previous work which defines a conceptual framework for the creation Semantic Web Services. WSMO is especially interested in solving the integration issues that appear in Web Service applications. The WSMO initiative includes two working groups, WSML (Web Service Modeling Language) and WSMX (Web Service Execution Environment), whose objective is the creation of a formal language and an execution environment for WSMO. The WSMO model provides four primitives to describe services: (i) Ontologies, which are the vocabulary used in the descriptions; (ii) Web Services, with associated interfaces, capabilities, ontologies, etc.; (iii) Goals, which are compared with user requirements during the matching process; (iv) Mediators, which connect heterogeneous components of the system.

SWSF (Semantic Web Services Framework) is a proposal for the development of enhanced Web Services, which combines features from OWL-S and WSMO [18]. The SWSF model has two components: the Semantic Web Services Language (SWSL) and the Semantic Web Services Ontology (SWSO). SWSL is composed of two sublanguages: SWSL-FOL, for ontology creation, and SWSL-Rules, for First Order Logic reasoning with SWSL ontologies. The SWSO expressed in SWSL-FOL is called FLOWS (First Order Logic Ontology for Web Services). FLOWS has a structure similar to OWL-S, and also in-

cludes three main sub-ontologies: Service Descriptors, Process Model, and Grounding.

WSDL-S [2] is a language for describing Semantic Web Services that claims to overcome some drawbacks of OWL-S and related proposals. WSDL-S extends WSDL with semantic descriptors that reflect OWL-S concepts, but it does not impose an ontology representation language. WSDL-S has been designed to be compatible with existing Web Services standards, mainly WSDL. Therefore, though it does not have the formal underpinnings of other approaches, experienced Web Service developers find it easier to use, and supporting tools can be upgraded effortlessly.

Differences between these proposals are mainly due to the fact that each Semantic Web Service language focuses on a certain problem. OWL-S is especially concerned with simplicity, whereas WSMO targets integration issues. As a result, OWL-S is easier to use, and WSMO offers more expressive primitives to describe service workflow. SWSF also tackles this problem and defines a formalism based on PSL (Process Specification Language) to model service workflow. WSDL-S looks for compatibility with current standards, which makes it suitable for previous service-oriented applications that need to be enhanced with semantic features.

Regarding the ontology language used, OWL-S is an OWL DL ontology, with its corresponding computational properties. WSML is based on F-Logic, which is similar but not equivalent to Description Logics. For that reason, WSML is structured in levels in order to be interoperable with OWL and other formalisms, as First Order Logic. The characteristic feature of SWSL is its new rule-based language, SWSL-Rules, which allows reasoning with service descriptions. WSDL-S is independent of the representation language, and its specification states that any annotation language, such as OWL or UML, may be used.

The degree of maturity of these proposals is different and, despite the fact that none of them has been acknowledged as a standard, OWL-S is the most widely used and the one that has the most supporting tools [170].

B.6 The future of the Semantic Web

The Semantic Web, as envisioned by Berners-Lee, Hendler, and Lassila, is still a long ways from reality. Agents setting medical appointments and scheduling daily tasks are still not available, and will not be in the coming years. There are many reasons for this, and of course, one of them is the overly high expectations raised by initial research in the area.

Nevertheless, that does not mean that Semantic Web is losing popularity. On the contrary, several current initiatives are showing the potential of Semantic Web technologies and proving their usefulness. Probably, the best explanation is that besides the *great* Semantic Web (i.e. the Semantic Web of logical languages and autonomous agents), a Semantic Web with a smaller scope seems to be blooming.

The maxim of this alternative Semantic Web, often called *lowercase semantic web*, is "metadata for everything and everybody". In other words, this approach provides the means to attach metadata to Web resources that do not require users to be experts in logic-based languages. This is the objective of microformats, and, extensively, of Web 2.0.

Microformats are *ad hoc* vocabularies for entering metadata in XHTML documents. For example, the hCard microformat defines simple terms that describe contact information. Other microformats are hCalendar (for calendars and events), XFN (social networks), Rel-License (content licenses), and hReview (opinions and ratings).

Web 2.0 is a label that identifies a set of Web technologies and procedures aimed at supporting collaborative content creation [197]. Web 2.0 promotes tagging of Web resources with simple labels, the addition of metadata to documents, blogging, etc., which drives eventually to the creation of social connections between authors.

Ideally, Semantic Web users should be able to describe and link resources without making a great effort, as occurs on the Web 2.0 [8, 178]. Actually, some of the creators of the Semantic Web have remarked that this approach

combining the Web 2.0 and Semantic Web (i.e. Web 3.0) reflects their initial idea [113] quite well.

The W3C is aware of these advances and several working groups have been created to study this kind of applications. As a result, specifications such as GRDDL and RDFa, are currently being developed.

On the other hand, languages for the *uppercase* Semantic Web continue to be developed. One of the most relevant new proposals is OWL 1.1, an extension of OWL that has been submitted for the consideration of the W3C [202]. OWL 1.1 is based on the Description Logic $\mathcal{SROIQ}$, which adds new class and property constructors to $\mathcal{SHOIN}$, the logic underlying OWL.

The OWL Working Group[18] has taken OWL 1.1 as a starting point for OWL 2, which will be the successor of the current OWL specification [200]. Apart from a set of new constructors, a notable feature of the preliminary OWL 2 specification is the definition of three profiles. The objective of OWL 2 profiles is to specify segments of the language with certain computational properties in addition to the Lite, DL and Full dialects of OWL.

Thus, the EL++ profile is a subset of OWL 2 that produces representations that are less complex, in terms of interpretation by users and reasoning with inference engines. The EL++ profile is called a tractable fragment of OWL 2. DL Lite is a profile created to achieve interoperability between ontologies and databases. OWL-R, in turn, is intended to manage rules in OWL 2 representations.

Knowledge Mobilization can remain, to some extent, agnostic in respect to the progress of the Semantic Web. However, the Semantic Web is probably the most important current initiative aimed at providing universal and effective access to information, an objective which is shared by Knowledge Mobilization. Therefore, as has been suggested throughout this dissertation, suitable Semantic Web technologies can and must be incorporated into KMob applications.

[18] http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

Bibliography

[1] Agnar Aamodt and Mads Nygard. “Different roles and mutual dependencies of data, information, and knowledge – An AI perspective on their integration”. In: *Data & Knowledge Engineering* 16 (1995). Pp. 191–222.

[2] Rama Akkiraju, Joel Farrel, John Miller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth, and Kunal Verma. *Web Service Semantics - WSDL-S*. Online. W3C Member Submission. Nov. 2005. URL: http://www.w3.org/Submission/WSDL-S/.

[3] Christopher Alexander. *The timeless way of building*. Oxford University Press, 1979.

[4] David K. Allen. “Spreading the load: mobile information and communications technologies and their effect on information overload ”. In: *Information Research* 10.2 (2005).

[5] OSGi Alliance, ed. *OSGi Service Platform: The OSGi Alliance*. IOS Press, 2003.

[6] Josef Altmann, Franz Gruber, Ludwig Klug, Wolfgang Stockner, and Edgar Weippl. “Using mobile agents in real world: A survey and evaluation of agent platforms”. In: *Proceedings of the 2nd Workshop on Infrastructure for Agents, MAS, and Scalable MAS at Autonomous Agents*. Montreal, Canada 2001. Pp. 33–39.

[7] Christos B. Anagnostopoulos, Yiorgos Ntarladimas, and Stathes Hadjiefthymiades. “Situational Computing: An innovative architecture with imprecise reasoning”. In: *Journal of Systems and Software* 80.12 (2007). Pp. 1993–2014.

[8] Anupriya Ankolekar, Markus Krötzsch, Thanh Tran, and Denny Vrandecic. “The two cultures: mashing up Web 2.0 and the Semantic Web”. In: *Proceedings of the 16th International World Wide Web Conference (WWW 2007)*. Banff, Alberta, Canada 2007. Pp. 825–834.

[9] Arvind and Jamey Hicks. “A mobile phone ecosystem: MIT and Nokia’s joint research venture”. In: *IEEE Intelligent Systems* 21 (2006). Pp. 78–79.

[10] Koray Atalağ. “Archetype based domain modelling for Health Information Systems”. PhD thesis. Middle East Technical University (METU), 2007.

[11] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. “DBpedia: A Nucleus for a Web of open data”. In: *LNCS 4825. Proceedings of the 6th International Semantic Web Conference (ISWC 07)*. Busan, South Korea 2007. Pp. 722–735.

[12] Franz Baader. “Terminological cycles in a description logic with existential restrictions”. In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*. Acapulco, Mexico 2003. Pp. 325–330.

[13] Franz Baader, Sebastian Brandt, and Carsten Lutz. “Pushing the $\mathcal{EL}$ envelope”. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*. Vol. 5. Edinburgh, Scotland 2005. Pp. 364–370.

[14] Franz Baader, Ian Horrocks, and Ulrike Sattler. “Description Logics as Ontology Languages for the Semantic Web”. In: *LNCS 2605. Mechanizing Mathematical Reasoning*. 2005. Pp. 228–248.

[15] Franz Baader, Ian Horrocks, and Ulrike Sattler. “Handbook of Knowledge Representation”. In: ed. by Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. Elsevier, 2008. Chap. Description Logics, pp. 135–180.

[16] Franz Baader and Ralf Küsters. “Mathematical Problems from Applied Logic I”. In: vol. 4. International Mathematical Series. Springer New York, 2006. Chap. Nonstandard Inferences in Description Logics: The Story So Far, pp. 1–75.

[17] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel -Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[18] Steve Battle et al. *Semantic Web Services Framework (SWSF) Overview*. Online. W3C Member Submission. Sept. 2005. URL: http://www.w3.org/Submission/SWSF/.

[19] Christoph Baumer, Markus Breugst, Sang Choy, and Thomas Magedanz. “Grasshopper: a universal agent platform based on OMG MASIF and FIPA standards”. In: *Proceedings of the First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99)*. Ottawa, Canada 1999. Pp. 1–8.

[20] Sean Bechhofer, Ralf Moller, and Peter Crowther. “The DIG Description Logic Interface”. In: *CEUR-WS 81. Proceedings of the 2003 Description Logic Workshop (DL 2003)*. Rome, Italy 2003.

[21] Sean Bechhofer, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *OWL Web Ontology Language Reference*. Online. W3C Recommendation. Feb. 2004. URL: http://www.w3.org/TR/owl-ref/.

[22] Fabio Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley Publishing, 2007.

[23] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. “JADE: a FIPA2000 compliant agent development environment”. In: *AGENTS ’01: Proceedings of the 5th International Conference on Autonomous Agents*. Montreal, Quebec, Canada 2001. Pp. 216–217.

[24] Chihab BenMoussa. “Supporting the sales force through mobile Information and Communication Technologies”. PhD thesis. Abo Akademi (Turku, Finland), 2007. Pp. 74–78.

[25] Federico Bergenti and Agostino Poggi. “LEAP: A FIPA platform for handheld and mobile devices”. In: *LNCS 2333. Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL 2001)*. Seattle,Washington, USA 2002. Pp. 436–446.

[26] Tim Berners-Lee. *Relational Databases and the Semantic Web*. Online. Sept. 1998. URL: http://www.w3.org/DesignIssues/RDB-RDF.html.

[27] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific American* 284.5 (2001). Pp. 28–37.

[28] Carole Bernon, Massimo Cossentino, and Juan Pavón. “Agent-oriented software engineering”. In: *The Knowledge Engineering Review* 20 (2005). Pp. 99–116.

[29] Carole Bernon, Massimo Cossentino, and Juan Pavón. “An overview of current trends in European AOSE research”. In: *Informatica* 29.4 (2005). Pp. 379–390.

[30] Veli Bicer, Ozgur Kilic, Asuman Dogac, and Gokce B. Laleci. “Archetype-based semantic interoperability of Web Service messages in the Healthcare Domain”. In: *International Journal on Semantic Web and Information Systems* 1.4 (2005). Pp. 1–22.

[31] Phillip Bishop and Nigel Warren. *JavaSpaces in Practice*. Pearson Education, 2002.

[32] Christian Bizer and Richard Cyganiak. “D2R Server - Publishing Relational Databases on the Semantic Web”. In: *Proceedings of the 5th International Semantic Web Conference (ISWC 06)*. Poster. Athens, Georgia, USA 2006.

[33] Christian Bizer and Andy Seaborne. “D2RQ-Treating non-RDF databases as virtual RDF graphs”. In: *Proceedings of the 3rd International Semantic Web Conference (ISWC 04)*. Poster. Hiroshima, Japan 2004.

[34] Carl Blakeley. “Mapping relational data to RDF with Virtuoso’s RDF Views”. In: *OpenLink Software* (2007). Online. URL: http://www.openlinksw.com/virtuoso/Whitepapers/html/rdf_views/virtuoso_rdf_views_example.html.

[35] Luciano Barrios Blasco, José Nicolás García Rodríguez, and Francisco Pérez Torres. “La Historia Clínica Electrónica en Andalucía”. In: *Proceedings of the Seminar Innovaciones en Tecnologías de la Información en Salud*. Segovia, Spain 2002. URL: http://www.conganat.org/seis/segovia2002/barrios.htm.

[36] Eva Blomqvist. “OntoCase - A pattern-based ontology construction approach”. In: *LNCS 4803. On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*. Vilamoura, Portugal 2007. Pp. 971–988.

[37] Eva Blomqvist. “Semi-automatic ontology engineering using patterns”. In: *LNCS 4825. Proceedings of the 6th International Semantic Web Conference (ISWC 07)*. Busan, South Korea 2007. Pp. 911–915.

[38] Eva Blomqvist and Kurt Sandkuhl. “Patterns in ontology engineering – Classification of ontology patterns”. In: *Proceedings of the 7th International Conference on Enterprise Information Systems (ICEIS 2005)*. Miami, USA 2005.

[39] Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero. “A Crisp Representation for Fuzzy $\mathcal{SHOIN}$ with Fuzzy Nominals and General Concept Inclusions”. In: *CEUR-WS 218. Proceedings of the*

2nd International Workshop on Uncertainty Reasoning for the Semantic Web in the 5th International Semantic Web Conference. Athens, Georgia, USA 2006.

[40] Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero. "Crisp representations and reasoning for fuzzy Description Logics". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* (To Appear).

[41] Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero. "Optimizing the Crisp Representation of the Fuzzy Description Logic $\mathcal{SROIQ}$". In: *CEUR-WS 327. Proceedings of the 3rd International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 07) in the 6th International Semantic Web Conference*. Busan, South Korea 2007.

[42] Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero. "Representation of context-dependant knowledge in ontologies: A model and an application". In: *Expert Systems with Applications* (In Press).

[43] Fernando Bobillo, Juan Gómez-Romero, and Ramon Pérez-Pérez. "Towards Semantic Web Services: A brief overview". In: *Proceedings of the IADIS International Conference WWW/Internet 2005*. Lisbon, Portugal 2005. Pp. 19–26.

[44] Fernando Bobillo and Umberto Straccia. "fuzzyDL: An Expressive Fuzzy Description Logic Reasoner". In: *Proceedings of the 2008 International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*. Hong Kong, China (Accepted).

[45] Olivier Bodenreider. "The Unified Medical Language System (UMLS): integrating biomedical terminology". In: *Nucleic Acids Research* 32 (2004). Pp. 267–270.

[46] Piero Bonatti, Carsten Lutz, and Frank Wolter. "Expressive non-monotonic description logics based on circumscription". In: *Principles of Knowledge Representation and Reasoning: Proceedings of*

the Tenth International Conference (KR-06). Lake District, UK 2006. Pp. 400–410.

[47] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. *Web Services Architecture*. Online. W3C Working Group Note. Feb. 2004. URL: http://www.w3.org/TR/ws-arch/.

[48] Pim Borst, Hans Akkermans, and Jan Top. "Engineering ontologies". In: *International Journal of Human-Computer Studies* 46.2-3 (1997). Pp. 365–406.

[49] Genevieve Bossu and Pierre Siegel. "Saturation, nonmonotonic reasoning and the closed-world assumption". In: *Artificial Intelligence* 25 (1985). Pp. 13–63.

[50] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. "Contextualizing ontologies". In: *LNCS 2870. Proceedings of the 3rd International Semantic Web Conference (ISWC 04)*. Hiroshima, Japan 2004. Pp. 164–179.

[51] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, 2004.

[52] Ronald Brachman and Hector Levesque. "The tractability of subsumption in frame-based description languages". In: *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84)*. Austin, Texas, USA 1984. Pp. 34–37.

[53] P. Brézillon. "Context in problem solving: a survey". In: *The Knowledge Engineering Review* 14 (1999). Pp. 47–48.

[54] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. "Sesame: A generic architecture for storing and querying RDF and RDF Schema". In: *Proceedings of the 1st International Semantic Web Conference*. 2002. Pp. 54–68.

[55] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven Shafer. “EasyLiving: Technologies for Intelligent Environments”. In: *Proceedings of Handheld and Ubiquitous Computing: Second International Symposium (HUC 2000)*. Bristol, UK 2000. Pp. 12–29.

[56] Mario Bunge. *Treatise on Basic Philosophy, vol. 3. Ontology I: The Furniture of the World*. Springer, 1977.

[57] Luca Buriano, Marco Marchetti, Francesca Carmagnola, Federica Cena, Cristina Gena, and Ilaria Torre. “The role of ontologies in context-aware recommender systems”. In: *Proceedings of the 7th International Conference on Mobile Data Management (MDM 2006)*. Nara, Japan 2006. Pp. 80–81.

[58] Diego Calvanese. “Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity”. In: *Proceedings of the 12 European Conference on Artificial Intelligence (ECAI’96)*. Budapest, Hungary 1996. Pp. 303–307.

[59] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. “Handbook of Automated Reasoning”. In: ed. by Alan Robinson and Andrei Voronkov. Elsevier Science Publishers, 2001. Chap. Reasoning in Expressive Description Logics, pp. 1581–1634.

[60] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. “Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family”. In: *Journal of Automated Reasoning* 39.3 (2007). Pp. 385–429.

[61] Christer Carlsson. *Knowledge Mobilisation: Executive Summary*. Tech. rep. Institute for Advanced Management Systems Research, 2007.

[62] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. “Jena: Implementing the Semantic Web recommendations”. In: *Proceedings of the 13th International World Wide Web Conference (WWW 2004) on Alternate track papers & posters*. New York, USA 2004. Pp. 74–83.

[63] Radovan Cervenka and Ivan Trencansky. *The Agent Modeling Language - AML*. Ed. by Birkhäuser Basel. Whitestein Series in Software Agent Technologies and Autonomic Computing. 2007.

[64] B. Chandrasekaran, John R. Josephson, and V. Richard Benjamins. "What are ontologies, and why do we need them?". In: *IEEE Intelligent Systems* 14.1 (1999). Pp. 20–26.

[65] David Chappell. *Understanding .NET (2nd Edition)*. Addison-Wesley Professional, 2006.

[66] Harry Chen, Tim Finin, and Anupam Joshi. "Ontologies for Agents: Theory and Experiences". In: Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser Basel, 2005. Chap. The SOUPA Ontology for Pervasive Computing, pp. 233–258.

[67] Harry Chen, Tim Finin, Joshi Anupam, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. "Intelligent agents meet the Semantic Web in smart spaces". In: *IEEE Internet Computing* 8 (2004). Pp. 69–79.

[68] David M. Chess, Colin G. Harrison, and Aaron Kershenbaum. "Mobile agents: are they a good idea?". In: *MOS '96: Selected Presentations and Invited Papers Second International Workshop on Mobile Object Systems - Towards the Programmable Internet*. Linz, Austria 1997. Pp. 25–45.

[69] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. *Documenting software architectures: Views and beyond*. Addison-Wesley Professional, 2002.

[70] College of American Pathologists. *SNOMED Clinical Terms User Guide*. Online. IHTSDO standard. Jan. 2007. URL: http://www.ihtsdo.org/fileadmin/user_upload/Docs_01/Technical_Docs/snomed_ct_user_guide.pdf.

[71] Oscar Corcho, Mariano Fernández-López, and Asuncion Gómez-Pérez. "Methodologies, tools and languages for building ontologies. Where is their meeting point?". In: *Data & Knowledge Engineering* 46 (2003). Pp. 41–64.

[72] Dave Crane, Eric Pascarello, and Darren James. *Ajax in action*. Manning, 2006.

[73] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. "A logical framework for modularity of ontologies". In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*. Hyderabad, India 2007. Pp. 298–304.

[74] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. "Modular Reuse of Ontologies: Theory and Practice". In: *Journal of Artificial Intelligence Research* 31 (2008). Pp. 273–318.

[75] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI". In: *IEEE Internet Computing* 6.2 (2002). Pp. 86–93.

[76] J. Davies, R. Studer, and P. Warren. *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley & Sons, 2006.

[77] Miguel Delgado, Juan Gómez-Romero, Pedro J. Magaña, and Ramon Pérez-Pérez. "A flexible architecture for distributed knowledge based systems with nomadic access through handheld devices". In: *Expert Systems with Applications* 29.4 (2005). Pp. 965–975.

[78] Michael L. Dertouzos. "The future of computing". In: *Scientific American* 281.2 (1999). Pp. 36–47.

[79] A.K. Dey and G.D. Abowd. "Towards a Better Understanding of Context and Context-Awareness". In: *Proceedings of the Workshop on the What, Who, Where, When, and How of Context-Awareness (CHI 2000)*. The Hague, Netherlands 2000.

[80] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications". In: *Human-Computer Interaction* 16.2-4 (2001). Pp. 97–166.

[81] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. “Principles of Knowledge Representation”. In: CSLI Studies In Logic, Language And Information. 1996. Chap. Reasoning in Description Logics, pp. 191–236.

[82] Raymond G. Duncan and Michael M. Shabot. “Secure remote access to a clinical data repository using a wireless personal digital assistant (PDA)”. In: *Proceedings of the AMIA Annual Symposium (AMIA 2000)*. Vol. 155. Los Angeles, CA, USA 2000. Pp. 210–214.

[83] Angela Edmunds and Anne Morris. “The problem of information overload in business organisations: a review of the literature”. In: *International Journal of Information Management* 20 (2000). Pp. 17–28.

[84] Marco Eichelberg, Thomas Aden, Jörg Riesmeier, Asuman Dogac, and Gokce B. Laleci. “A survey and analysis of Electronic Healthcare Record standards”. In: *ACM Computing Surveys* 37.4 (2005). Pp. 277–315.

[85] Martin J. Eppler and Jeanne Mengis. “The Concept of Information Overload: A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines”. In: *The Information Society* 20 (2004). Pp. 325–344.

[86] Hans-Erik Eriksson, Magnus Penker, Brian Lyons, and David Fado. *UML 2 Toolkit*. Wiley Publishing, 2004.

[87] Ali F. Farhoomand and Don H. Drury. “Managerial information overload ”. In: *Communications of the ACM* 45 (2002). Pp. 127–131.

[88] Cristina Feier. *Web Service Modeling Ontology Primer*. Online. W3C Member Submission. June 2005. URL: http://www.w3.org/Submission/WSMO-primer/.

[89] Edward A. Feigenbaum, Pamela McCorduck, and H. Penny Nii. *The rise of the expert company*. Times Books New York, NY, USA, 1988.

[90] Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann, and Susie Stephens. “The Semantic Web in Action”. In: *Scientific American* 297 (2007). Pp. 90–97.

[91] Jacques Ferber. *Multi-agent Systems: An Introduction to Distributed Artificial Intelligence*. Harlow: Addison-Wesley Longman, 1999.

[92] Richard Fikes and Tom Kehler. “The role of frame-based representation in reasoning”. In: *Communications of the ACM* 28.9 (1985). Pp. 904–920.

[93] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. “KQML as an agent communication language”. In: *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM94)*. 1994. Pp. 456–463.

[94] FIPA. *FIPA Specifications*. Online. Accesed: May 2008. URL: http://www.fipa.org/specifications/index.html.

[95] Giorgios Flouris, Dimitris Plexousakis, and Grigoris Antoniou. “On applying the AGM theory to DLs and OWL”. In: *LNCS 3729. Proceedings of the 4th International Semantic Web Conference (ISWC 05)*. Galway, Ireland 2005. Pp. 216–231.

[96] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Professional, 1994. P. 416.

[97] Aldo Gangemi. “Ontology design patterns for Semantic Web content”. In: *LNCS 3729. Proceedings of the 4th International Semantic Web Conference (ISWC 05)*. Galway, Ireland 2005. Pp. 262–276.

[98] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste. “Project Aura: Toward distraction-free Pervasive Computing”. In: *IEEE Pervasive Computing* 1.2 (2002). Pp. 22–31.

[99] David Gelernter. “Generative communication in Linda”. In: *ACM Transactions on Programming Languages and Systems* 7.1 (1985). Pp. 80–112.

[100] Michael R. Genesereth and Richard Fikes. *Knowledge Interchange Format, Version 3.0. Reference Manual*. Online. Knowledge Systems Laboratory, University of Stanford. 1992. URL: http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps.

[101] Silvio Ghilardi, Carsten Lutz, and Frank Wolter. "Did I damage my ontology? A case for conservative extensions in Description Logics". In: *Proceeding of the of 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*. Lake District, UK 2006. Pp. 187–197.

[102] Peter G. Goldschmidt. "HIT and MIS: Implications of health information technology and medical information systems". In: *Communications of the ACM* 48.10 (2005). Pp. 68–74.

[103] Asuncion Góomez-Pérez, Oscar Corcho, and Mariano Fernández-López. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, 2004.

[104] Ian Gorton. "Essential Software Architecture". In: ed. by Ian Gorton. Springer, 2006. Chap. Understanding Software Architecture, pp. 1–15.

[105] Thomas R. Gruber. "A translation approach to portable ontology specifications". In: *Knowledge Acquisition* 5.2 (1993). Pp. 199–220.

[106] Tao Gu, Hung Keng Pung, and Da Qing Zhang. "A service-oriented middleware for building context-aware services". In: *Journal of Network and Computer Applications* 28.1 (2005). Pp. 1–18.

[107] Ramanathan Guha, Rob McCool, and Richard Fikes. "Contexts for the Semantic Web". In: *LNCS 3298. Proceedings of the 3rd International Semantic Web Conference (ISWC 04)*. Hiroshima, Japan 2004. Pp. 32–46.

[108] Volker Haarslev and Ralf Moller. “Description of the RACER System and its Applications”. In: *CEUR-WS 49. Proceedings of the International Workshop on Description Logics (DL-2001)*. Stanford University, California, USA 2001.

[109] Peter Haase, Pascal Hitzler, Sebastian Rudolph, and Guilin Qi. *Formalisms for context sensitivity (state-of-the-art review)*. Tech. rep. D.3.1.1. Institute AIFB, University of Karlsruhe, 2006. URL: http://www.neon-project.org/web-content/index.php?option=com_weblinks&task=view&catid=17&id=30.

[110] E.S. Hall, D.K. Vawdrey, C.D. Knutson, and J.K. Archibald. “Enabling remote access to personal electronic medical records”. In: *IEEE Engineering in Medicine and Biology Magazine* 22.3 (2003). Pp. 133–139.

[111] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. “The anatomy of a context-aware application”. In: *Wireless Networks* 8.2 (2002). Pp. 187–197.

[112] James Hendler. “Agents and the Semantic Web”. In: *IEEE Intelligent Systems* 16.2 (2001). Pp. 30–37.

[113] James Hendler. *Shirkyng my responsibility*. Online. Nov. 2007. URL: http://www.mindswap.org/blog/2007/11/21/shirkyng-my-responsibility/.

[114] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. “Design Science in Information Systems Research”. In: *MIS Quarterly* 28.1 (2004). Pp. 75–105.

[115] Jason I. Hong and James A. Landay. “An infrastructure approach to context-aware computing”. In: *Human-Computer Interaction* 16.2-5 (2001). Pp. 287–303.

[116] Matthew Horridge, Sean Bechhofer, and Olaf Noppens. “Igniting the OWL 1.1 Touch Paper: The OWL API”. In: *CEUR-WS 258. Proceedings of the OWLED’07 Workshop on OWL: Experiences and Directions*. Innsbruck, Austria 2007.

[117] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai H. Wang. “The Manchester OWL Syntax”. In: *CEUR-WS 216. Proceedings of the OWLED’06 Workshop on OWL: Experiences and Directions*. Athens, Georgia, USA 2006.

[118] Ian Horrocks. “Using an Expressive Description Logic: FaCT or Fiction?”. In: *Procedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR’98)*. Trento, Italy 1998. Pp. 636–645.

[119] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. “The even more irresistible $\mathcal{SROIQ}$”. In: *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*. Lake District, UK 2006. Pp. 452–457.

[120] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. “From $\mathcal{SHIQ}$ and RDF to OWL: the making of a Web Ontology Language”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 1.1 (2003). Pp. 7–26.

[121] Ian Horrocks and Peter Patel-Schneider. “Reducing OWL entailment to description logic satisfiability”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 1.4 (2004). Pp. 345–357.

[122] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. “Practical reasoning for expressive Description Logics”. In: *LNCS 1705. Logic for Programming and Automated Reasoning*. 1999. Pp. 161–180.

[123] Wen-Chen Hu, Chung-Wei Lee, and Weidong Kou. *Advances in security and payment methods for mobile commerce*. Idea Group Publishing, 2004.

[124] IEEE. *IEEE Standards Description: 1471-2000*. Online. 2000. URL: http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html.

[125] Juhani Iivari. “A paradigmatic analysis of contemporary schools of IS development”. In: *European Journal of Information Systems* 1.4 (1991). Pp. 249–272.

[126] Juhani Iivari. “Information Systems development: Advances in theory, practice, and education”. In: ed. by Olegas Vasilecas, Wita Wojtkowski, Jože Zupančič, Albertas Caplinskas, W. Gregory Wojtkowski, and Stanislaw Wrycza. Springer, 2005. Chap. Information Systems as a Design Science. Advances in theory, practice, and education.

[127] Juhani Iivari. “Towards Information Systems as a science of meta-artifacts”. In: *Communications of the Association for Information Systems* 12 (2003). Pp. 568–581.

[128] Informa Telecoms and Media. *Mobile Converged Devices: Enabling IMS, SIP, UMA & VCC servicesWorldwide market Analysis, Strategic Outlook & Forecasts to 2012*. Tech. rep. 2007. URL: http://www.the-infoshop.com/pdf/itm55508.pdf.

[129] Heecheol Jeon, Charles Petrie, and Mark R. Cutkosky. “JATLite: A Java Agent Infrastructure with Message Routing”. In: *IEEE Internet Computing* 4.2 (2000). Pp. 87–96.

[130] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ulrike Sattler, Thomas Schneider, and Rafael Berlanga. “Safe and economic re-use of ontologies: A logic-Based methodology and tool support”. In: *LNCS 5021. Proceedings of the 5th European Semantic Web Conference (ESWC 08)*. Tenerife, Spain 2008. Pp. 185–199.

[131] Jin Jing, Abdelsalam Sumi Helal, and Ahmed Elmagarmid. “Client-server computing in mobile environments”. In: *ACM Computing Surveys* 31.2 (1999). Pp. 117–157.

[132] Brad Johanson, Armando Fox, and Terry Winograd. “The Interactive Workspaces project: experiences with ubiquitous computing rooms”. In: *Pervasive Computing, IEEE* 1.2 (2002). Pp. 67–74.

[133] Rod Johnson. *Expert one-on-one J2EE design and development*. Wrox, 2002. ISBN: 0764543857.

[134] Diane Jordan and John Evdemon. *Web Services Business Process Execution Language Version 2.0*. Online. Oasis Standard. Apr. 2007. URL: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[135] Eero Kasanen, Kari Lukka, and Arto Siitonen. "The constructive approach in Management Accounting research". In: *Journal of Management Accounting Research* 5 (1993). Pp. 243–264.

[136] Yarden Katz and Bijan Parsia. "Towards a Nonmonotonic Extension to OWL". In: *CEUR-WS 188. Proceedings of the OWLED'05 Workshop on OWL: Experiences and Directions*. Galway, Ireland 2005.

[137] Nickolas Kavantzas, David Burdett, Gregory Ritzinger, Tony Fletcher, Yves Lafon, and Charlton Barreto. *Web Services Choreography Description Language Version 1.0*. Online. W3C Candidate Recommendation. Nov. 2005. URL: http://www.w3.org/TR/ws-cdl-10/.

[138] Peter G. Keen and Ron Mackintosh. *The freedom economy*. McGraw-Hill Professional, 2001.

[139] Mohamed Khedr and Ahmed Karmouch. "ACAI: agent-based context-aware infrastructure for spontaneous applications". In: *Journal of Network and Computer Applications* 28.1 (2005). Pp. 19–44.

[140] Deepali Khushraj, Ora Lassila, and Tim Finin. "sTuples: semantic tuple spaces". In: *Proceedings of Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004*. Boston, Massachusetts, USA 2004. Pp. 268–277.

[141] Michael Kifer. "Rules and Ontologies in F-Logic". In: *LNCS 3564. Proceedings of Reasoning Web: First International Summer School*. Msida, Malta 2005. Pp. 22–34.

[142] Michael Kifer, Georg Lausen, and James Wu. "Logical Foundations of Object-Oriented and Frame-Based Languages". In: *Journal of the ACM* 42 (1995). Pp. 741–843.

[143] Ozgur Kilic, Veli Bicer, and Asuman Dogac. *Mapping Archetypes to OWL*. Online. 2005. URL: http://www.srdc.metu.edu.tr/webpage/publications/2005/MappingArchetypestoOWLTechnical.pdf.

[144] Eunhoe Kim and Jaeyoung Choi. "A context-awareness middleware based on Service-Oriented Architecture". In: *LNCS 4611. Proceedings of the Ubiquitous Intelligence and Computing Conference (UIC-07)*. Hong Kong, China 2007. Pp. 953–962.

[145] Tim Kindberg and John Barton. "A Web-based nomadic computing system". In: *Computer Networks* 35 (2001). Pp. 443–456.

[146] George J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Prentice Hall, 1995.

[147] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. "The Protégé OWL plugin: An open development environment for Semantic Web applications". In: *LNCS 3298. Proceedings of the 3rd International Semantic Web Conference (ISWC 04)*. Hiroshima, Japan 2004. Pp. 229–243.

[148] Vladimir Kolovski, Bijan Parsia, and Yarden Katz. "Implementing OWL Defaults ". In: *CEUR-WS 216. Proceedings of the OWLED'06 Workshop on OWL: Experiences and Directions*. Athens, Georgia, USA 2006.

[149] Panu Korpipää, Jani Mäntyjärvi, Juha Kela, Heikki Keränen, and Eski-Juhani Malm. "Managing context information in mobile devices". In: *Pervasive Computing, IEEE* 2 (2003). Pp. 42–51.

[150] Michal Kostic. "Code Generation from AML". PhD thesis. Univerzity Komenského, Bratislava, 2006.

[151] Georg von Krogh, Kazuo Ichijo, and Ikujiro Nonaka. *Enabling knowledge creation: How to unlock the mystery of tacit knowledge and release the power of innovation*. Oxford University Press, 2000.

[152] Reto Krummenacher, Francisco J. Martin-Recuerda, Martin Murth, and Johannes Riemer. *TSC Deliverable D1.2 TSC Framework*. Online. Project Deliverable. July 2006. URL: http://tsc.deri.at/deliverables/D12v11.html.

[153] Yu-Kwong Ricky Kwok and Vincent K.N. Lau. *Wireless Internet and Mobile Computing: Interoperability and Performance*. Wiley-IEEE Press, 2007.

[154] Ohbyung Kwon, Sungchul Choi, and Gyuro Park. "NAMA: a context-aware multi-agent based web service approach to proactive need identification for personalized reminder systems". In: *Expert Systems with Applications* 29.1 (2005). Pp. 17–32.

[155] Cristian Pérez de Laborda Schwankhart. "Incorporating Relational Data into the Semantic Web". PhD thesis. Heinrich-Heine-Universität Düsseldorf, 2006.

[156] Jennifer Lai, Anthony Levas, Paul Chou, Claudio Pinhanez, and Marisa Viveros. "BlueSpace: personalizing workspace through awareness and adaptability". In: *International Journal of Human-Computer Studies* 57 (2002). Pp. 415–428.

[157] Ora Lassila. "Using the Semantic Web in Mobile and Ubiquitous Computing". In: *Proceedings of the IFIP International Federation for Information Processing : Industrial Applications of Semantic Web*. 2005. Pp. 19–25.

[158] Ora Lassila and Deepali Khushraj. "Contextualizing Applications via Semantic Middleware". In: *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 05)*. San Diego, California, USA 2005.

[159] Valentino Lee, Heather Schneider, and Robbie Schell. *Mobile applications: Architecture, design, and development*. Prentice Hall, 2004.

[160] Fritz Lehmann. "Semantic networks". In: *Computers & Mathematics with Applications* 23 (1992). Pp. 1–50.

[161] Hector Levesque and Ronald Brachman. “Expressiveness and tractability in knowledge representation and reasoning”. In: *Computational intelligence* 3.1 (1987). Pp. 78–93.

[162] Hector Levesque, Fiora Pirri, and Ray Reiter. “Foundations for the situation calculus”. In: *Linköping Electronic Articles in Computer and Information Science* 3.18 (1998).

[163] Jinshan Liu, Daniele Sacchetti, Françoise Sailhan, and Valérie Issarny. “Group management for mobile ad hoc networks: design, implementation and experiment”. In: *Proceedings of the 6th International Conference on Mobile Data Management*. Ayia Napa, Cyprus 2005. Pp. 192–199.

[164] Steve Love. *Understanding mobile human-computer interaction*. Information Systems Series. Butterworth-Heinemann, 2005.

[165] Thomas Lukasiewicz and Umberto Straccia. *An Overview of Uncertainty and Vagueness in Description Logics for The Semantic Web*. Tech. rep. Institut für Informationssysteme, Technische Universität Wien, Austria, 2006.

[166] Alexander Maedche and Steffen Staab. “Measuring similarity between ontologies”. In: *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*. Sigüenza, Spain 2002. Pp. 251–263.

[167] Frank Manola and Eric Miller. *RDF Primer*. Online. W3C Recommendation. Feb. 2004. URL: http://www.w3.org/TR/rdf-primer/.

[168] Jani Mäntyjärvi and Tapio Seppänen. “Adapting applications in handheld devices using fuzzy context information”. In: *Interacting with Computers* 15.4 (2003). Pp. 521–538.

[169] Salvatore T. March and Gerald F. Smith. “Design and natural science research on information technology”. In: *Decision Support Systems* 15.4 (1995). Pp. 251–266.

[170] David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massimo Paolucci, Katia Sycara, Deborah L. McGuinness, and Even Sirin. “Bringing Semantics to Web Services with OWL-S”. In: *World Wide Web* 10.3 (2007). Pp. 243–277.

[171] David Martin et al. *OWL-S: Semantic Markup for Web Services*. Online. W3C Member Submission. Nov. 2004. URL: http://www.w3.org/Submission/OWL-S/.

[172] Ana Mas, ed. *Agentes software y Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones*. Pearson-Prentice Hall, 2005.

[173] Ryusuke Masuoka, Yannis Labrou, Bijan Parsia, and Even Sirin. “Ontology-enabled Pervasive Computing applications”. In: *IEEE Intelligent Systems* 18.5 (2003). Pp. 68–72.

[174] Neil Mawston. *Nokia reaches 40% share as 332 million cellphones ship worldwide in Q4 2007*. Tech. rep. Strategy Analytics, 2008.

[175] Brian McBride. *RDF Vocabulary Description Language 1.0: RDF Schema*. Online. W3C Recommendation. Feb. 2004. URL: http://www.w3.org/TR/rdf-schema/.

[176] D.L McGuiness and F. van Harmelen. *OWL Web Ontology Language Overview*. Online. W3C Recommendation. Feb. 2004. URL: http://www.w3.org/TR/owl-features/.

[177] Sheila A. McIlraith, Tran C. Son, and Honglei Zeng. “Semantic Web services”. In: *Intelligent Systems* 16.2 (2001). Pp. 46–53.

[178] Alexander Mikroyannidis. “Toward a Social Semantic Web”. In: *Computer* 40.11 (2007). Pp. 113–115.

[179] Dejan Milojicic et al. “MASIF: The OMG mobile agent system interoperability facility”. In: *Personal and Ubiquitous Computing* 2.2 (1998). Pp. 117–129.

[180] Jack Minker. “An overview of nonmonotonic reasoning and logic programming”. In: *The Journal of Logic Programming* 17 (1993). Pp. 95–126.

[181] Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler. “Can OWL and Logic Programming live together happily ever after?”. In: *LNCS 4273. Proceedings of the 5th International Semantic Web Conference (ISWC 06)*. Athens, Georgia, USA 2006. Pp. 501–514.

[182] Bernd Mrohs, Marko Luther, Raju Vaidya, Matthias Wagner, Stephan Steglich, Wolfgang Kellerer, and Stefan Arbanowski. “OWL-SF - A distributed semantic service framework”. In: *Proceedings of the Workshop on Context Awareness for Proactive Systems (CAPS05)*. Helsinki 2005. Pp. 67–77.

[183] Joel Murach and Andrea Steelman. *Murach’s Java Servlets and JSP (2nd Edition)*. Mike Murach & Associates, 2008.

[184] Amy L. Murphy, Gian Pietro Picco, and Gruia-Catalin Roman. “Software architecture for Mobile Computing”. In: *Proceedings of the 3rd International School on Formal Methods for the Design of Computer, Communication and Software Systems: Software Architectures (SFM 2003)*. Bertinoro, Italy 2003. Pp. 182–206.

[185] National Institutes of Health. National Center for Research Resources. *Electronic Health Records In Academic Medical Centers*. Online. 2006. URL: http://www.ncrr.nih.gov/publications/informatics/EHR.pdf.

[186] Eric Newcomer. *Understanding Web Services: XML, WSDL, SOAP and UDDI*. Addison-Wesley Professional, 2002.

[187] Eric W. T. Ngai and Angappa Gunasekaran. “A review for mobile commerce research and applications”. In: *Decision Support Systems* 43 (2007). Pp. 3–15.

[188] Ilkka Niiniluoto. “The aim and structure of applied research”. In: *Erkenntnis* 38.1 (1993). Pp. 1–21.

[189] Lyndon Nixon, Olena Antonechko, and Robert Tolksdorf. “Towards semantic tuplespace computing: the Semantic Seb Spaces system”. In: *Proceedings of the 2007 ACM symposium on Applied Computing*. Seoul, Korea 2007. Pp. 360–365.

[190] Natalya F. Noy, Ray W. Fergerson, and Mark A. Musen. “The knowledge model of Protege-2000: Combining interoperability and flexibility”. In: *LNCS 1937. Proceedings of the 2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*. Juan-les-Pins, France 2000. Pp. 69–82.

[191] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A guide to creating your first Ontology*. Online. (Written). Knowledge Systems Laboratory. 2001. URL: http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html.

[192] Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubezy, Ray W. Fergerson, and Mark A. Musen. “Creating Semantic Web contents with Protege-2000”. In: *IEEE Intelligent Systems* 2 (2001). Pp. 60–71.

[193] Jay F. Nunamaker, Minder Chen, and Titus D.M. Purdin. “Systems development in Information Systems research”. In: *Journal of Management Information Systems* 7.3 (1991). Pp. 99–106.

[194] Tero Ojanperä. “Convergence Transforms Internet”. In: *Wireless Personal Communications* 37 (2006). Pp. 167–185.

[195] *Ontology Engineering and Patterns Task Force*. Online. Feb. 2008. URL: http://www.w3.org/2001/sw/BestPractices/OEP/.

[196] Charles Oppenheim. “Managers' use and handling of information”. In: *International Journal of Information Management* 17 (1997). Pp. 239–248.

[197] Tim O'Reilly. *What Is Web 2.0*. Online. Sept. 2005. URL: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[198] Amir Padovitz, Seng W. Loke, and Arkady Zaslavsky. “The ECORA framework: A hybrid architecture for context-oriented pervasive computing”. In: *Pervasive and Mobile Computing* (In Press).

[199] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1993.

[200] Bijan Parsia and Peter F. Patel-Schneider. *OWL 2 Web Ontology Language primer*. Online. W3C Working Draft. Apr. 2008. URL: http://www.w3.org/TR/owl2-primer/.

[201] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. Online. W3C Recommendation. Feb. 2004. URL: http://www.w3.org/TR/owl-semantics/.

[202] Peter F. Patel-Schneider, Ian Horrocks, and Bernardo Cuenca Grau. *OWL 1.1 Web Ontology Language Overview*. Online. W3C Member Submission. Dec. 2006. URL: http://www.w3.org/Submission/2006/SUBM-owl11-overview-20061219/.

[203] Terry R. Payne. "Web Services from an Agent Perspective". In: *IEEE Intelligent Systems* 23.2 (2008). Pp. 12–14.

[204] Alex P. Pentland. "Smart rooms". In: *Scientific American* 274 (1996). Pp. 54–62.

[205] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. "Jadex: Implementing a BDI-Infrastructure for JADE Agents". In: *EXP – in search of innovation* 3 (2003). Pp. 76–85.

[206] Shankar R. Ponnekanti, Brad Johanson, Emre Kiciman, and Armando Fox. "Portability, Extensibility and Robustness in iROS". In: *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003)*. Dallas, Texas, USA 2003. P. 11.

[207] Ian Poole. *Cellular Communications Explained: From Basics to 3G*. Newnes, 2006.

[208] Stefan Poslad, Phil Buckle, and Rob Hadingham. "The FIPA-OS agent platform: Open Source for Open Standards". In: *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents* (2000). Pp. 355–368.

[209] Eric Prud'hommeaux. *SPASQL: SPARQL Support In MySQL*. Online. 2005. URL: http://www.w3.org/2005/05/22-SPARQL-MySQL/XTech.

[210] Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*. Online. W3C Recommendation. Jan. 2008. URL: http://www.w3.org/TR/rdf-sparql-query/.

[211] Gilin Qi. *Context representation formalism*. Tech. rep. D.3.1.2. Universität Karlsruhe (TH), 2007.

[212] He Qiu-sheng and Tu Shi-liang. "A lightweight architecture to support context-aware ubiquitous agent system". In: *LNCS 4088. Agent Computing and Multi-Agent Systems: 9th Pacific Rim International Workshop on Multi-Agents*. Guilin, China 2006. Pp. 696–701.

[213] M. Ross Quillian. "Word concepts: A theory and simulation of some basic semantic capabilities". In: *Behavioral Science* 12.5 (1967). Pp. 410–430.

[214] Carlos Ramos, Juan Carlos Augusto, and Daniel Shapiro. "Ambient Intelligence – The next step for Artificial Intelligence". In: *IEEE Intelligent Systems* 23 (2008). Pp. 15–18.

[215] Anand Ranganathan, Jalal Al-Muhtadi, and Roy H. Campbell. "Reasoning about uncertain contexts in pervasive computing environments". In: *IEEE Pervasive Computing* 3.2 (2004). Pp. 62–70.

[216] Anand Ranganathan and Roy H. Campbell. "A middleware for context-aware agents in Ubiquitous Computing environments". In: *Proceedings of ACM/IFIP/USENIX International Middleware Conference*. Rio de Janeiro, Brazil 2003. Pp. 16–20.

[217] Anand Ranganathan, Robert E. McGrath, Roy H. Campbell, and M. Dennis Mickunas. "Use of ontologies in a pervasive computing environment". In: *Knowledge Engineering Review* 18.3 (2003). Pp. 209–220.

[218] Alan Rector and Jeremy Rogers. “Ontological and practical issues in using a Description Logic to represent medical concept systems: Experience from GALEN”. In: *LNCS 4126. Reasoning Web. Second International Summer School, Tutorial Lectures*. Lisbon, Portugal 2006. Pp. 197–231.

[219] Jacqueline R. Reich. “Ontological design patterns: Metadata of molecular biological ontologies, information and knowledge”. In: *Proceedings of the 11th International Conference on Database and Expert Systems Applications (DEXA 2000)*. London, UK 2000. Pp. 698–709.

[220] Bradley J. Rhodes, Nelson Minar, and Josh Weaver. “Wearable Computing meets Ubiquitous Computing: Reaping the best of both worlds”. In: *Third International Symposium on Wearable Computers (ISWC’99). Digest of Papers*. San Francisco, California, USA 1999. Pp. 141–149.

[221] Jeremy Rogers, Angus Roberts, Danny Solomon, Egbert van der Haring, Christopher Wroe, Pieter Zanstra, and Alan Rector. “GALEN ten years on: tasks and supporting tools”. In: *Proceedings of Congress of the International Medical Informatics Association (MEDINFO2001)*. London, UK 2001. Pp. 256–260.

[222] Isabel Román. *Ontology for EHR*. Online. (Accessed). 2008. URL: http://trajano.us.es/~isabel/EHR/.

[223] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, 1995.

[224] Ivo Salmre. *Writing mobile Code: Essential software engineering for building mobile applications*. Addison-Wesley Professional, 2005.

[225] Ellie Sanchez, ed. *Fuzzy Logic and the Semantic Web*. Elsevier Science, 2006.

[226] Mahadev Satyanarayanan. “Pervasive computing: vision and challenges”. In: *IEEE Personal Communications* 8.4 (2001). Pp. 10–17.

[227] Andrea Schaerf. “Reasoning with individuals in concept languages”. In: *Data Knowledge and Engineering* 13.2 (1994). Pp. 141–176.

[228] Manfred Schmidt-Schauss and Gert Smolka. “Attributive concept descriptions with complements”. In: *Artificial Intelligence* 48.1 (1991). Pp. 1–26.

[229] Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. “There is more to context than location”. In: *Computers & Graphics* 23.6 (1999). Pp. 893–901.

[230] Nigel Shadbolt, Wendy Hall, and Tim Berners-Lee. “The semantic Web revisited”. In: *Intelligent Systems* 21.3 (2006). Pp. 96–101.

[231] David Shenk. *Data Smog: Surviving the information glut*. HarperCollins Publishers, 1997. P. 250.

[232] Clay Shirky. *The Semantic Web, Syllogism, and Worldview*. Online. Nov. 2003. URL: http://www.shirky.com/writings/semantic_syllogism.html.

[233] Abhishek Singh and Michael Conway. *Survey of context aware frameworks: Analysis and criticism*. Online. 2006. URL: http://its.unc.edu/teap/tap/core/caf_review.pdf.

[234] Even Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. “Pellet: A practical OWL-DL reasoner”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 5.2 (2007). Pp. 51–53.

[235] John Soldatos, Ippokratis Pandis, Kostas Stamatis, Lazaros Polymenakos, and James L. Crowley. “Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services”. In: *Computer Communications* 30.3 (2007). Pp. 577–591.

[236] John Soldatos, Kostas Stamatis, Siamak Azodolmolky, Ippokratis Pandis, and Lazaros Polymenakos. “Semantic Web technologies for Ubiquitous Computing resource management in smart spaces”. In: *International Journal of Web Engineering and Technology* 3 (2007). Pp. 353–373.

[237] John F. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley, 1984.

[238] John F. Sowa. *Knowledge Representation. Logical, Philosophical and Computational Foundations*. Brooks/Cole, 2000.

[239] Steffen Staab, Michael Erdmann, and Alexander Maedche. "Engineering ontologies using semantic patterns". In: *Proceedings of the IJCAI-01 Workshop on E-Business & the Intelligent Web*. Seattle, USA 2001. Pp. 174–185.

[240] Andrew J. Stanley and Philip S. Clipsham. "Information overload: myth or reality?". In: *Proceedings of the IEEE Colloquium on IT Strategies for Information Overload*. London, UK 1997. Pp. 1–4.

[241] Damian Steer. *SquirrelRDF*. Online. (Accessed). 2008. URL: http://jena.sourceforge.net/SquirrelRDF/.

[242] Giorgos Stoilos, Giorgos Stamou, Jeff Z. Pan, V Tzouvaras, and Ian Horrocks. "Reasoning with very expressive fuzzy Description Logics". In: *Journal of Artificial Intelligence Research* 30 (2007). Pp. 273–320.

[243] Giorgos Stoilos, Nikos Simou, Giorgos Stamou, and Stefanos Kollias. "Uncertainty and the Semantic Web". In: *IEEE Intelligent Systems* 21 (2006). Pp. 84–87.

[244] Peter Stone and Manuela Veloso. "Multiagent Systems: A survey from a Machine Learning perspective". In: *Autonomous Robots* 8.3 (2000). Pp. 345–383.

[245] Umberto Straccia. "Fuzzy Logic and the Semantic Web". In: ed. by Elie Sanchez. Vol. 1. Capturing Intelligence. Elsevier Science, 2006. Chap. Uncertainty and Description Logic programs over lattices, pp. 115–133.

[246] Umberto Straccia. "Reasoning within fuzzy Description Logics". In: *Journal of Artificial Intelligence Research* 14 (2001). Pp. 137–166.

[247] Umberto Straccia. “Transforming fuzzy Description Logics into classical Description Logics”. In: *LNCS 3229. Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-04)*. Lisbon, Portugal 2004. Pp. 385–399.

[248] Thomas Strang and Claudia Linnhoff-Popien. “A context modeling survey”. In: *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*. 2004.

[249] H. Stuckenschmidt. “Toward multi-viewpoint reasoning with OWL ontologies”. In: *LNCS 4011. Proceedings of the 3rd European Semantic Web Conference (ESWC 06)*. Budva, Montenegro 2006. Pp. 259–272.

[250] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. “Knowledge engineering: principles and methods”. In: *Data Knowledge Engineering* 25.1-2 (1998). Pp. 161–197.

[251] Gerhard Sutschet. *The CHIL reference model architecture for multimodal perceptual Systems*. Online. 2007. URL: http://chil.server.de/servlet/is/6503/.

[252] Vojtěch Svátek. “Design patterns for Semantic Web ontologies: Motivation and discussion”. In: *Proceedings of the 7th Conference on Business Information Systems (BIS2004)*. Poznaň, Poland 2004.

[253] Simon G. Thompson and Behnam Azvine. “No Pervasive Computing without Intelligent Systems”. In: *BT Technology Journal* 22.3 (2004). Pp. 39–49.

[254] Stephan Tobies. “Complexity Results and Practical Algorithms for Logics in Knowledge Representation”. PhD thesis. RWTH Aachen, 2001.

[255] Robert Tolksdorf, Lyndon Nixon, Elena Paslaru Bontas, Duc Minh Nguyen, and Franziska Liebsch. “Enabling real world Semantic Web applications through a coordination middleware”. In: *Proceedings of the 2nd European Semantic Web Conference*. Heraklion, Greece 2005. Pp. 679–693.

[256] Ivan Trencansky and Radovan Cervenka. “Agent Modelling Language (AML): A comprehensive approach to modelling MAS”. In: *Informatica* 29.4 (2005). Pp. 391–400.

[257] Paul Warren. “From Ubiquitous Computing to Ubiquitous Intelligence”. In: *BT Technology Journal* 22.2 (2004). Pp. 28–38.

[258] Mark Weiser. “The computer for the 21st century”. In: *Scientific American* 265.3 (1991). Pp. 94–104.

[259] Gerhard Weiss, ed. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, 1999. P. 619.

[260] Wayne Wolf. *Computers as components: Principles of Embedded Computing system design*. Morgan Kaufmann, 2001.

[261] Stephen S. Yau, Fariaz Karim, Yu Wang, Bin Wang, and Sandeep K.S. Gupta. “Reconfigurable context-sensitive middleware for Pervasive Computing”. In: *IEEE Pervasive Computing* 01.3 (2002). Pp. 33–40.

[262] Giulio Zambon and Michael Sekler. *Beginning JSP, JSF and Tomcat Web Development: From Novice to Professional*. Apress, 2007.

[263] Weishan Zhang, Thomas Kunz, and Klaus Marius. “Product Line Enabled Intelligent Mobile Middleware”. In: *Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems*. Auckland, New Zealand 2007. Pp. 148–160.

[264] Evgeny Zolin. *Description Logic Complexity Navigator*. Online. (Accessed). Apr. 2008. URL: http://www.cs.man.ac.uk/~ezolin/dl/.