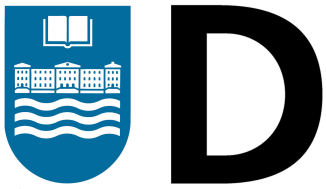


Informe científico final

ISMED: Intelligent Semantic Middleware for Embedded Devices

Universidad de Deusto
Mondragon Goi Eskola Politeknikoa

Bilbao-Mondragón, diciembre de 2010

Índice general

1. Introducción **1**

2. Estado del Arte **3**
2.1. Modelado y coordinación semántica 3
2.2. Aprendizaje 49
2.3. Composición de servicios 60
2.4. Descubrimiento de recursos 74
2.5. Razonamiento 96

3. Diseño **113**
3.1. Modelado y coordinación semántica 113
3.2. Aprendizaje 128
3.3. Composición de servicios 131
3.4. Descubrimiento de recursos 139
3.5. Razonamiento 142
3.6. Interrelación capa de coordinación semántica y módulos de descubrimiento, composición, razonamiento y aprendizaje 144

4. Implementación **147**
4.1. Modelado y coordinación semántica 147
4.2. Aprendizaje 151
4.3. Composición de servicios 157
4.4. Descubrimiento de recursos 157
4.5. Razonamiento 164

5. Validación **171**
5.1. Definición de escenario de uso y evaluación 171
5.2. Modelado y coordinación semántica 172
5.3. Aprendizaje 178
5.4. Descubrimiento de recursos 182
5.5. Razonamiento 182
5.6. Test de integración 182
5.7. Análisis de resultados 184

6. Conclusiones **185**

A. Publicaciones derivadas del trabajo en ISMED **187**

Bibliografía (Libros y artículos) **189**

Índice de figuras

2.1. Esquema del patrón Master-Worker. 4
2.2. Ejemplo de uso de servicios web y protocolos que usa. 5
2.3. La red vista desde el punto de vista de los seres humanos y de las máquinas. . . . 6
2.4. Esquema del mecanismo de comunicación en triple space. 7
2.5. Esquema de los componentes de TripCom. 11
2.6. Los Interrelación entre los tres componentes analizados con mayor énfasis. 11
2.7. Esquema de las capas en las que se apoya el DM. para localizar el espacio que albega determinada información. 14
2.8. Esquema que muestra el modo en el que QPP se relaciona con otros componentes de TripCom. 17
2.9. Diagrama de actividad para el Query Optimizer y el DM. 19
2.10. Modelo de anuncio suscripción. 22
2.11. División en distintos niveles de expresividad del API de TripCom. 23
2.12. Esquema del funcionamiento de los distintos tipos de elementos que pueden componer una red de Jxta. 29
2.13. Estructura de Sesame. 31
2.14. Estructura de la Plataforma ORDI. 32
2.15. Coordinación entre dispositivos J2ME y Relays JXTA. 38
2.16. Esquema del funcionamiento de peers JXME con el resto de peers de Jxta. 39
2.17. Módulo de Millenial. 44
2.18. Nodos sensores de Crossbow. 45
2.19. Modelo M2510 de la familia SmartMesh IA-500. 46
2.20. Nodo de la red SensiNet. 47
2.21. Modelo de mota de SquidBee Mote. 47
2.22. Hogar del futuro. 50
2.23. MavHome y MavLab. 57
2.24. iDorm. 58
2.25. Modelos de Composición. 62
2.26. Orquestación de Servicios Web [631]. 65
2.27. Coreografía de Servicios Web [631]. 66
2.28. Enfoques de descubrimiento sintácticos y semánticos, adaptado de [421] 88
2.29. Enfoques de emparejamiento semántico de servicios. 89
2.30. Comparativa de los vocabularios. 97
2.31. Motor de reglas y semántico separados. 106
2.32. Motor semántico embebido. 106
2.33. Un único motor de reglas. 108
2.34. Proceso de inferencia. 109

3.1. Funcionamiento esquemático de la primitiva query. 117
3.2. Funcionamiento esquemático de la primitiva queryMultiple. 117
3.3. Funcionamiento esquemático de la primitiva read. 117
3.4. Funcionamiento esquemático de la primitiva take. 117
3.5. Funcionamiento esquemático de la primitiva write. 117
3.6. Funcionamiento esquemático de las primitivas subscribe y advertise. 118
3.7. Services over tsc++: a) registration, b) invocation from the consumer point of view and c) invocation from the service provider point of view. 119
3.8. Esquema de la arquitectura propuesto. 121

3.9. Message exchanging between the nodes beyond the space, the WOT Gateway and a SunSPOT. . . . 121
3.10. Diagrama de clases de la tscSE. . . . 122
3.11. Diagrama de clases de la librería tscME simplificado. . . . 124
3.12. A node which joins a space ask to the rest of the members about the demands they have received. . . . 125
3.13. Esquema de la ontología de ISMED. . . . 127
3.14. Esquema de la ontología de ISMED. . . . 128
3.15. Arquitectura global del aprendizaje 128
3.16. *Conversation* representado mediante *Automaton*, *DataFlow* y *ContextFlow*. . . . 132
3.17. Automatón requerido. . . . 132
3.18. Tipos de composición soportada. . . . 133
3.19. Dimensiones para configurar las variantes de la composición basada en conversaciones. 134
3.20. Variantes de ConversationReq en base a la adaptabilidad de AdaptivityMode. . . . 136
3.21. Técnicas de composición soportadas. . . . 137
3.22. Flexibilidad en los resultados en función del grado de relajación. . . . 138
3.23. Tipo de composición en función del tipo de resultado deseado. . . . 139
3.24. Diagrama de clases del módulo de descubrimiento. . . . 141
3.25. Ontologías distribuidas frente razonamiento distribuido. . . . 143
3.26. Diagrama de clases del módulo de descubrimiento. . . . 145

4.1. Take sobre el espacio tsc://espacio5, de la que se esperan los resultados definidos en el archivo filename2.owl. . . . 148
4.2. Esquema de la aplicación de prueba de integración de tscME. . . . 148
4.3. Pantalla de conexión. . . . 149
4.4. Gestión de espacios. . . . 150
4.5. Situación de partida para el ejemplo explicado en este epígrafe. . . . 150
4.6. Pantalla inicial del interfaz gráfico. . . . 151
4.7. Transformar los datos en secuencias. . . . 152
4.8. Definición de los parámetros que permiten descubrir conjuntos de acciones frecuentes. 152
4.9. Definición de los parámetros que permiten descubrir la topología. . . . 153
4.10. Definición de los topología con la mínima granularidad. . . . 154
4.11. Selección del algoritmo para el descubrimiento de relaciones temporales. . . . 154
4.12. Tablas a separar para la generación de condiciones específicas. . . . 155
4.13. Representación de las condiciones generales. . . . 155
4.14. Esquema de la aplicación de integración desarrollada. . . . 156
4.15. Entrada de un nuevo dispositivo - conexión. . . . 159
4.16. Suscripción y notificación de llegada. . . . 160
4.17. Descubrimiento de dispositivos presentes en la red. . . . 161
4.18. Salida de un dispositivo de la red. . . . 162
4.19. Cierre de la conexión. . . . 163
4.20. Nodo monitor sin conexión. . . . 164
4.21. Nodo conectado sin dispositivos activos. . . . 165
4.22. Conexión Nokia-N95. . . . 165
4.23. Dispositivo móvil y SunSpot conectados. . . . 166
4.24. Desconexión SunSpot 166
4.25. Ventana de consulta desde la que se pueden escribir tripletas y realizar consultas de tipo queryMultiple. . . . 167
4.26. Dos nodos escribiendo información en el mismo espacio. . . . 168
4.27. El tercer nodo hace una consulta de tipo *queryMultiple* sobre el espacio y los dos nodos restantes le contestan. . . . 168

5.1. Implementation of the described scenario using resource based approach. . . . 172

Índice de tablas

2.1. Core API. 12
2.2. Extended API. 13
2.3. Further extended API. 13
2.4. Repositorio de atajos para la capa Tiple Provider y Recommender. 14
2.5. Ejemplo de pares clave-valor insertadas por cada tripleta almacenada. 15
2.6. Tabla con los atajos a kernels favoritos. 15
2.7. Comparativa de las características generales de los distintos proyectos que usan Triple Space Computing. 20
2.8. Comparativa de las principales implementaciones del paradigma del Triple Space Computing. 21
2.9. Primitivas usadas por las principales proyectos de Triple Space Computing. 23
2.10. Comparativa de los distintos productos la línea Basix. 40
2.11. Comparativa de los distintos productos la línea Connex. 41
2.12. Comparativa de los distintos productos la línea Verdex. 41
2.13. Comparativa de los distintos productos de la línea Verdex pro. 42
2.14. Características de Overo Earth. 42
2.15. Características de Sun SPOT. 43
2.16. Técnicas de Machine Learning con sus fortalezas y debilidades. 55
2.17. Resultados de la comparativa del grupo de expresividad del Lenguaje. 67
2.18. Resultados de la comparativa del grupo de Modelo de Composición. 69
2.19. Resultados de la comparativa del grupo de Modelo de Ejecución. 70
2.20. Resultados de la comparativa del grupo Entorno de Ejecución. 71
2.21. Lenguajes de representación semántica de servicios vs Efectos y precondiciones. . . 85
2.22. Ventajas y desventajas de los distintos tipos de OWL. 96
2.23. Limitaciones en la expresividad de OWL. 98
2.24. Aspectos sintácticos. 100
2.25. Problemas de OWL 1 en lo que a metamodelado se refiere. 102
2.26. Versionado e importado. 104
2.27. Anotaciones. 104
2.28. Semánticas OWL 105
2.29. OWL Full. 106
2.30. OWL1 Lite. 107
2.31. Validación de tipos OWL. 108
2.32. Comparación entre diseños de motores de inferencia. 109
2.33. Comparativa de razonadores. 111
2.34. Comparativa de razonadores. 112

3.1. Primitivas del middleware basado en Triple Space diseñado agrupadas por su naturaleza. *space* es la URI que identifica el conocimiento del grupo de nodos, *graph* es una URI que identifica un conjunto de tripletas escritas en un espacio, *triples* es un conjunto de triples y *template* expresa una secuencia de patrones de tripletas adyacente que especifican la clausula WHERE-clauses de una consulta SPARQL. *Service* e *invocation* son interfaces que permiten definir los datos necesarios para definir un servicio y su invocación. 118

4.1. Consulta SPARQL que muestra la última medida de luminosidad tomada por cualquier dispositivo ubicado en una habitación cualquiera. 169
4.2. Templates básicos obtenidos al descomponer la consulta de la Tabla 4.5. 169

4.3. Extracto de código modificado para permitir que Sesame infiera al realizar consultas. 169
4.4. Extracto de código en el que se inserta un repositorio en el mapa creado a tal efecto. 170

5.1. Número de patrones descubiertos en cada trial y tiempo de ejecución (en milisegundos). 180
5.2. Número de patrones donde ha sido posible identificar relaciones temporales cuantitativas y el tiempo de ejecución. 180
5.3. Número de patrones (en porcentaje) donde ha sido posible descubrir condiciones específicos y el tiempo de ejecución. 181
5.4. Acciones involucrados en cada una de las actividades y su orden. 181
5.5. Resultados de la evaluación de TscME (en segundos) 183
5.6. Resultados de la evaluación de acceso a datos de TscME (en segundos). 183
5.7. Rendimiento de la primitiva *demand* en un dispositivo móvil (en segundos). 183
5.8. Rendimiento del gatewayWOT (en segundos). 183
5.9. Mediciones del escenario propuesto (en segundos) 184

1. INTRODUCCIÓN

El siguiente documento recoge la memoria científica final, comprendiendo el periodo 2008-10, del proyecto ISMED (Intelligent Semantic Middleware for Embedded Devices), realizado en colaboración por la Facultad de Ingeniería de la Universidad de Deusto y Mondragon Goi Eskola Politeknikoa (MGEP). Comprende los siguientes capítulos: a) análisis crítico del estado del arte que recoge lo más avanzado en descubrimiento, composición, razonamiento y aprendizaje aplicado a entornos y dispositivos empotrados, b) el diseño de la plataforma ISMED y sus distintos componentes, c) detalles sobre la implementación de la misma para diferentes plataformas (teléfonos móviles Symbian con Java ME, Sun SPOTs, Xbee, etc.) y d) una validación técnica del trabajo desarrollado mediante la realización de un prototipo o escenario domótico sencillo que ejercita los diferentes componentes desarrollados de la plataforma ISMED.

2. ESTADO DEL ARTE

Este capítulo describe todo el análisis del estado del arte realizado en el proyecto ISMED.

2.1 Modelado y coordinación semántica

2.1.1 Introducción

Este capítulo analiza el estado del arte en modelado semántico y coordinación basada en modelos semánticos para dispositivos empotrados y móviles. Este exhaustivo estado del arte ha servido para el diseño del módulo de modelado y coordinación semántica que constituye el core de la plataforma ISMED en la sección 3.1.5.1.

2.1.2 El paradigma Triple Space computing

"Triple Space Computing" es un paradigma que trata de usar conocimiento expresado mediante semántica bajo el paradigma de "Space based computing" para lograr que distintas entidades se comuniquen mediante la publicación y consulta de su conocimiento en una memoria distribuida.

2.1.2.1 Conceptos previos

Para comprender el "Triple Space Computing" es fundamental conocer de dónde proviene, por lo que en los siguientes epígrafes se detallarán los paradigmas en los que se fundamenta.

2.1.2.1.1. Web semántica. La web semántica se basa en añadir información adicional (describiendo el contenido, significado y la relación de los datos) a la World Wide Web, para que sea posible evaluar dicha información de forma automática por parte de las máquinas. De esta forma, se mejorará Internet ampliando la interoperabilidad entre los sistemas informáticos, reduciendo la mediación de los operadores humanos.

Entre los conceptos a destacar dentro de la web semántica, se pueden enumerar los siguientes:

- RDF. Es un modelo de datos para expresar los recursos y las relaciones que se puedan establecer entre ellos. Aporta una semántica básica para este modelo de datos que puede representarse mediante XML.
- OWL. Añade más vocabulario para describir propiedades y clases: tales como relaciones entre clases (p.ej. disyunción), cardinalidad (por ejemplo "únicamente uno"), igualdad, tipologías de propiedades más complejas, caracterización de propiedades (por ejemplo simetría) o clases enumeradas.
- OWL-S. Es un lenguaje que proporciona un conjunto de construcciones para representar las relaciones presentes en una ontología, de forma que dichas ontologías puedan ser fácilmente procesables por máquinas.
- WSMO. El proyecto WSMO es una ontología para la descripción de servicios web semánticos, que no sólo pretende crear la especificación, sino que, también crea la arquitectura y facilita un conjunto de herramientas para soportar la especificación.

2.1.2.1.2. Tuple spaces. Tuple spaces es una implementación del paradigma de memoria compartida para computación distribuida con el que se logra una gran escalabilidad. Este enfoque hace que se elimine la complejidad inherente del procesado de los mensajes que de otra forma tendrían que intercambiar los procesos, haciendo que los procesos se comuniquen de forma sencilla y desacoplada escribiendo, eliminando y leyendo tuplas en un espacio global persistente visible por todos ellos.

De esta forma, por un lado existirán unas entidades denominadas productores que envían sus datos en forma de tuplas (grupos de campos ordenados) a una memoria distribuida (repositorio de tuplas) a la que se puede acceder de forma concurrente desde otros productores o consumidores de tuplas. Y por otro lado, los denominados consumidores pueden consultar o coger la información existente en el espacio en base a un determinado patrón de consulta. Así, las entidades que siguen el paradigma explicado, se comunican y coordinan compartiendo su estado en un repositorio común, siguiendo el patrón conocido como Master-Worker [62].

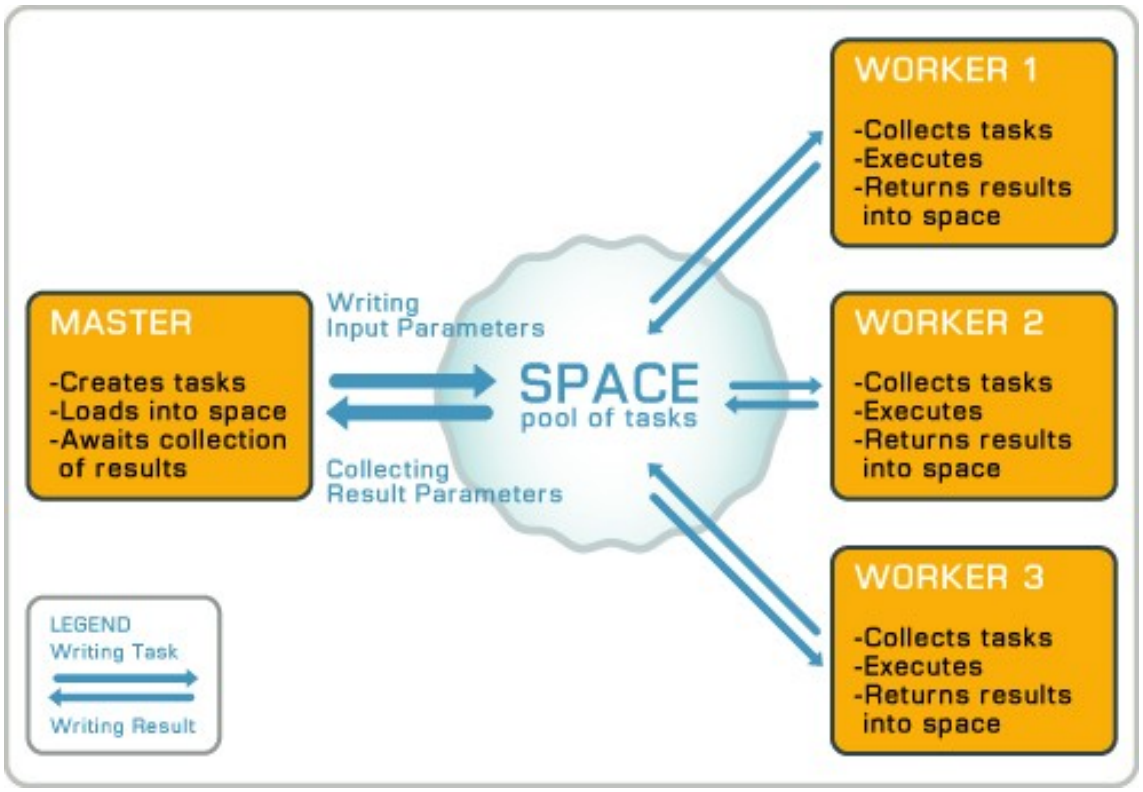


Figura 2.1.: Esquema del patrón Master-Worker.

En él, un maestro deja unidades de trabajo en un espacio. Los trabajadores leen dicho espacio, procesan el trabajo y escriben de nuevo en dicho espacio los resultados del trabajo una vez completado el mismo. Finalmente el maestro consulta los resultados de los trabajos que han realizados los trabajadores. En un entorno típico, habría varios espacios, varios maestros y muchos trabajadores. Añadiendo una capa de coordinación, este paradigma se puede convertir en un paradigma de comunicación entre procesos que no tienen por qué estar en el mismo sistema. La principal **ventaja** de dicho paradigma consiste en que logra procesos completamente desacoplados en tres dimensiones distintas referencia, tiempo y espacio.

- *Referencia:* los procesos que se comunican entre sí, no necesitan saber explícitamente de su existencia mutua. No necesitan establecer un canal comunicación entre sí.
- *Tiempo:* La comunicación puede ser completamente asíncrona porque el repositorio de tuplas garantiza el almacenamiento de los datos mediante los que se comunican las entidades.

- *Espacio:* los procesos pueden ejecutarse en entornos computacionales completamente distintos dado que todos ellos podrán acceder al mismo repositorio de tuplas.

Con dicho desacoplamiento se logran avances de diseño para definir aplicaciones reusables, distribuidas, heterogéneas y de cambios rápidos [292]. Además, mediante tuple spaces se puede lograr la exclusión mutua entre productores y consumidores de la información de forma sencilla si cada vez que se accede a un dato en el repositorio se elimina del mismo y se vuelve a depositar una vez se finalice su uso. El paradigma de "Tuple spaces" surgió de forma teórica a raíz del lenguaje de programación paralela Linda, desarrollado en la Universidad de Yale, pero hoy en día existen distintas implementaciones de dicho paradigma en distintos lenguajes, entre los que cabe destacar la realizada para el lenguaje Java denominada JavaSpaces.

2.1.2.1.3. Servicios web. De acuerdo a la definición de la W3C, "un servicio web es un sistema de software identificado por una URI, cuyos interfaces públicos y vínculos se definen y describen mediante XML. Otros sistemas de software pueden descubrir su definición, y estos pueden interactuar con dicho servicio web de la forma indicada en su definición, usando mensajes basados en XML transportados mediante los protocolos de Internet."

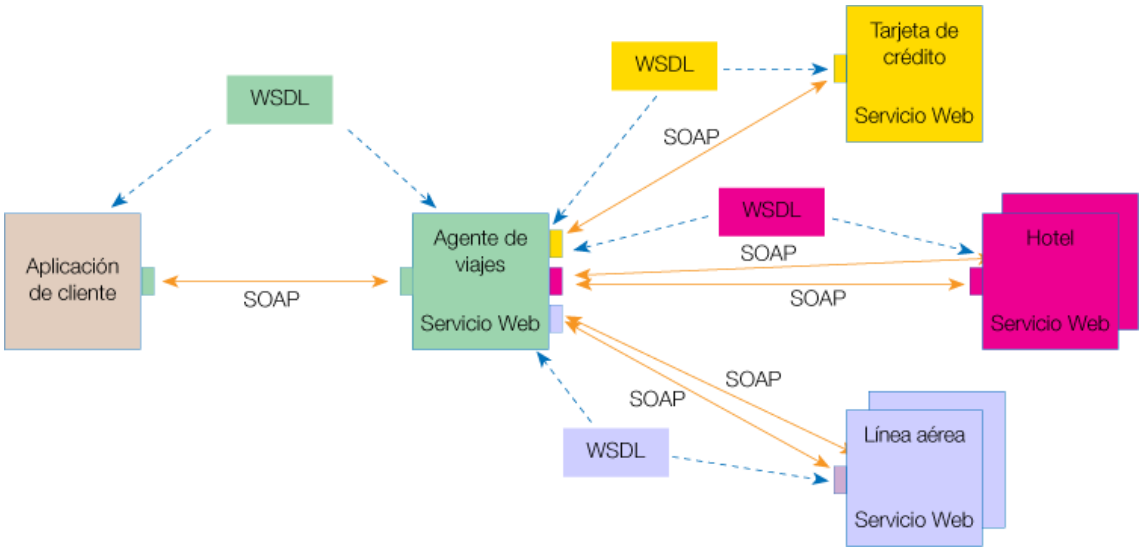


Figura 2.2.: Ejemplo de uso de servicios web y protocolos que usa.

"Dichos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar". [219]

2.1.2.1.4. Servicios web semánticos. Los servicios web semánticos [58] son una nueva infraestructura para los Servicios Web. Mientras que en los servicios web tradicionales las interfaces del servicio se definen sólo a nivel sintáctico, los servicios web semánticos proveen y extienden la descripción formal de sus interfaces mediante la notación de web semántica. Así, la aproximación de servicios web semánticos permite la integración de la Web Semántica y los Servicios Web como tecnologías complementarias. Esta nueva infraestructura permite realizar buscadores basados en Web Semántica para Servicios Web, pudiendo realizar búsquedas basadas en restricciones complejas.

```
Servicios Web (Web de aplicaciones) + Web Semántica (Web de datos) = Servicios Web

Semánticos (Web de datos y aplicaciones)
```

De está forma, los servicios semánticos permiten usar la web semántica de tal forma que mediante las ontologías se describan los servicios para que las máquinas puedan procesarlos con la mínima intervención humana.

No obstante, la actual infraestructura de registro de servicios web impide a las tecnologías de servicios web semánticos lograr plenamente sus objetivos porque el almacenamiento de los datos semánticos queda restringido a un entorno cerrado. Por eso, el usuario necesita saber explícitamente la localización de los registros al igual que el modo de acceso a los mismos. Así es cómo actualmente pese a que los servicios web se construyen y despliegan de forma independiente, habitualmente se descubren usando un registro centralizado [714].

Esta aproximación no permite mucha escalabilidad en cuanto aumentan el número de consultas y descripciones de servicios web, mermando claramente la flexibilidad de un sistema que use registros para especificar los servicios que ofrece. Además, la comunicación mediante dichos servicios sigue el paradigma de intercambio de mensajes de forma que la comunicación entre el proveedor y quien pide los datos, se lleva a cabo sin estado y de forma fugaz y síncrona.

El paradigma de Triplespace se hace especialmente útil en este momento, dado que, como se verá posteriormente, permite mejorar los servicios webs tradicionales haciendo que adopten una forma flexible, asíncrona y semánticamente mejorada.

2.1.2.2 Triple space computing

Triple Space Computing es un nuevo paradigma de comunicación y coordinación entre máquinas que trata de proveer de una infraestructura semántica de comunicación que posibilite la interacción entre las mismas. Es una extensión del paradigma tuple-spaces a la que en esencia se le han añadido todas las funcionalidades provistas por la web semántica (RDF).

En palabras de los creadores de Tripcom, una de las primeras implementaciones en desarrollo de este paradigma, "Triple Space se convertirá en la web para las máquinas igual que la Web basada en HTML se convirtió en la Web para los humanos".

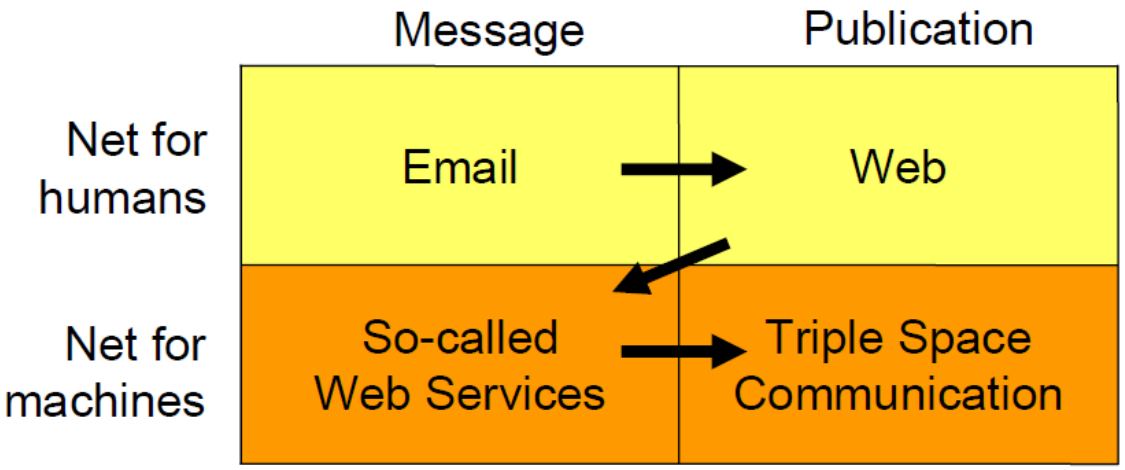


Figura 2.3.: La red vista desde el punto de vista de los seres humanos y de las máquinas.

De esta forma, igual que los servidores web publican páginas web que leen los seres humanos, los servidores Triple Space publican datos interpretables por las máquinas. Los proveedores y consumidores podrían publicar y consumir tripletas mediante una infraestructura globalmente accesible.

Así, diversos servidores Triple Space podrían estar en diferentes máquinas y cada nodo en un proceso de comunicación podría elegir su espacio preferido de forma análoga a la Web actual. Las ventajas inherentes de dicho mecanismo son que los proveedores de datos podrían publicar en cualquier momento (autonomía en el tiempo), independientemente de su mecanismo de almacenamiento interno (autonomía de localización), sin necesidad de conocer a sus potenciales lectores

(autonomía de referencia) e independientemente del esquema de datos utilizado internamente por el repositorio (autonomía de esquema) [155].

Así, se desea pasar de un entorno en el que las máquinas se comunican directamente a través de mensajes que intercambian, al uso de un intermediario que facilite el intercambio de mensajes simplificando el esquema y ofreciendo una interfaz simple.

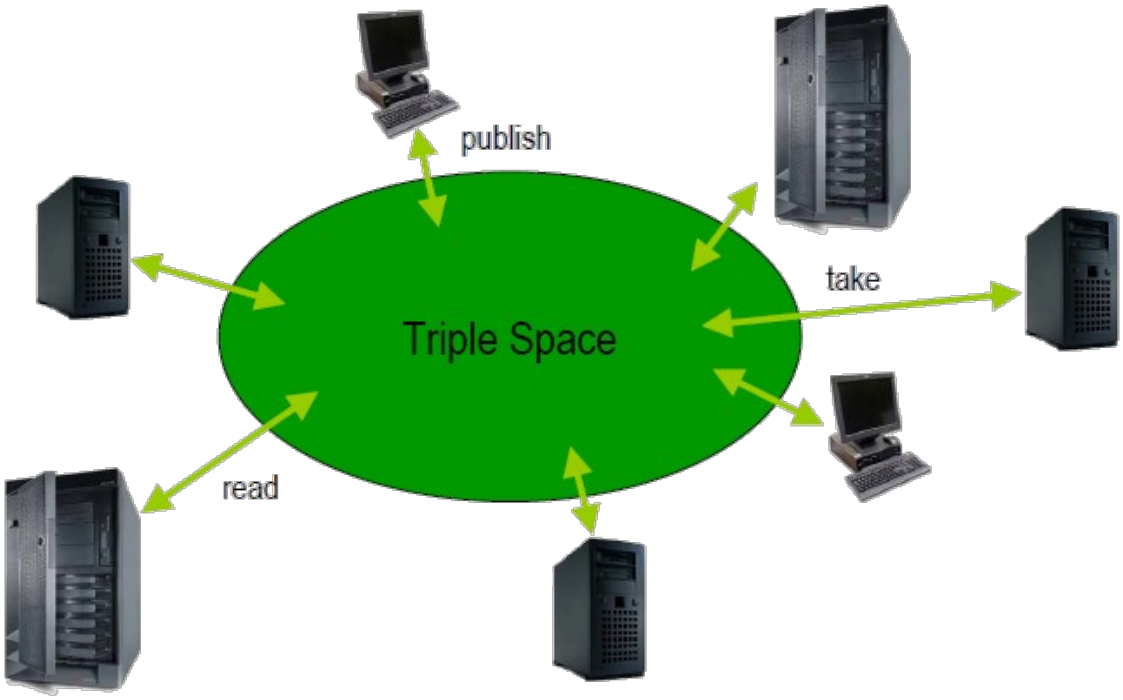


Figura 2.4.: Esquema del mecanismo de comunicación en triple space.

A continuación se detallan diversos proyectos actuales que tratan de desarrollar sistemas bajo el paradigma de Triple Space Computing.

De esta forma, se permite a las maquinas comunicarse de forma asíncrona, dotando a las aplicaciones que hagan uso de Triple Space de mayor autonomía. Resumiendo, se podrían citar las siguientes ventajas que aporta TSC [155]:

- *Comportamiento web.* El paradigma Web es bien conocido y por tanto su uso es sencillo de entender. Permite configurar una comunicación más eficiente entre los procesos que siguen el paradigma.
- *Datos semánticos.* La infraestructura TSC hace que los datos de comunicación estén semánticamente definidos, por lo que dichos datos serán entendibles por las máquinas.
- *Comunicación asíncrona.* Permite que el productor y consumidor de datos mantengan autonomía entre si, leyendo o escribiendo sólo acorde a sus necesidades e independientemente el uno del otro.
- *Reducción de la incertidumbre, retraso y complejidad.*
  - La incertidumbre en la comunicación se reduce dado que los datos se pueden escribir persistentemente (así ni siquiera hace falta que el TSC devuelva un mensaje de error, porque éste se podrá consultar más tarde en dicho espacio). No existen más preocupaciones por las interrupciones de la red.
  - El retraso en la comunicación se elimina dado que el lector puede leer cuando le sea necesario y el productor puede escribir siempre que lo necesite.

- Mínimo impacto en el lector o productor de información. No existe impacto salvo por el acceso al Triple space mediante el protocolo de transferencia de Triple space, que es tan ligero como el propio protocolo HTTP. Además, no existe la necesidad de esperarse mutuamente para poder realizar la comunicación de forma síncrona.

2.1.2.3 Proyectos relacionados

A continuación se detallan diversos proyectos actuales que tratan de desarrollar sistemas bajo el paradigma de Triple Space Computing.

2.1.2.3.1. Triple Space Computing (TSC). El objetivo del proyecto TSC es desarrollar una plataforma genérica que sirva de prototipo para entornos Triple Space Computing, posibilitando la comunicación y coordinación de la web semántica y de los servicios web semánticos. Así, extiende el paradigma de "Tuple Space Computing" realizando un middleware mejorado con semántica, escalable y que presta especial atención a los servicios web semánticos. TSC permite a aplicaciones Java crear y acceder a "Triple Spaces" para leer y escribir grafos RDF y realizar todas las operaciones definidas en el modelo conceptual de Triple Space Computing. La aplicación se compone de tres bloques principales [680]:

- *Corso (Coordinated Shared Objects Space),* usado para la capa de coordinación del Triple Space (TS) kernel, distribuye los espacios TSC entre múltiples TS kernels.

  CORSO es un middleware escrito en lenguaje C de memoria compartida para sistemas distribuidos heterogéneos, que permite solucionar los problemas de las aplicaciones distribuidas (compartir una memoria local de cada cliente con otras en un espacio): replicación, migración de datos, información de acceso, transacciones y tolerancia a fallos.

- *YARS (Yet Another RDF Repository),* es el repositorio semántico que usa la capa de acceso a datos del TS kernel, responsable de buscar grafos en base a determinados patrones de búsqueda. YARS es un servicio web que debe desplegarse en un servidor web Tomcat, en cada ordenador que ejecute TS kernel (idealmente los clientes ligeros podrían usar un repositorio de clientes más pesados).

  YARS permite agregar información, recuperarla o eliminarla mediante simples consultas HTTP. Los resultados se devuelven en formato NTriples y las consultas se pueden realizar de forma más expresiva que usando simples patrones con la forma "<sujeto,predicado,objeto>", mediante el lenguaje de consultas N3QL.

- *Librería Java TS kernel.* Implementa el modelo TSC y los acopla con Corso y YARS, implementando la capa de coordinación y acceso a datos.

  Así, para mantener la capa de coordinación y la de acceso a datos sincronizados, se implementan dos eventos: GraphListEvent (que se activa cada vez que un grafo cambia en un determinado espacio) y GraphTakeEvent (que se activa cuando cambia un objeto y se marca como eliminable).

  De esta forma, cada grafo escrito en un espacio que es conocido por el kernel local tiene que sincronizarse con el repositorio YARS local usando Corso cuando se den esos eventos.

Ente los principales inconvenientes del TSC se pueden enumerar los siguientes:

- La robustez del TS kernel no tolera una desconexión temporal de Corso. Si esto ocurre, se debe crear una nueva instancia del TS Kernel.

- Cómo Corso se usa en modo persistente, almacenando espacios de forma confiable, actualmente la capa de acceso a datos sólo se usa para consultas mediante patrones de búsqueda. Para realizar este tipo de consulta puede ser suficiente una representación en memoria o incluso un índice de los grafos.

  De esa forma, la capa de acceso a datos podría funcionar en la misma máquina virtual de Java que la aplicación cliente, para lo que habría que portar YARS (lo cual no debería suponer un esfuerzo dado que YARS está implementado en Java.

- Idealmente también Corso, responsable de distribuir espacios en la red, podría ejecutarse en la misma máquina virtual de Java que la aplicación cliente. Desgraciadamente, portar Corso supondría un esfuerzo demasiado grande dado que está implementado en C.

- Latencia en el acceso a la capa de acceso a datos, dado que cada operación sobre el repositorio se traduce en una serie de consultas HTTP a YARS.

- El diseño de las estructuras de datos que usa Corso hace que a menos objetos Corso, se necesiten menos consultas a Corso, pero que se obtengan más datos desde Corso (que al final habrán de ser procesados por el cliente de la máquina virtual de Java). Por el contrario, más objetos acarrean más consultas a Corso, pero menor proceso a llevar a cabo en la máquina virtual Java del cliente. Haciendo que las estructuras de Corso fueran finamente granulares (fine-granular) se reducirían los conflictos transaccionales en caso de operaciones concurrentes sobre el mismo espacio.

- Cada componente puede proveer acceso a múltiples clientes a un solo TS kernel, manteniendo replicas de Triple Spaces. El TS kernel puede ser usado por un único cliente (cliente pesado), lo que quiere decir que cada cliente necesitará una instalación partícular de Corso y YARS, lo que puede suponer una sobrecarga innecesaria si múltiples clientes se sitúan en una misma máquina o área local.

- La licencia de Corso es una licencia privativa, que no permite por tanto acceder y modificar el código.

Está última desventaja hizo que el proyecto de TSC como tal fuese reconcebido y modificado sustancialmente, dando paso al proyecto tsc++ que se explica en el siguiente epígrafe.

2.1.2.3.2. tsc++. tsc++ [450] es una implementación del paradigma TSC, que hace uso de P2P y tecnologías de web semántica como RDF(tripletas) y SPARQL. Este projecto trata de desarrollar el paradigma Triple Space Computing como un framework middleware para la comunicación y coordinación de web semántica y servicios web semánticos. Usa Sesame u Owlim para almacenamiento persistente de las tripletas RDF, mientras para la comunicación de red y coordinación se realiza a traves del framework JXTA.

tsc++ está basado en el proyecto TSC explicado anteriormente y dados los problemas de licencias causados por el software propietario CORSO en la capa de coordinación se sustituyó éste por Jxta.

Pese a que en sus primeras versiones el proyecto tsc++ heredó el repositorio semántico usado por TSC YARS, éste fue sustituido paulatinamente por Sesame y Owlim, hasta que en la versión 1.1 (la última publicada hasta la fecha) fue suprimido por completo.

Tsc++ puede requerir gran cantidad de memoria y ancho de banda, por lo que la ejecución de multiples instancias del kernel en una sola máquina es dificultosa.

2.1.2.3.3. Proyecto TripCom. A continuación se muestra un análisis de los componentes del proyecto TripCom relacionados con la capacidad de realizar consultas SPARQL complejas, que más adelante (en el epígrafe referencia) dará paso a una reflexión sobre la idoneidad de adaptar las distintas técnicas usadas en TripCom o no al proyecto de investigación ISMED.

2.1.2.4 Introducción

El proyecto TripCom es altamente modular y cada kernel está compuesto por los siguientes componentes [584]:

- Triple Store Adapter recibe las consultas y lee los datos almacenados en el repositorio pertinente.

- Security Manager asegura que las operaciones no violan la política de seguridad especificada.

- Mediation Manager ejerce de intermediario entre las tripletas o grafos entrantes y salientes.

- Metadata Manager es responsable de la implementación de la funcionalidad descrita por la ontología de Triple Space. Además, ejerce de razonador, validando la instancia de la ontología con su esquema y razona para encontrar información de los datos requeridos por los usuarios tales como estadísticas de acceso para tripletas, grafos, etc.
- Query Processor descompone las consultas en partes que pueden ser satisfechas por el repositorio local y en partes que deben ser enviadas a otros kernels.
- Transaction Manager gestiona las transacciones locales y participa en las transacciones distribuidas, en las que puede ser el coordinador de la misma si el kernel al que pertenece inició dicha transacción.
- Distribution Manager es el responsable de realizar un Triple Space distribuido, reenviando consultas y peticiones a kernels que probablemente sean capaces de satisfacerlas parcialmente y reenviando las peticiones de escritura a los kernels apropiados.
- Management API se usa por los administradores para inicializar las estructuras de datos y coordinar componentes del kernel. Es quien inicia todo el proceso del kernel.
- Triple Space API define el conjunto de operaciones que se ofrecen a los clientes del TSC.
- Web Service Invocation traduce las interacciones con los servicios web a operaciones del TS API.
- Web Service Discovery buscar descripciones de servicios web que concuerden con los objetivos de los solicitantes del servicio.
- Web Service Registry ofrece la funcionalidad necesaria para transformar descripciones de servicios web en grafos RDF y para almacenarlos en el Triple Space.

Además, existen las siguientes entidades externas al kernel.

- Triple Space Client permite crear clientes que acceden al Triple Space usando directamente su API.
- Service servicios webs definidos a un nivel superior del Triple Space que pueden ser usados por clientes de servicios web.
- Mediation Service provee la funcionalidad para mapear entre conceptos y ontologías, bien vía la API del servicio web bien vía el TS API.
- Orchestration Engine permite la composición de servicios.

Para la problemática concreta que nos preocupa, se ha analizado el Triple Space API (dónde se especifican las operaciones que soporta cada Kernel), el Distribution Manager (encargado de localizar la información distribuida en los espacios y kernels) y el Query Preprocessor (por su capacidad de realizar consultas de una expresividad mayor).

En la Ilustración 2.6, se puede ver a un nivel alto de abstracción la relación entre los tres componentes anteriomente mencionados. El TS API provee a los clientes de una serie de operaciones. Para cumplir dichas operaciones, se apoya en el Distribution Manager, que se coordina con otros kernels remotos y con el repositorio semántico local y que por otro lado puede apoyarse en el Query Preprocessor para llevar a cabo consultas de una mayor complejidad.

2.1.2.4.1. Definiciones. Para comprender plenamente lo que se explicará en las siguientes secciones, cabe aclarar cierta terminología usada en TripCom [584].

- Entidades lógicas:
  - Triple Space: es un conjunto de datos semánticos identificados por un mismo nombre. Puede estar distribuido a lo largo de un conjunto de nodos al que se puede acceder a través de las primitivas del API Triple Space.

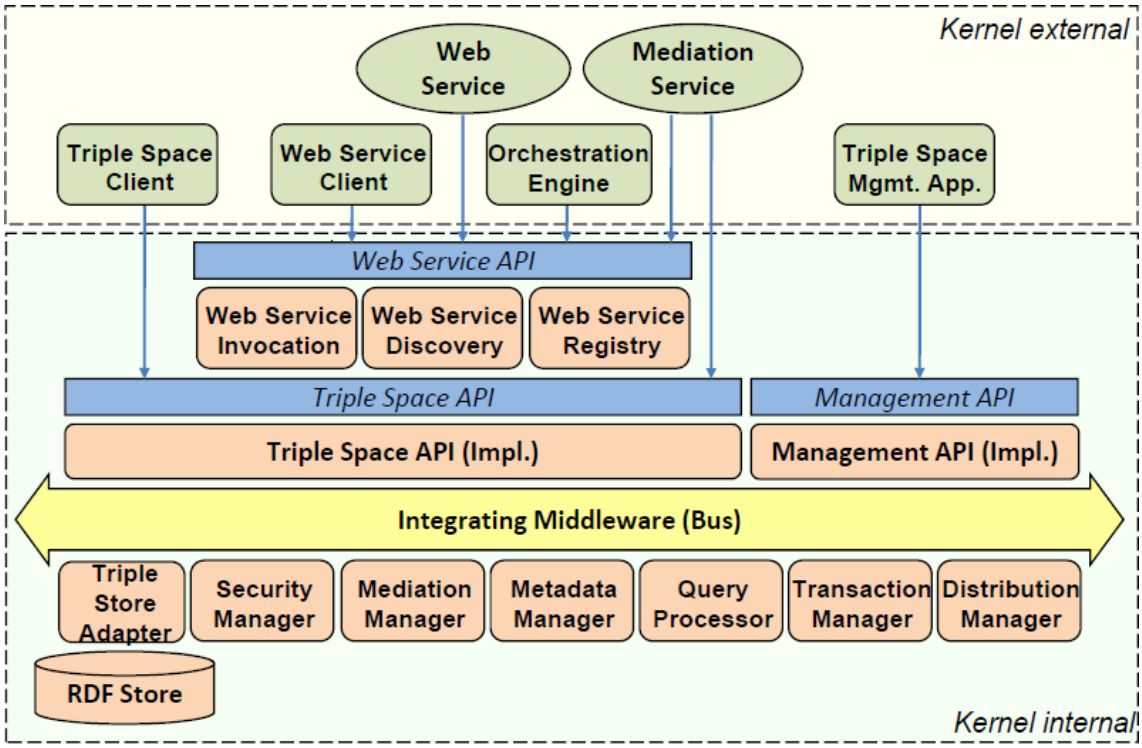


Figura 2.5.: Esquema de los componentes de TripCom.

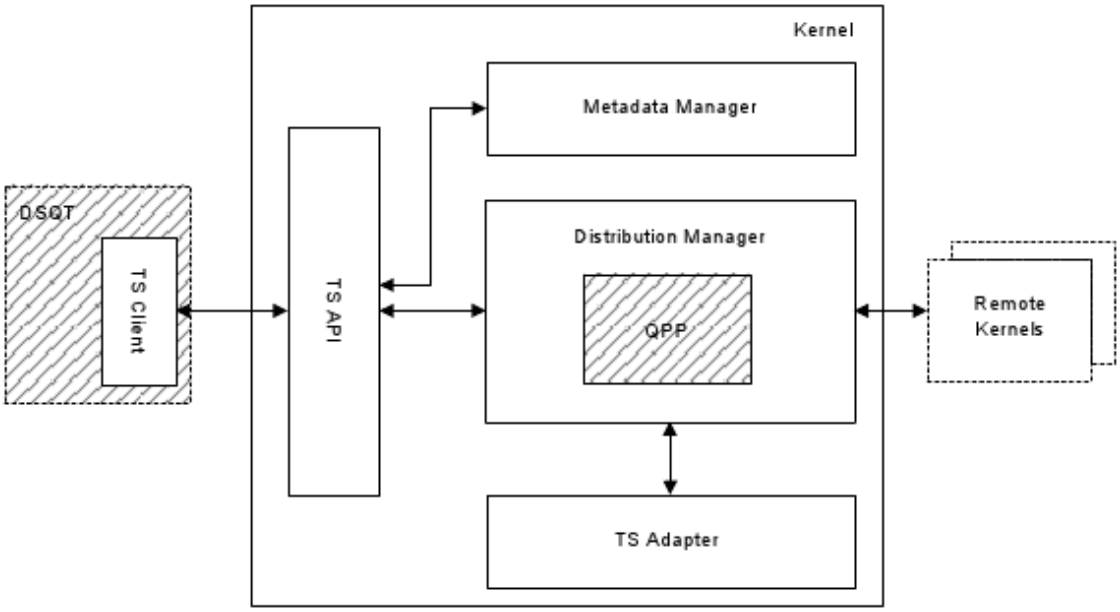


Figura 2.6.: Los Interrelación entre los tres componentes analizados con mayor énfasis.

- Subespacio: es un subconjunto de datos en un Triple Space identificado por un contexto específico. Cumple con la definición de un Triple Space y puede tener sus propios subespacios.
- Nodo: un nodo es una instancia de un kernel que consecuentemente ofrece un API a sus clientes. Una autoridad única gestiona cada nodo que es identificado por un identificador lógico único.

▪ Entidades físicas:

- Kernel: es una implementación física única compuesta de todos los componentes que implementan el API Triple Space. Un kernel puede desplegarse en una infraestructura física y puede ser direccionado usando una dirección de red física (bien como una única máquina física, bien como un único clúster de máquinas) como podría ser tsc://kernel1.morelab.deustotech.es:2588. Finalmente un kernel puede albergar un número arbitrario de nodos.
  - Kernel local. Para un cliente determinado, un kernel local es aquel al que se encuentra conectado y el que usa para realizar operaciones sobre el Triple Space.
  - Kernel remoto. Para un determinado cliente, es aquel kernel al que no se encuentra directamente conectado.
- Componente del kernel: es una parte del kernel que proporciona una funcionalidad específica y asume cierta responsabilidad en la arquitectura del kernel.

2.1.2.5 TS API

TripCom ofrece tres APIs distintas [600] sobre las que realizar operaciones de diferentes niveles de complejidad sobre el Triple Space. Estos tres niveles son: básica (core API), extendida (extended API) y otra más extendida aún (further Extended API).

Operación	Parámetros	Devuelve	Descripción
out	Triple, URL	-	Escribe atómicamente una tripleta en el espacio especificado.
rd	SingleTemplate, URL, timeout	Set<Triple>	Devuelve un resultado que coincida con el patrón especificado en el espacio identificado por su URL.
rd	SingleTemplate, timeout	Set<Triple>	Misma operación que la anterior, pero dado que no especifica la URL del espacio, debe seleccionar un kernel sobre el que tenga permiso de lectura y leer.

Tabla 2.1.: Core API.

2.1.2.5.1. Core API. En este nivel inicial, las plantillas sobre los que se realizan las busquedas de tripletas, son patrones de tripletas individuales (tripletas que pueden contener variables).

2.1.2.5.2. Extended API. En este nivel se permite el uso de plantillas más expresivas (por ejemplo, consultas SPARQL).

2.1.2.5.3. Futher extended API. Véase la tabla 2.3

Operación	Parámetros	Devuelve	Descripción
out	Triple, URL	-	Escribe atómicamente un conjunto de tripletas.
rd	Template, URL, timeout	Set<Triple>	Es la misma operación descrita en la Core API pero usando plantillas más expresivas para la consulta.
rd	Template, timeout	Set<Triple>	Mismo caso que la anterior.
rdmultiple	Template, URL, timeout	Set<Set<Triple>>	Devuelve múltiples coincidencias para una plantilla dada.
subscribe	Template, Callback, URL	URI	Se crea suscripción de modo que se avise al Callback cuando se cumpla la plantilla dada en el espacio identificado por la URL.
unsubscribe	URI	-	Cancela una suscripción.

Tabla 2.2.: Extended API.

Operación	Parámetros	Devuelve	Descripción
in	Template, URL, timeout	Set<Triple>	Funciona como la rd, pero elimina los triples devueltos dentro del espacio definido.
inmultiple	Template, URL, timeout	Set<Set<Triple>>	Elimina las múltiples respuestas que devuelve.
createTransaction	String [“local”, “shared”]	URI	Crea una transacción en un cliente (sólo él conoce la URI que la identifica), que puede compartirse con otros clientes.
getTransaction	URI	boolean	Se usa para unir transacciones de tipo shared con otros clientes.
beginTransaction	URI	boolean	Comienza la transacción y consecuentemente todas las interacciones entre agentes se realizarán “o todas o ninguna”.
commitTransaction	URI	boolean	Se ejecuta una operación de commit sobre la transacción dada.
rollbackTransaction	URI	boolean	Se ejecuta una operación de rollback sobre la transacción dada.

Tabla 2.3.: Further extended API.

2.1.2.6 Distribution Manager

El conocido como Distribution Manager, en adelante DM, es el componente encargado de buscar kernels remotos, encaminar operaciones de otros kernels y recopilar información para dar respuestas a las operaciones pedidas por los clientes.

Como se ha comentado previamente, una URL identifica cada espacio. Sabiendo sobre qué espacio queremos realizar una operación, para llegar a él es tan sencillo como resolver mediante DNS a que máquina debemos dirigir dicha operación.

Sin embargo, cuando en una operación no se facilita la URL de un determinado espacio, debemos localizar quien es el encargado de la información concreta tratada en dicha operación para que la resuelva. En este caso el DM cobra especial importancia dado que es quién nos permitirá saber qué espacio buscamos.

Para dicha tarea, el DM se apoya en cuatro estrategias diferentes: tres que hacen uso de un conocimiento semántico local y que permiten un mayor rendimiento y otra que hace uso de índices sobre las tripletas manejadas.

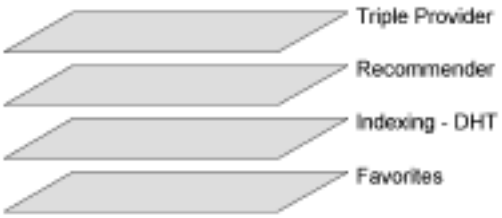


Figura 2.7.: Esquema de las capas en las que se apoya el DM. para localizar el espacio que albega determinada información.

Las capas a las que se ha hecho referencia anteriormente son las siguientes [599]:

- **Triple Provider.** Esta capa sabe quién supo responder a una determinada consulta en el pasado (el llamado Triple Provider Kernel). Para ello crea atajos a los kernels siempre que recibe una respuesta y mantiene una tabla local con dicha información.

- **Recommender.** De forma semejante al Triple Provider almacena atajos a kernels que pese a que no respondieron directamente una consulta (no son TPKs), sí que supieron encaminarla hacía algún kernel que supo responderla (TPK). Así, tras consultar a esta capa, se reenviará la consulta a un kernel que supo responderla en el pasado, bajo la premisa de que si supo responderla en el pasado sabrá responderla en el presente (dicho de otro modo, seguirá conociendo al TPK).

  El uso de esta segunda capa, puede llevar al reenvío de operaciones a través de múltiples kernels, lo cual a su vez podría llevar a que se creen ciclos. Para evitar dicha situación, con cada consulta se reenvía una lista de kernels visitados en el proceso que será tenida en cuenta en cada DM.

  Estas dos primeras capas almacenan sus atajos de forma similar (especificando en el tipo T o R dependiendo del caso del que se trate) en el mismo repositorio local, que tendrá una estructura semejante a la mostrada en la Tabla 2.6.

Graph Patterns	KernelURL Basic	Hits	Type	Timestamp
(?s, rdf:type, ?o)	tsc://fu-berlin.de	43	T	54656476476
(?s, rdf:type, dam:cc)	tsc://inf.fu-berlin.de	12	R	45747473457

Tabla 2.4.: Repositorio de atajos para la capa Tiple Provider y Recommender.

Clave	Valor
Hash(s)	(s,p,o,SpaceURL)
Hash(p)	(s,p,o,SpaceURL)
Hash(o)	(s,p,o,SpaceURL)
Hash(SpaceURL)	(s,p,o,SpaceURL)

Tabla 2.5.: Ejemplo de pares clave-valor insertadas por cada tripleta almacenada.

- **Indexing – DHT.** Esta capa utiliza índices para localizar información relativa a los espacios. Estos índices se almacenan en una base de datos distribuida (UniStore), que usa el sistema de P2P PGrid como red subyacente.

  Para comprender su funcionamiento, bastaría con poner a modo de ejemplo un proceso de consulta, en el que partiendo de la misma, se realizarían los siguientes pasos:

  1. Determinar las claves (basic triple patterns) en base a la consulta SPARQL.
  2. Construir una consulta SQL para las claves obtenidas en el primer paso, y enviarlas al almacén de índices distribuido. El sistema P2P encaminará la consulta y la respuesta hacía y desde el kernel con la instancia del almacén de índices que contiene los datos relativos a la clave consultada.
  3. Obtener la respuesta, con una serie de spaceURLs que contienen datos para las claves facilitadas.
  4. Reenviar la consulta SPARQL a uno de los espacios.

- **Favorites.** Partiendo de los atajos almacenados en el repositorio de atajos local usado para las capas Triple Provider y Recommender, se crea otra tabla con los kernels más activos y que consecuentemente se cree que pueden responder más rápido a una consulta.

  Para ello se clasifican los kernels atendiendo a un factor que se obtiene multiplicando el número de atajos que un kernel ha creado por el número de kernels remotos que conoce (dato que se conoce preguntando directamente a los mismos). Dicho factor se actualiza siempre que se actualiza la tabla de atajos. En dicha tabla sólo se mantienen los kernels más favoritos, dado que serán los que se usarán para las operaciones. De esa forma se consiguen reducir sustancialmente el número de mensajes necesarios para actualizar cada tupla.

KernelURL	Shortcuts	Kernels	F. Factor
tsc://fu-berlin.de	20	30	600
tsc://inf.fu-berlin.de	32	10	320
tsc://mi.fu-berlin.de	15	20	300

Tabla 2.6.: Tabla con los atajos a kernels favoritos.

2.1.2.6.1. Proceso de almacenamiento. Para comprender mejor el papel que juega el Distribution Manager al almacenar tripletas, por ejemplo al dar respuesta a una operación out, a continuación se detalla el proceso que seguiría [599].

1. Un cliente se conecta directamente a un kernel para acceder al Triple Space API. 2. El cliente realiza una consulta out especificando un espacio.
   a. El kernel local comprueba si es responsable de dicho espacio y por lo tanto es el encargado directo a la hora de resolver la operación.
      i. Si no es el responsable, reenvía la operación al kernel pertinente que será quien lleve a cabo la operación.
   b. Si el kernel local es responsable, se comprueba si la indexación está permitida.

i. En caso de ser así, envía una petición de almacenamiento para el valor (s,p,o,SpaceURL) al sistema de almacenamiento de índices.

c. Almacena la tripleta en la capa de almacenamiento RDF mediante el TS Adapter.

Dado que en una operación out se índica en todo momento en que URL se desea almacenar una tripleta, queda patente que la distribución es controlada en todo momento por el usuario (pese a que una vez escrita no se permita la reubicación de la misma en otro kernel distinto).

Por ello, no es misión del DM distribuir dichas tripletas en sí (pese a que se contempló la posibilidad de usar distintas estrategias de distribución en [599]), sino localizarlas a través de los espacios en los que se encuentran distribuidas.

2.1.2.6.2. Proceso de consulta. Cada consulta se procesa en el Triple Store Adapter del kernel responsable de una determinada tripleta.

Existen dos tipos de consulta, con espacio y sin espacio. Las consultas con espacios se ejecutan de forma similar al proceso de almacenamiento, las consultas sin espacios deben hacer uso de las cuatro capas explicadas con anterioridad para localizar el kernel al que reenviar una consulta.

- Operaciones Read con Space URL. El DM comprueba si el espacio objetivo está en el kernel local o en un remoto.
  - Para reenviar la operación de lectura a un kernel remoto, el DM coge la URL del espacio y resuelve la dirección IP del kernel usando DNS. La comunicación entre kernels se realiza usando el protocolo SOAP y servicios web (DM usa un servicio web para conectarse a la implementación del API del kernel remoto).
  - Por cada lectura exitosa, crea un atajo para obtener más conocimiento de la red gradualmente.
- Operaciones READ sin Space URL.
  - Comprueba los espacios locales. Si la consulta se puede resolver localmente, devuelve los resultados al cliente.
  - Si no, comprueba la tabla de atajos y si existe una coincidencia exacta con reenvía la consulta a dicho kernel.
  - Si no encuentra un atajo, envía la consulta al almacén de índices distribuidos para obtener valores para dichos índices. El sistema P2P encaminará la consulta hacía el DM del kernel correspondiente y devolverá sus resultados.
  - Si no obtiene respuesta alguna del almacén, se reenviará la consulta a los kernels favoritos.
  - Pasado un periodo de timeout, si no consigue localizar un atajo, probará con la capa por defecto que es la tabla de índices distribuida.

Finalmente, para realizar procesamiento de consultas distribuido (sólo para read sin espacio), hay que usar el subcomponente QPP que se explicará con mayor detenimiento en el siguiente epígrafe.

2.1.2.6.3. Sobre la inferencia de nuevas tripletas. Una vez realizado el proceso de consulta y encontrado kernels que potencialmente respondan una consulta, se reenvía la consulta a los mismos y la consulta se resolverá en su almacén RDF local usando su TS Adapter.

En este momento, las tripletas inferidas o virtuales (que no existen explícitamente en el almacén) se pueden añadir a la respuesta en cada kernel local usando el razonador del mismo. Desafortunadamente, el proceso de razonamiento distribuido sobre el conocimiento global completo de la infraestructura Triple Space no fue objeto del proyecto TripCom.

2.1.2.7 Query PreProcessor

QPP es el componente de TripCom encargado de realizar el procesamiento de consultas distribuidas para operaciones *read sin espacio*. Este modulo es completamente independiente a la capa de razonamiento y almacenamiento, y se basa en SPARQL cost model, que es una técnica adaptada del campo de las bases de datos distribuidas.

QPP tiene a su vez dos partes principales [598]:

- **Query Optimizer.** Este componente permite reescribir consultas, estimar costes, ordenar joins y select, descomposición de consultas y construcción de respuestas.
- **Inconsistency reasoner.** Este detecta y evita inconsistencias como primer paso, de modo que asegura que los resultados obtenidos del procesador de consultas locales es consistente con la ontología antes de devolverlos al cliente.

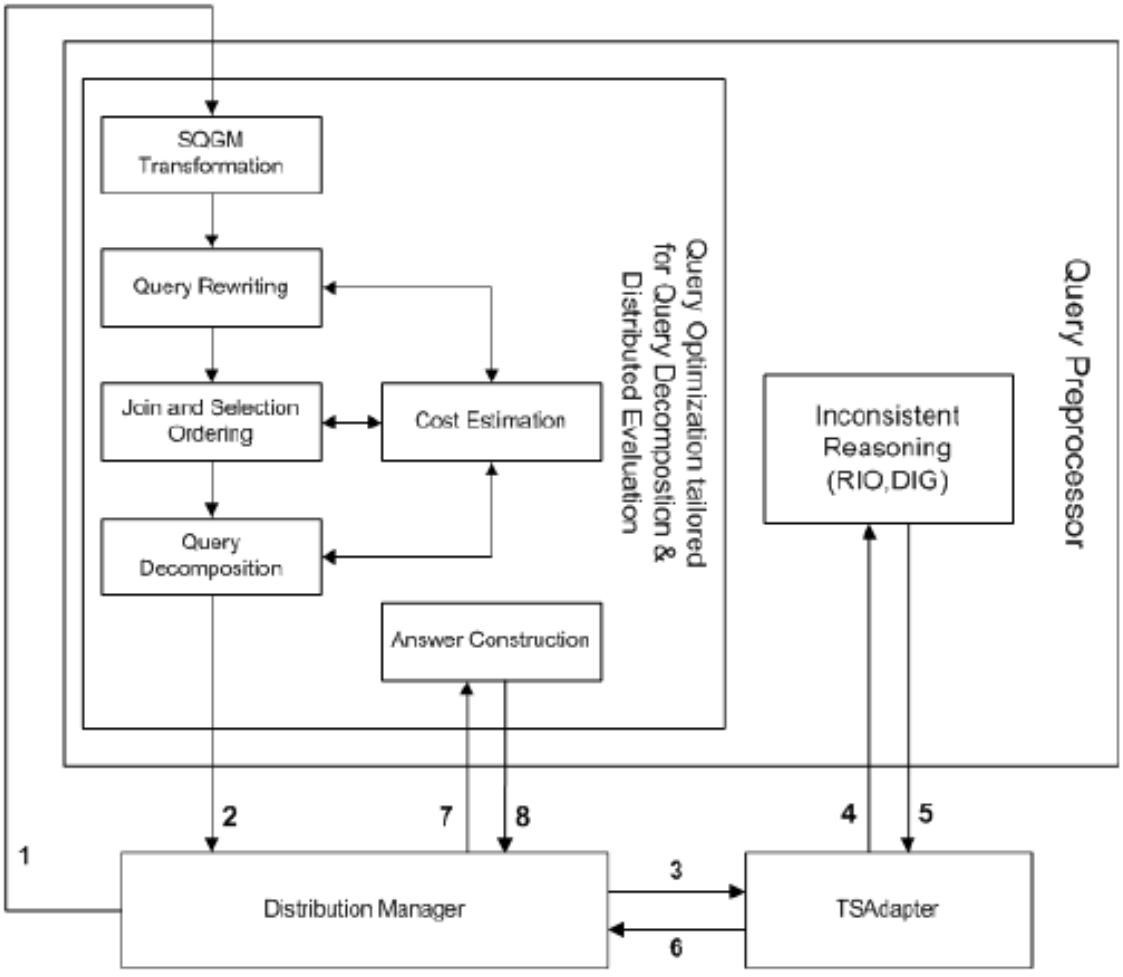


Figura 2.8.: Esquema que muestra el modo en el que QPP se relaciona con otros componentes de TripCom.

En la Ilustración 2.8 se muestra un esquema del modo de proceder del QPP para dar respuesta a una consulta. En ella se pueden apreciar las interacciones de los distintos subcomponentes del QPP con otros componentes de TripCom.

2.1.2.7.1. Query Optimizer. Para comprender mejor el funcionamiento del QueryOptimizer del QPP, se describirá a continuación el proceso que sigue el componente analizado al responder a una consulta [597].

1. Parsing. Se obtiene un árbol de operadores de la consulta SPARQL mediante el motor ARQ.
2. Logical optimization. Se optimiza la consulta transformándola mediante reglas de escritura en una equivalente pero más eficiente en coste temporal.
3. Query decomposition. En base a los metadatos recibidos del DM sobre los kernels, se encuentra el camino más eficiente mediante el que resolver la consulta y en base a ello se descompone la consulta en subconsultas (más concretamente en planes de consulta).
4. Optimización física y ejecución de la consulta.
   a. Se estima el coste físico en base a estadísticas (número de instrucciones de CPU, número de operaciones de E/S y retrasos introducidos por la red a la hora de acceder a datos remotos) para escoger el mejor plan a ejecutar
   b. Se reenvía a los DM los planes de consulta parciales
   c. El DM devuelve los resultados al QPP cuando le llegan
   d. El QPP construye respuestas parciales (realizando los bindings correspondientes)
   e. En base a los resultados obtenidos el QPP crea las siguientes consultas parciales a procesar y repite el proceso desde el punto b
   f. Eventualmente, tras un número finito de repeticiones, se obtendrá un conjunto de soluciones mapeadas

2.1.2.7.2. Inconsistency reasoner. Este subcomponente del QPP provee de datos al TS Adapter de modo que le permita soportar comprobación de la inconsistencia en las respuestas que obtiene del repositorio semántico local [598].

En caso de encontrar inconsistencias, se obtiene un subconjunto del resultado que no contiene ninguna inconsistencia con las ontologías del Triple Space. Para realizar esa comprobación, se usa un razonador especializado conocido como RIO. El RIO encuentra y resuelve cualquier inconsistencia devolviendo un subconjunto consistente de la ontología.

De esa forma, se puede realizar de nuevo la consulta, tras haber eliminado la inconsistencia de la ontología, sobre una teoría lógica consistente.

2.1.2.7.3. Comparativa. Cómo se ha visto, existen dos plataformas principales, ambas todavía en proceso de desarrollo, que siguen el paradigma Triple Spaces: TSC y TripCom. Ambas constan de múltiples capas, de entre las que destacan por su importancia la capa de coordinación y la de acceso a datos. Como se resume en la siguiente tabla, ambas implementaciones hacen uso de distintas tecnologías en las distintas capas.

El proyecto TSC como tal no se finalizó, dado que como se ha explicado, se paso a desarrollar el tsc++ que no es sino una versión mejorada del mismo. Pese ello, el TSC planteaba dos dificultades principales que se han podido ver en la tabla:

- La capa de coordinación estaba implementada con Corso, que no sólo estaba desarrollada en C y compilada para Windows, sino que tenía una licencia privativa que no permitía realizar modificaciones sobre el mismo con el fin de portarla a un hipotético dispositivo móvil que ejecutase programas en C.
- La capa de acceso a datos hacía uso de YARS, que es un servicio web que debe desplegarse en un **servidor web con contenedor de servlets** como **Tomcat**. La posibilidad de hacer que el dispositivo móvil pudiese ejecutar una versión reducida de Tomcat resulta utópica, pero para paliarlo se barajó la posibilidad de que clientes ligeros (por ejemplo dispositivos móviles) accediesen al servicio web YARS de clientes más pesados.

Analizando las características de Tsc++ y TripCom, podemos apreciar cómo ambos utilizan repositorios semánticos semejantes, y que su principal diferencia reside en la forma en la que se lleva a cabo la coordinación entre distintos Kernels. Así mientras Tsc++ opta por usar Jxta para hacer que los Kernels se comuniquen entre si y se pidan los datos necesarios, TripCom utiliza DNS

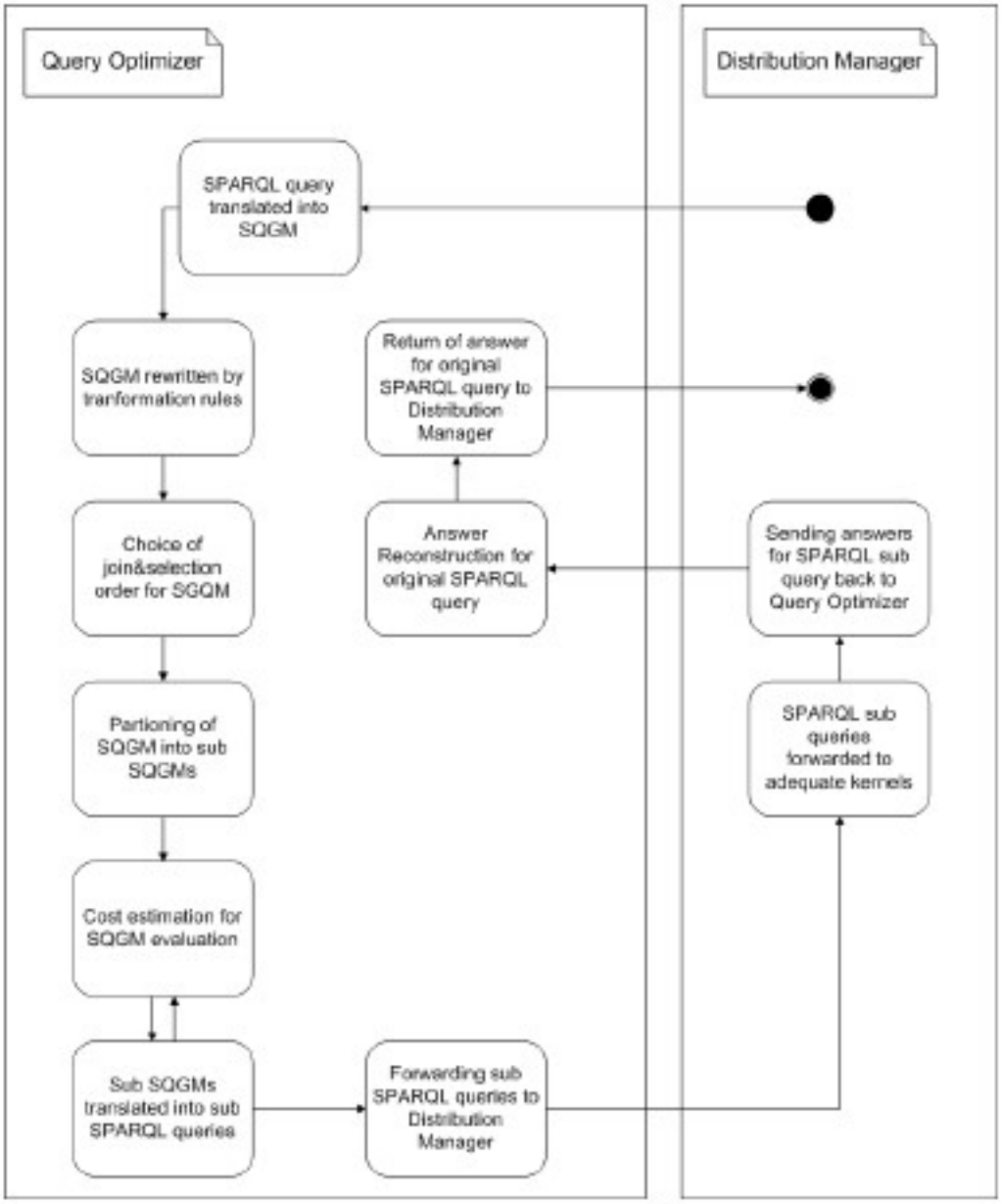


Figura 2.9.: Diagrama de actividad para el Query Optimizer y el DM.

	TSC	Tsc++	TripCom
Capa de coordinación	Corso	Jxta	DNS y P-Grid
Capa de acceso a datos	Yars	Sesame, Owlim	ORDI ▪ SGDB ▪ YARS ▪ OWLIM ▪ Sesame
Licencia	Licencia libre salvo en la capa de coordinación	GNU Lesser General Public License (LGPL)	GNU Lesser General Public License (LGPL)

Tabla 2.7.: Comparativa de las características generales de los distintos proyectos que usan Triple Space Computing.

para localizar los kernels y P-Grid para llevar a cabo una indexación de los nombres de los kernels que tienen determinados datos en cada uno de los kernels.

Si se tiene en cuenta que DNS es una base de datos distribuida y jerárquica que almacena información asociada, y que se tarda un tiempo considerable en actualizar dicha información a lo largo de internet, no parece que sea un estructura lo suficientemente flexible como para adaptarse al entorno cambiante que forman los dispositivos empotrados y móviles que se consideran en ISMED. Además, no se ha encontrado referencia a ninguna versión equivalente de P-Grid para dispositivos móviles.

Por otro lado, aunque más sencilla, la estructura del Tsc++ no podría ser usada en entornos móviles, dado que hace uso de la versión para Java SE de Jxta. Además, como más tarde se verá, dado que no existen repositorios semánticos para dispositivos móviles, la estructura de tsc++ no será fácilmente expandible en entornos móviles.

En la siguiente tabla se observan otras características de ambos proyectos. Mientras tsc++ hace uso de complejas y pesadas librerías, que sólo funcionan en Java SE, en el caso de TripCom se le suma el inconveniente de que además tiene una gran complejidad interna ya que está compuesto por siete componentes que deben ser arrancados manualmente (pudiendo hacerse uso de un gestor, pero no parece en cualquier caso que sea sencillo de iniciar de un modo más automatizado).

Todo ello nos hace pensar que pese a que TripCom es muy completo, probablemente se deba optar por usar un enfoque más semejante a tsc++. Pese a todo, tsc++ no parece ser suficientemente sencillo (necesita de mucha memoria y ancho de banda, como se reconoce en su propia guía de usuario [450]) para ejecutarlo en los tipos de dispositivos objeto dentro de ISMED.

2.1.2.7.4. Primitivas. Por lo general, las primitivas sobre triples spaces se podrían agrupar en las siguientes categorías:

- Write: almacena información en el repositorio RDF.
- Read: realiza una búsqueda en base a un patrón en todos los grafos devolviendo los grafos que contienen tripletas que cumplen dicho patrón.
- Take: es análogo a read, pero tras leer las tripletas, las elimina.
- Query: realiza una búsqueda en base a un patrón en todos los grafos devolviendo solamente las tripletas que lo cumplen.

	Tsc++	TripCom
Plataforma	JDK 1.5 o superior	JDK 1.5 o superior
Estándares soportados	RDF SPARQL (parcialmente mediante plantillas) TURTLE RDFXML N3 (para carga de ficheros)	RDF SPARQL (en el nivel extendido)
Librerías necesarias	Jxta (14MB) ▪ jxta.jar ▪ javax.servlet.jar ▪ org.mortbay.jetty.jar ▪ bcprov-jdk14.jar log4j (349KB) ▪ log4j.jar Yet Another Swing Library (23KB) ▪ slf4j-api-1.4.3.jar ▪ slf4j-log4j12-1.4.3.jar Sesame (1,35MB) ▪ openrdf-sesame-2.1.3-onejar.jar OWLIM (200KB) ▪ trree-3.0.beta7.jar ▪ owlim-3.0.beta7.jar	P-Grid Blitz
Tamaño de las librerias	16MB	¿?
Tamaño del núcleo	200KB	105MB
Componentes en los que se divide el núcleo	1	7

Tabla 2.8.: Comparativa de las principales implementaciones del paradigma del Triple Space Computing.

- Subscribe: para solucionar el problema de la diseminación de información se suelen usar el paradigma publish/subscribe. Así los suscriptores pueden expresar su interés por determinados datos suscribiéndose.
- Advertise: permiten publicar cierta información cumpliendo con el paradigma previamente mencionado.

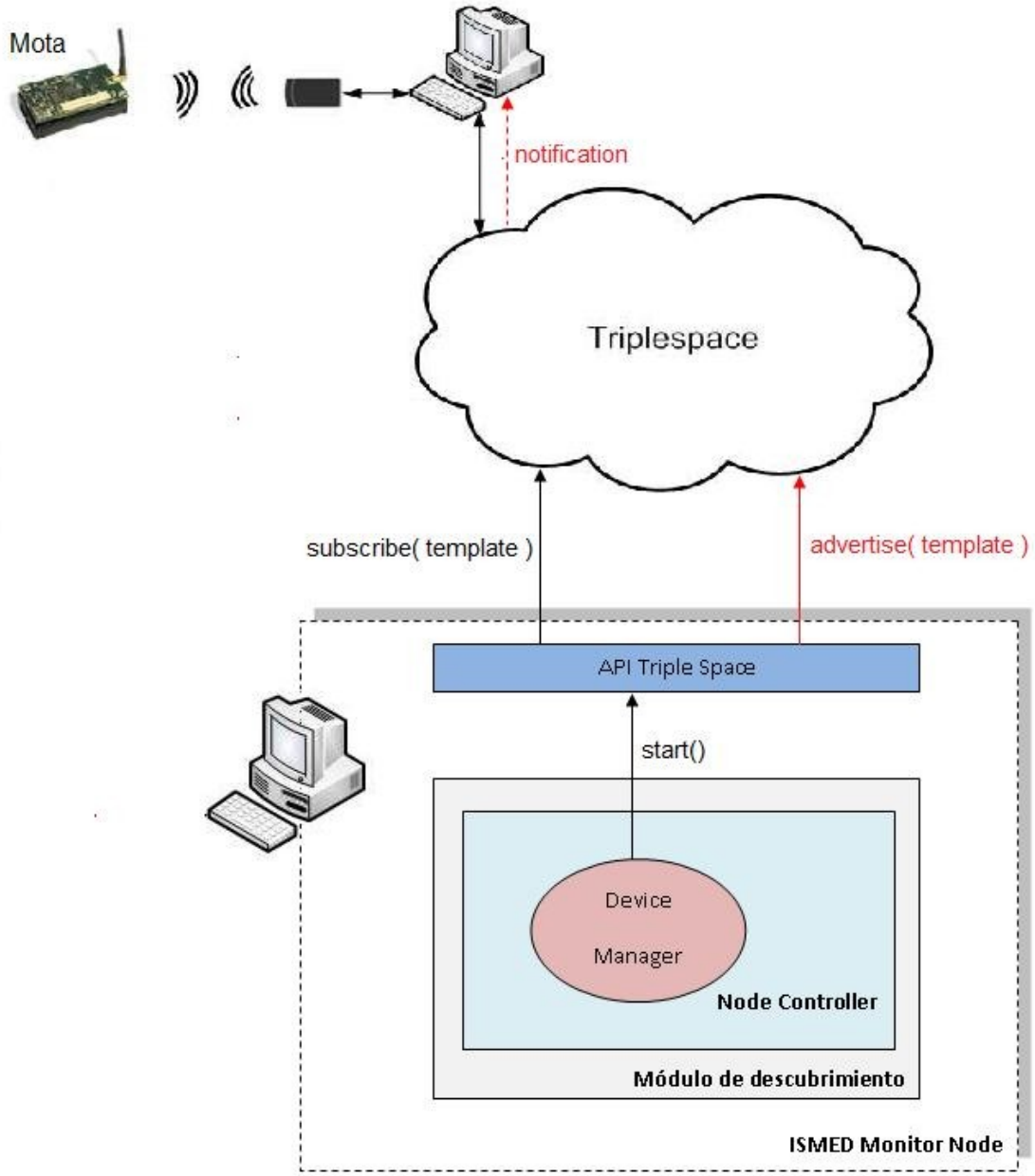


Figura 2.10.: Modelo de anuncio suscripción.

- Transacción: permite realizar transacciones sobre el triple space.

El proyecto TripCom divide su API en distintos niveles dependiendo de la expresividad que dichas operaciones permiten. Pese a que a priori podría considerarse que más expresividad es recomendable, también hay que tener en cuenta que ésta limita la escalabilidad del sistema.

Las primitivas usadas por tsc++ [450] y TripCom [600] se pueden resumir en la siguiente tabla.

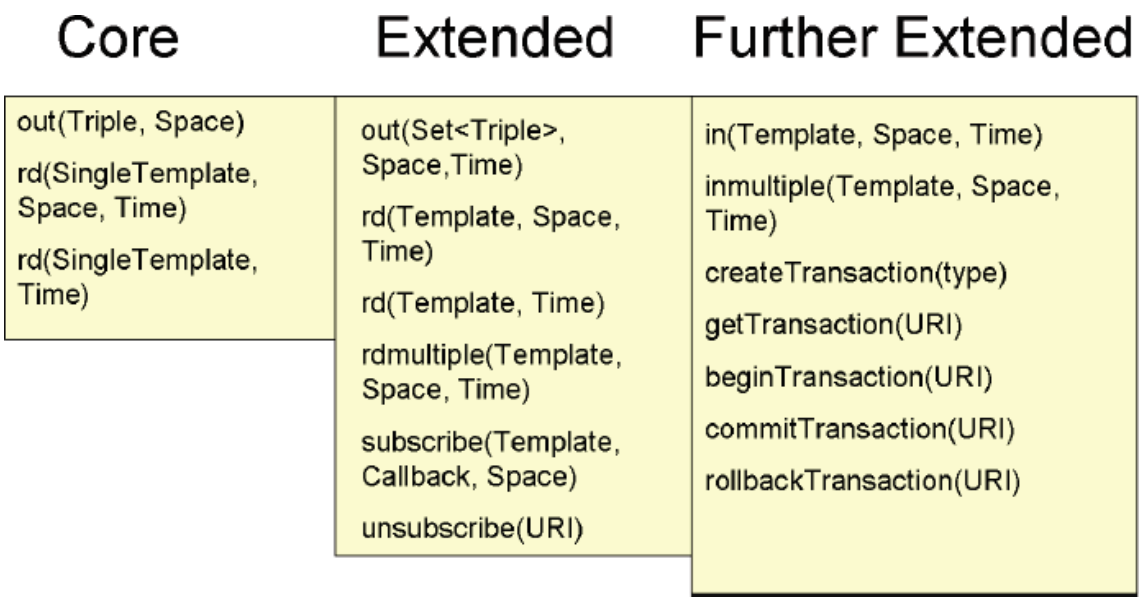


Figura 2.11.: División en distintos niveles de expresividad del API de TripCom.

Tabla 2.9.: Primitivas usadas por las principales proyectos de Triple Space Computing.

	Tsc++	TripCom
Write	URI write(URI space, Set<ITriple>triples) Hace que las tripletas se escriban en el espacio localmente. En ese momento otros kernels podrán acceder a dichas tripletas remotamente. Al escribir, el conjunto de tripletas se considera un grafo que se identifica con una URI. La ventaja reside en que las mismas tripletas crearán el mismo identificador.	void out(Triple t, URL space) void out(Set<Triple>t,URL space) Escribe atómicamente una o varias tripletas en el espacio.
Read	Set<ITriple>read(URI space, URI graph) Set<ITriple>read(URI space, ITemplate template) La diferencia con Query, es que sólo mira dentro de un grafo si se cumple el patrón especificado por template, y devuelve el grafo entero.	-
Take	Set<ITriple>take(URI space, URI graph) Set<ITriple>take(URI space, ITemplate template) Lee y elimina del repositorio.	Set<Triple>in(Template t, URL space, Time timeout) Set<Set<Triple>> inmultiple(Template t, URL space, Time timeout) Son operaciones paralelas a las explicadas en rd, pero se encarga de eliminar las tripletas después de leerlas.
Continúa en la siguiente página		

Continuación de la tabla		
	Tsc++	TripCom
Query	Set<ITriple>query(URI space, ITemplate template) Consulta tripletas en todos los grafos y devuelve las tripletas que lo cumplen. Combina tripletas buscadas remotamente con las buscadas localmente. Existe la posibilidad de indicarle que tan pronto como reciba los primeros resultados devuelva dichas tripletas (sin llegar a combinar nada).	Set<Triple>rd(SingleTemplate t, URL space, Time timeout) Set<Triple>rd(SingleTemplate t, Time timeout) Devuelve una tripleta que concuerda con el patrón facilitado (sin importar si existe más de una tripleta que cumpla el mismo). En la API extendida se pueden facilitar patrones de búsqueda más complejos (Template). Cuándo no se facilita el espacio, el sistema selecciona cualquier tripleta que coincida con el patrón sin importar el espacio en el que esté (en el apartado 2.1.2.8.3 se explica dicho proceso más detalladamente), siempre que el cliente tenga permisos de lectura. Set<Set<Triple>> rdmultiple(Template t, URL space, Time timeout) Devuelve todas las tripletas que cumplen un patrón dado en un espacio. La diferencia con "rd" es que rdmultiple devuelve más de un resultado.
Advertise	URI advertise(URI spaceURI, ITemplate template) void unadvertise(URI spaceURI, URI advertisement) Hace uso de los "advertisements" que facilita Jxta.	
Suscribe	URI subscribe(URI spaceURI, ITemplate template, INoticationListener listener) void unsubscribe(URI spaceURI, URI subscription) Dado un patron y una clase sobre la que se realizará la llamada de callback cuando dicho patrón haya cambiado, se suscribe al espacio definido por el parámetro spaceURI.	URI subscribe(Template t, Callback c, URL space) void unsubscribe(URI subscription) Se facilita un patrón y la clase a la que el sistema avisará en el momento en el que se encuentre una coincidencia para ese patrón.
Continúa en la siguiente página		

Continuación de la tabla		
	Tsc++	TripCom
Transacciones	-	URI createTransaction(String type) boolean getTransaction(URI transactionID) Se une a una transacción ya creada. boolean beginTransaction(URI transactionID) Todas las interacciones iniciadas por los agentes que compartan dicha transacción se realizarán "todas o ninguna". boolean commitTransaction(URI transactionID) boolean rollbackTransaction(URI transactionID)
	Tsc++	TripCom

Tanto en las operaciones "take" como "query" se puede apreciar que existe un último parámetro llamado **timeout**, que es el tiempo que el kernel deberá esperar a la resolución de la consulta antes de desbloquear al proceso que inició la consulta. Cómo se ha podido ver en el anterior apartado, existen distintos patrones, que ofrecen distintos niveles de expresividad a la hora de realizar consultas. A continuación se describen los mismos.

- Tsc++
  - Especifican las clausulas WHERE de las consultas SPARQL. Sustituyen el siguiente patrón por "?s, ?p, ?o".

```
CONSTRUCT {
  ?s, ?p, ?o
} WHERE {
  GRAPH <ts://espacio>  {  ?s, ?p, ?o  }
}
```

- TripCom
  - SimpleTemplate

    Es una tripleta que puede contener variables (semejante a la usada por tsc++).
  - Template

    Es una consulta genérica. Se pueden realizar consultas más complejas, como por ejemplo, consultas SPARQL.

Así, dados los siguientes triples almacenados en el espacio "ts://www.ejemplo.org/espacio":

```
@prefix rdf : <http:/ /www.w3 .org/1999/02/22rdfsyntaxns#> .

@prefix foaf : <http://xmlns.com/foaf/0.1/> .

<http:/ /members.sti2.at/membername> rdf:type foaf: Person .

<http://members.sti2.at/membername> foaf:name "Member Name" .

<http://members.sti2.at/membername> foaf:firstName "Member" .

<http://members.sti2.at/membername> foaf:phone <callto://membername> .

<callto://membername> rdf:type <http://skype.com> .
```

Ejemplos de consultas en tsc++:

- Query con patrón simple

```
<http://members.sti2.at/membername>

<http://www.w3.org/1999/02/22-rdf-syntax-ns#rstName>

?o.
```

Tanto tsc++ cómo en TripCom[1] devolverían todas aquellas tripletas que cumplen el patrón:

```
<http://members.sti2.at/membername>

<http://www.w3.org/1999/02/22-rdf-syntax-ns#rstName>

‘‘Member’’.
```

Además, si en TripCom se usase la primitiva “rd” y existiese más de una tripleta que cumpliese dicho patrón, el cliente que realizase las consultas sólo obtendría una tripleta como respuesta a cada llamada “rd”, que no necesariamente tendría porque ser distinta cada vez [797].

- Query con patrón complejo

Los ejemplos de consultas complejas, se corresponderían con el funcionamiento normal de las consultas SPARQL.

Por ejemplo, en la siguiente consulta se devolverían tripletas de la forma <nombre-persona de país>de aquellos participantes de simposios que existen.

[1] En el ejemplo, el resultado sería el mismo usando “rd” o “rdmultiple” porque sólo existe una tripleta que cumple dicho patrón.
En caso de usar la primitiva “rdmultiple” se devolverían todas las tripletas que lo cumpliesen. Si se optase por usar “rd”, se devolvería una sola tripleta por cada consulta realizada.

```
PREFIX ns: <http://example.com/any#>

CONSTRUCT {

?name ns:from ?country

} WHERE {

?x rdf:type ns:Symposium .

?x ns:hasParticipant ?p .

?p ns:name ?name; ns:resides ?country. }
```

- Read

```
<http://members.sti2.at/membername>

<http://www.w3.org/1999/02/22-rdf-syntax-ns#rstName>

?o.
```

En tsc++ se devuelve todo el grafo que fue escrito junto con la tripleta que cumple el patrón:

```
<http://members.sti2.at/membername>

<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

<http://xmlns.com/foaf/0.1/Person>.

<http://members.sti2.at/membername>

<http://xmlns.com/foaf/0.1/name>

‘‘Member Name’’.

<http://members.sti2.at/membername>

<http://xmlns.com/foaf/0.1/rstName>

‘‘Member’’.

<http://members.sti2.at/membername>

<http://xmlns.com/foaf/0.1/phone>

<callto://membername>.

<callto://membername>

<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

<http://skype.com>.
```

2.1.2.8 Tecnologías base para implementar el paradigma Triple Space Computing

2.1.2.8.1. Jena. Jena es una plataforma desarrollada por los laboratorios HP de Bristol. Este proyecto open- source provee soporte completo para inferencia en RDFS, soporte parcial para OWL y además permite que el usuario pueda crear motores de reglas personalizados.

2.1.2.8.2. Jxta. Jxta [218] es una plataforma peer-to-peer open source creada por Sun Microsystems que define un conjunto de seis protocolos basados en XML que permiten que dispositivos conectados a la red (no necesariamente tiene porque ser la misma red local) intercambien mensajes entre si. Los seis protocolos son asíncronos y están basados en un modelo de consulta/respuesta. A continuación se detallan dichos protocolos:

- *Peer Discovery Protocol (PDP).* Se usa por peers para publicitar sus propios recursos (peers, grupos, pipes o servicios) y descubrir los de otros peers.
- *Peer Information Protocol (PIP).* Se usa para obtener el estado de otros peers.
- *Peer Resolver Prototol (PRP).* Permite a los peers enviar consultas genéricas y recibir respuestas. Este protocolo permite definir e intercambiar información arbitraria que se necesitará. Pipe Binding Protocol (PBP). Establece un canal de comunicación virtual (un pipe), uniendo dos o más extremos de la comunicación.
- *Endpoint Routing Protocol (ERP).* Se usa para buscar rutas y puertos de destino en otros peers. La información de encaminamiento tiene una secuencia ordenada de identificadores, que se puede usar para enviar mensajes al destino.
- *Rendezvous Protocol (RVP).* Se usan para la resolución de recursos, propagar mensajes, publicar recursos locales y organizarse con otros peers en caso de ser rendezvous (aquellos peers que ayudan a otros con la propagación de mensajes).

Existen diversas implementaciones de Jxta en distintos lenguajes de programación y plataformas (c, symbian, java me, etc.) que permiten su funcionamiento en un amplio rango de dispositivos (ordenadores, teléfonos móviles, PDAs), logrando que se comuniquen de forma descentralizada.

2.1.2.8.3. P-Grid. P-Grid [614] es una plataforma para la gestión de la información de forma distribuida que permite una compartición de ficheros sencilla. Esta plataforma provee de una estructura descentralizada de P2P que no requiere coordinación central ni conocimiento mutuo previo de las entidades que la forman.

Entre las características más destacables de P-Grid cabe destacar su completa descentralización, su capacidad de organización propia, el balanceo de carga descentralizado que permite, las funcionalidades para la gestión de datos que incluye, la gestión de IPs dinámicas e identidades que implementa y la búsqueda eficiente que posibilita.

P-Grid se usa en el proyecto TripCom para llevar a cabo una indexación de los datos almacenados en cada nodo contenedor de conocimiento. En concreto, sirve para implementar las tablas de hash distribuido (Distributed Hash Tables, DHT).

Las DHT son una clase de sistemas distribuidos descentralizados que reparten la propiedad de un conjunto de claves entre los nodos que participan en una red, y son capaces de encaminar de forma eficiente mensajes al dueño de una clave determinada. Cada nodo es análogo a una celda de una tabla hash.

Así, al almacenar una tripleta en el TripleSpace de TripCom identificado por una URL, se crean los índices que apuntan a la URL donde finalmente se guardará la tripleta en distintos **host P-Grid** indicados por la función hash aplicada a los distintos elementos de la tripleta.

Más tarde, en caso de realizar una consulta sin indicar la URL del espacio en el que se desea buscar, se calculará la función hash sobre los parámetros que se especifiquen en dicho patrón, para encontrar el número del **host P-Grid** en cuya tabla de referencias se encuentra una referencia a la URL que contiene los datos pertinentes para realizar la consulta final.

Para comprenderlo mejor, se usará el ejemplo de [797].

- Dada la operación "out(<a,p,j>,URL)":

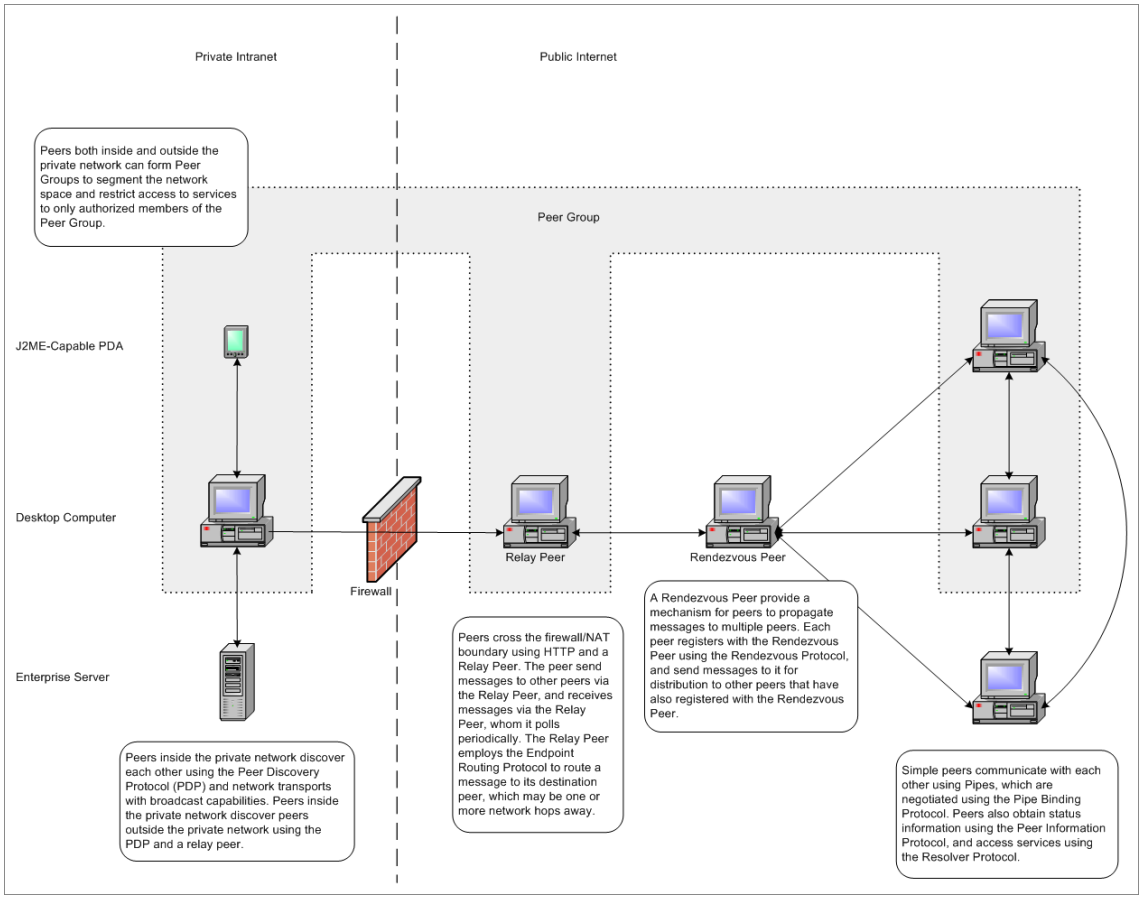


Figura 2.12.: Esquema del funcionamiento de los distintos tipos de elementos que pueden componer una red de Jxta.

- Se calcula el hash de a: h(a)=78
  - Se almacena la URL en la tabla de referencias del host P-Grid 78.
- Se calcula el hash de a: h(p)=45
  - Se almacena la URL en la tabla de referencias del host P-Grid 45.
- Se calcula el hash de a: h(j)=67
  - Se almacena la URL en la tabla de referencias del host P-Grid 67.
- Se almacena la tripleta <a,p,j>en la URL

Así, tras dicha lectura, las tablas de referencias almacenadas en los distintos hosts P-Grid comentados, podría ser:

Host P-Grid 78					Host P-Grid 45					Host P-Grid 45			
s	**p**	**o**	**TS URL**		**s**	**p**	**o**	**TS URL**		**s**	**p**	**o**	**TS URL**
a	p	j	tsa.com		a	p	j	tsa.com		a	p	j	tsa.com
b	q	m	tsc.com		c	r	n	tsb.edu					
					a	s	p	tsc.com					

- Si a continuación se realiza la operación rd(<a,?p,?o>).
  - Se realiza la función hash sobre a (el resto de elementos del patrón son variables), obteniéndose: h(a)=78
  - Luego, se obtiene del host P-grid 78, que para satisfacer la consulta se debe consultar en la URL del espacio "tsa.com". Realiza la consulta rd(<a,?p,?o>,URL)

2.1.2.8.4. Javaspace. JavaSpace [515] es una implementación en Java del paradigma de "Tuple Spaces" que forma parte de la tecnología Jini, que permite almacenar objetos de forma distribuida (persistente o no). De esta forma, se pueden almacenar tanto el estado del sistema como las implementaciones de algoritmos.

La API de JavaSpace que permite la manipulación de objetos "Entry" (aquellos que se pueden almacenar en el repositorio) consta de tres primitivas básicas:

- *Write,* que ubica una nueva entrada en el espacio.
- *Read,* que devuelve una copia de un objeto que concuerde con una consulta expresada con una plantilla determinada.
- *Take* - elimina un objeto que concuerde con una determinada plantilla y lo devuelve al objeto que realiza la llamada.

Las ventajas inherentes del uso de JavaSpace son la escalabilidad que se puede lograr gracias al procesamiento paralelo que posibilita y el almacenamiento confiable que permite el repositorio distribuido.

2.1.2.8.5. Blitz. El proyecto Blitz [649] es una implementación open source de JavaSpace, que trata de facilitar el desarrollo y despliegue de dicha tecnología de forma eficiente y compatible con Jini 2.x

Entre otras características destacables se encuentran las numerosas herramientas que ofrece para facilitar su gestión, la capacidad de personalizar la forma en la que se lleva a cabo la persistencia, indexación inteligente (usa almacenamiento en disco o en memoria logrando reducir el tiempo de búsqueda al mínimo), permite ver las entradas que hay en las distintas instancias del repositorio, posibilidad de uso de forma embebida para uso local en aplicaciones y facilidad de expansión.

2.1.2.8.6. Repositorios RDF. Los repositorios semánticos son sistemas similares a los sistemas gestores de bases de datos, ya que permiten almacenamiento, consultas y gestión de datos estructurados [6]. La principal diferencia entre ambos reside en el uso de ontologías como esquema semántico de las que hacen uso los repositorios semánticos y que permiten razonamiento. Además, las soluciones de almacenamiento suelen estar basadas en tecnologías de web semántica cómo RDF y SPARQL.

2.1.2.8.6.1. Sesame. Sesame es una plataforma RDF open source que permite realizar consultas (en SPARQL y SeRQL) e inferencias. Se puede desplegar sobre múltiples sistemas de almacenamiento (bases de datos relacionales, en memoria, etc.). En la Ilustración 2.13 se muestra las capas que componen Sesame entre las que cabe destacar:

- *Sail (Storage And Inference Layer),* que es un API de bajo nivel que abstrae de detalles de almacenamiento e inferencia.
- *Repository API,* que es un API de mayor nivel que ofrece métodos para subir archivos de datos, consultas, extracción y manipulación de datos.
- *HTTP Server,* que es un servlet de Java que implementa un protocolo para el acceso a los repositorios de Sesame mediante HTTP (es posible funcionar sin dicho servidor, pero en caso de querer hacerlo funcionar es necesario tener Tomcat instalado u otro contenedor de servlets).

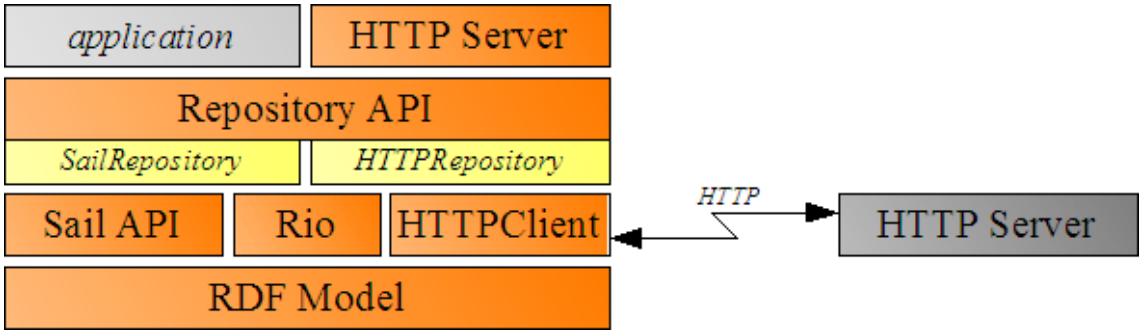


Figura 2.13.: Estructura de Sesame.

Sus características principales son:

- Su extensibilidad.
- Mínima sobrecarga de memoria (sólo la operación de escritura es directa)
- Consumo de memoria. Consultas y lecturas sobre información almacenada en disco duro.
- Crea una carpeta en la que almacena toda la información que gestiona.

Por otro lado, su mayor problema reside en el bajo rendimiento que logra al escribir grandes colecciones de triples.

2.1.2.8.6.2. Ontology Representation and Data Integration. ORDI [608] es un sistema de almacenamiento escalable, de alto rendimiento e independiente del lenguaje. Dicha plataforma, que es una extensión de Sesame, permite que se puedan acoplar distintos métodos que aseguren la persistencia de los datos y la capacidad de inferencia, mediante diversos adaptadores desarrollados a tal fin (por ejemplo, SesameAdapter, YARSAdapter o RDBMSAdapter) [716].

ORDI facilita:

- Integración posible de diferentes estructuras de datos incluidos sistemas de gestión de base de datos relacionales.

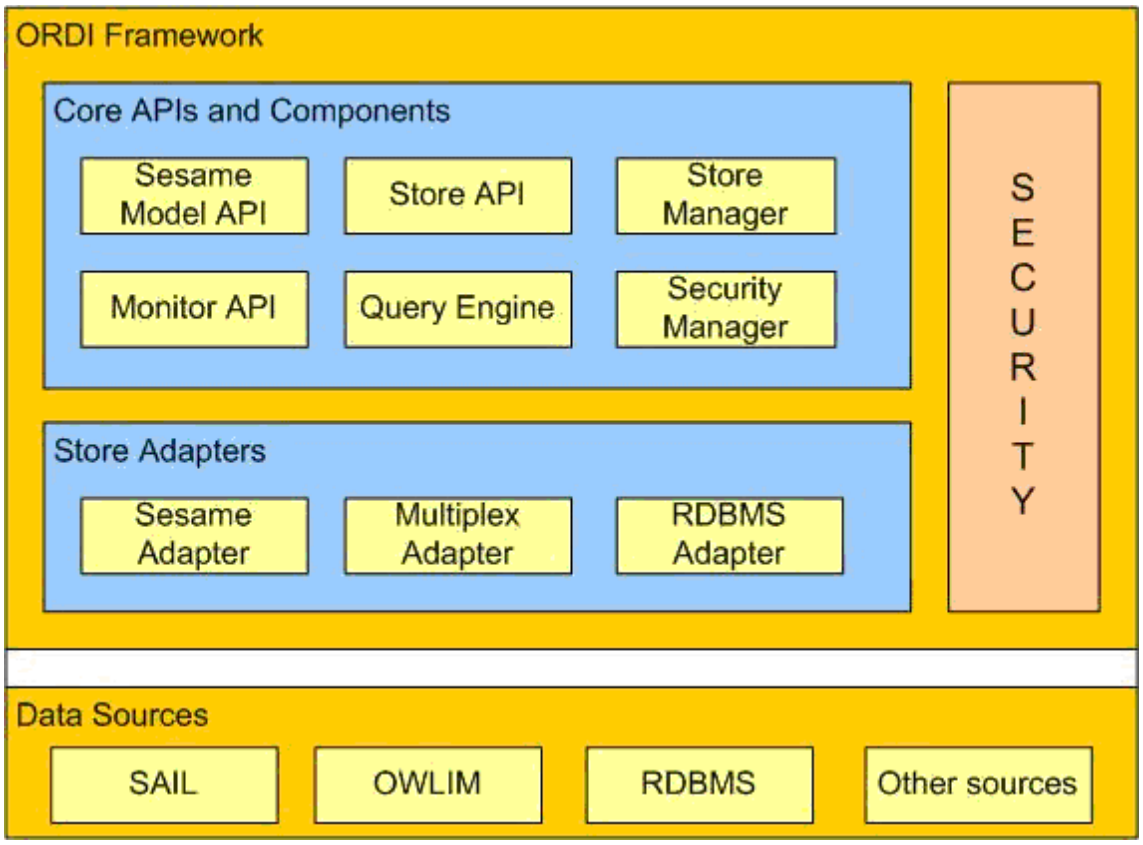


Figura 2.14.: Estructura de la Plataforma ORDI.

- Retrocompatiblidad con especificaciones RDF existentes y lenguaje de consultas SPARQL.
- Operaciones transaccionales sobre un modelo (en desarrollo).
- Proceso eficiente y almacenamiento de meta-datos o información contextual.
- Sentencias de grupo para gestionar grupos con los siguientes propósitos:
  - Definición de permisos de acceso y firmas.
  - Gestión de grupos de sentencias que corresponden a construcciones simples en lenguajes de alto nivel.
  - Gestión y rastreo (tracking) de transacciones.
- Sencilla gestión de datos desde distintas fuentes dentro de un mismo repositorio (o entorno computacional). Aquellos casos en el que se tienen datos importados de diferentes archivos (por ejemplo, diversas ontologías).

2.1.2.8.6.3. OWLIM. OWLIM [606] es un repositorio semántico que implementa la interfaz SAIL de Sesame, permitiendo la gestión, integración y análisis de datos heterogéneos, combinando ligeras capacidades de razonamiento. Se podría hacer la analogía de ver OWLIM como una base de datos RDF con alto rendimiento en el razonamiento. Existen dos tipos de soluciones (SwiftOWLIM y BigOWLIM) idénticas en API, sintaxis y lenguajes de consulta, pero que difieren en la su rendimiento.

Así, mientras SwiftOWLIM es bueno para experimentos y para cantidades de datos medias por su extrema rapidez en la carga de datos, BigOWLIM está diseñado para manejar grandes volúmenes de datos y consultas intensivas, de forma que posee una mayor escalabilidad con requisitos de memoria menores.

A continuación se ofrece una tabla comparativa con ambas soluciones:

	SwiftOWLIM	BigOWLIM
Escala (Millones de sentencias explícitas)	10 MSt, usando 1,6GB RAM 100 MSt, usando 16GB RAM	130 MSt, usando 1,6GB 1068 MSt, usando 8GB
Velocidad de procesamiento (carga+inferencia+almacenamiento)	30 KSt/s en un portátil 200 KSt/s en un servidor	5 KSt/s en un portátil 60 KSt/s en un servidor
Optimización de consultas	No	Sí
Persistencia	Back-up en N-Triples	Ficheros de datos binarios e índices
Licencia y disponibilidad	Open-source bajo LGPL; hace uso de SwiftTRREE que es gratuito pero no open-source.	Comercial. Las copias de evaluación o para investigación se facilitan de forma gratuita

OWLIM usa como lenguajes de consulta SPARQL, SeRQL, RQL y RDQL, y permite importar y exportar datos a XML, N3 y N-triples a través de Sesame.

La principal desventaja de N-Triples es su gran necesidad de memoria [450].

2.1.2.8.6.4. MemCached. Memcached [300] es un sistema de memoria caché distribuida de propósito general, originalmente desarrollado por Danga Interactive para el proyecto LiveJournal.

Su utilización se centra en la reducción del uso de la Base de Datos de grandes proyectos, implementando un sistema que se encarga de cachear accesos previos en una memoria distribuida.

Memcached utiliza una tabla distribuida de par clave/valor, a lo largo de toda la memoria configurada para tal uso. En caso de necesitar mayor cantidad de memoria de la configurada, se utilizan algoritmos LRU para eliminar la información que más tiempo lleva en la tabla.

Memcached permite almacenar en la memoria distribuida todo tipo de objetos y acceder a ellos desde cualquiera de las múltiples API existentes y se distribuye bajo licencia libre de Danga Interactive, Inc.[2]

[2]Memcached License. http://code.sixapart.com/svn/memcached/trunk/server/LICENSE

En la actualidad existen implementaciones de memcached en los siguientes lenguajes:

- Perl
- PHP
- Python
- Ruby
- Java
- C#
- C
- LUA

Aunque su uso se ha centrado en sitios web con gran cantidad de visitas y requisitos técnicos, su implementación de memoria distribuida puede usarse con propósito general.

Tal y como indican en la web existen numerosos proyectos de gran envergadura que utilizan dicha solución, entre los que cabe destacar: Youtube, LiveJournal, Slashdot, Wikipedia, SourceForge, Facebook, Digg, Twitter y NYTimes.com[3].

2.1.2.8.6.5. YARS. YARS (Yet Another RDF Repository) un repositorio semántico que se ejecuta a modo de servicio web en un servidor web Tomcat. YARS permite agregar información, recuperarla o eliminarla mediante simples consultas HTTP. Los resultados se devuelven en formato NTriples y las consultas se pueden realizar de forma más expresiva que usando simples patrones con la forma <sujeto,predicado,objeto>, mediante el lenguaje de consultas N3QL.

2.1.2.8.6.6. Otros. Pese a que no han sido utilizados en proyectos relacionados con el paradigma Triplespace, además de los analizados, existen otros repositorios semánticos que se listan a continuación:

- 3Store [678], es un almacen de tripletas RDF, escrito en C que usa MySQL y BerkeleyDB, diseñado para un manejo eficiente de bases de conocimiento RDF. Este repositorio permite el acceso a los datos RDF vía RDQL o SPARQL usando http, una interfaz de comandos o un API de C.
- RDFPeers [556], que es un repositorio de tripletas distribuido, donde los datos se guardan a través de un largo número de nodos, de forma que si uno de dichos nodos falla, la información aún es accesible dado que está replicada en otros nodos de la red. Pese a ello RDFPeers no es un repositorio semántico al uso, dado que los repositorios guardan fracciones de todos los datos, guardando índices del sujeto, predicado y objeto de las tripletas que almacenan.
- Kowari, que tiene soporte nativo de RDF y un lenguaje de consultas tipo SQL.
- TAP, que es un sistema que permite almacenamiento distribuido y búsqueda semántica.

2.1.2.8.7. Comparativa y adaptación a entornos móviles y empotrados. Sesame y Owlim [460] son librerías que pueden integrarse perfectamente en aplicaciones completamente hechas en java como tsc++. Pese a ello, y tal y como se puede ver a continuación, tienen requisitos computacionales excesivamente elevados:

2.1.3 Dispositivos Móviles

El objetivo principal del módulo de modelado y coordinación semántica es adaptar y trasladar el paradigma TSC analizado en el epígrafe anterior para que funcione en dispositivos empotrados y móviles con capacidad de cómputo, almacenamiento y memoria reducida.

[3] Who's using Memcached. http://www.danga.com/memcached/users.bml

	OWLIM 5	Sesame	ORDI
Última versión	3.0 beta5	2.2.1	0.5 alpha2
Lenguaje en el que están programadas	JDK 1.4.2	JDK 1.5	JDK 1.5
Consumo de memoria			

	OWLIM 5	Sesame	ORDI
Escala (Millones de sentencias explícitas)	10 MSt, usando 1,6GB RAM 100 MSt, usando 16GB RAM		
Velocidad de procesamiento (carga+inferencia+almacenamiento)	30 KSt/s en un portátil 200 KSt/s en un servidor		
Librerías de las que hace uso	trree-3.0.beta7.jar owlim-3.0.beta7.jar	openrdf-sesame-2.2.1-onejar.jar	
Espacio que ocupan las librerías	200KB	1,55MB	
Motor de razonamiento	TRREE 2.9.1		TRREEAdapter
Estándares	RDF, OWL DLP, y OWL Horst.	RDF, RDF Schema, OWL	Especificaciones RDF
Lenguajes de consulta	SeRQL, RQL y RDQL (SPARQL desde la v3.0b7)	SeRQL y SPARQL	SPARQL
Persistencia	Back-up en N-Triples	- Basado en memoria - Basado en disco (native store) - Usando RDBMS - Usando un repositorio HTTP	Múltiples opciones: tantos como tipos de adaptadores existan implementados.
Licencia y disponibilidad	Open-source bajo LGPL; hace uso de Swift-TRREE que es gratuito pero no open-source.	BSD	

2.1.3.1 Java ME

Java ME, anteriormente llamada J2ME, es una plataforma formada por un subconjunto de la plataforma Java que tiene como propósito proveer un conjunto de APIs para el desarrollo de software en dispositivos con capacidades limitadas. Dichos dispositivos pueden ser móviles, PDAs o Set-Top Boxes (STB) entre otros. Esta plataforma está dividida en dos configuraciones (CLDC y CDC). Cada una de estas configuraciones puede ser extendida mediante perfiles, los cuales definen los requisitos mínimos que los dispositivos deben cumplir.

Además de las configuraciones y perfiles, esta plataforma dispone de extensiones definidas mediante JSRs. Algunos ejemplos de estas extensiones son: Bluetooth (JSR-82), WMA 1.0 (JSR-120) y Web Services (JSR-172) y FileConnection (JSR-75).

En este punto se van a explicar las diferentes configuraciones y perfiles de Java ME. También se realizara una pequeña introducción a alguna de sus extensiones.

2.1.3.1.1. CLDC. CLDC (Connected Limited Device Configuration) está formado únicamente por el subconjunto mínimo de la librería de clases de Java necesario para que una máquina virtual pueda ejecutarse. Por ello es la configuración más adecuada para dispositivos de capacidades muy limitadas como teléfonos móviles y PDAs. Esta configuración esta especificada en los JSR 30 (CLDC 1.0) y 139 (CLDC 1.1). Como anteriormente se ha comentado, las configuraciones pueden ser ampliadas mediante perfiles. Los perfiles soportados por CLDC son MIDP e IMP.

2.1.3.1.1.1. MIDP. MIDP (Mobile Information Device Profile) es una especificación de perfil destinada a dispositivos embebidos tales como teléfonos móviles y PDAs. Este perfil está definido en los JSR 37 (MIDP 1.0) y 118 (MIDP 2.0). Actualmente se está definiendo el JSR 271 (MIDP 3.0).

La versión 1.0 de este perfil añade clases para el manejo de operaciones de entrada/salida, diseño de interfaces de usuario y almacenamiento persistente. Además añade la clase MIDLet.

La versión 2.0 extiende la librería de interfaces de usuario añadiendo soporte para el desarrollo de juegos e incluye nuevas funcionalidades que permiten crear aplicaciones multimedia y realizar conexiones seguras.

2.1.3.1.1.2. IMP. IMP (Information Module Profile) es una especificación de perfil destinada a dispositivos embebidos equipados con capacidades de visualización muy limitadas. Este perfil está definido mediante los JSR 195 (IMP 1.0) y 228 (IMP-NG).

La versión 1.0 de este perfil añade clases para el manejo de operaciones de entrada/salida y almacenamiento persistente. Además añade la clase MIDLet.

La versión NG incorpora el manejo de más tipos de redes y seguridad en éstas entre otras cosas.

2.1.3.1.2. CDC. CDC (Connected Device Configuration) está formado por casi toda la librería de clases exceptuando las relacionadas con la interfaz de usuario. Debido a esto, esta configuración requiere dispositivos con más capacidades que la configuración anteriormente explicada (CLDC), STB's por ejemplo. Esta configuración está especificada en los JSR 36 (CDC 1.0).

Los perfiles soportados por esta configuración son Foundation, Personal Basis y Personal.

2.1.3.1.2.1. Foundation Profile. Este perfil está diseñado para dispositivos que requieran implementación completa de la máquina virtual de Java así como una casi completa librería de clases de Java.

2.1.3.1.2.2. Personal Basis Profile. Personal Basis Profile es una extensión de Foundation Profile. Este perfil añade un subconjunto ligero de AWT.

2.1.3.1.2.3. Personal Profile. Personal Profile es una extensión de Personal Basis Profile. Este perfil soporta AWT completamente a diferencia de Personal Basis y además añade soporte para applets.

2.1.3.1.3. Extensiones. Como anteriormente se ha comentado, Java ME puede ampliarse mediante una serie de extensiones. Las extensiones más interesantes para este proyecto son las comentadas en los siguientes puntos.

2.1.3.1.3.1. WMA. WMA (Wireless Messaging API) es una extensión opcional de Java ME que permite acceder independientemente de la plataforma a servicios wireless tales como el envío de SMS. Esta extensión está especificada en los JSR 120 (WMA 1.0) y 205 (WMA 2.0).

2.1.3.1.3.2. WSA. WSA (Web Services API) permite la creación de clientes de Servicios Web proveyendo de un modelo de programación que cumple los estándares de Servicios Web. Esta extensión está especificada en el JSR 172.

Esta extensión puede ser utilizada tanto en CDC como en CLDC.

2.1.3.1.3.3. FileConnection. Esta extensión, especificada en el JSR 75, opcional permite acceder al sistema de ficheros de los dispositivos y a las tarjetas de memoria montadas en el mismo. Esta extensión puede ser utilizada tanto en CDC como en CLDC.

2.1.3.2 Librerías de interés para implementar Triple Space Computing

2.1.3.2.1. Motores de inferencia para Sistemas Embebidos. Existen numerosos motores de inferencia que funcionan en entornos donde los recursos computacionales son altos. Desgraciadamente, prácticamente no existen motores de inferencia para Sistemas Embebidos, destacando entre ellos el desarrollado por Iñaki Vázquez en su Tesis [826]. En la tesis se describen las características del denominado MiniOwlReasoner, el cual no es un razonador de ontologías completo, sino que es un razonador limitado, pero a la vez potente para entornos de Inteligencia Ambiental. Para realizar el razonamiento el motor de inferencia filtra las ontologías para seleccionar solamente las construcciones que es capaz de manipular, las cuales son las siguientes:

- rdfs:subClassOf
- owl:sameAs
- owl:TransitiveProperty
- owl:SymmetricProperty
- owl:inverseOf

Las construcciones mostradas en la lista anterior son los predicados más empleados para crear relaciones entre los conceptos de las ontologías. El razonador MiniOwlReasoner puede procesar cualquier tipo de característica transitiva o simétrica, que pese a ser poco, intrínsecamente aportan mucha inteligencia a cualquier ontología. El autor resalta que el razonador desarrollado implementa un pequeño conjunto de OWL Lite, pero que es más complejo y potente que RDF++ de Lassila [468] y equivalente a OWL Tiny [112].

2.1.3.2.2. Microjena. MicroJena [237] es una versión reducida de la API Jena que fue diseñada para permitir el diseño y la implementación de aplicaciones de web semántica en dispositivos móviles.

Microjena cuenta entre sus ventajas, el requerir un menor espacio en memoria (384 KBs frente a los 16MBs de Jena), la mayor velocidad que se alcanza (que se hace más palpable cuanto mayor sea la complejidad de los modelos) y la retrocompatibilidad con Jena que permite desarrollar de una forma similar a dicha API ahorrando de dicha manera tiempo de adaptación para los usuarios de Jena.

Entre sus problemas más destacables, se encuentran:

- Microjena excluye mecanismos de inferencia.
    - Permite realizar consultas sobre un modelo, realizando un filtrado y devolviendo las tripletas que cumplen dicho patrón, pero sin realizar ningún razonamiento. enditemize

- No tiene motores ARQ, de forma que no permite utilizar SPARQL, ni ningún otro lenguaje de consulta sobre el modelo semántico.
- No posee una forma de almacenar los modelos semánticos con los que trabaja, ni en base de datos, ni en ningún otro tipo de repositorios.
- El único formato en el que es capaz de exportar el modelo es en N-Triples.

Los creadores de Microjena proponen como solución a la limitación relacionada con la incapacidad de Microjena para inferir, el uso de un razonador DIG, aunque sin tener la certeza de la validez de dicha solución.

Las interfaces DIG proveen de un mecanismo neutral para el acceso a la funcionalidad que ofrece un razonador de lógica descriptiva. A alto nivel, el mecanismo consiste en mensajes XML enviados al razonador por conexiones HTTP y las respuestas que éste facilita.

Ante la incapacidad para almacenar el modelo en los dispositivos móviles, en posteriores epígrafes se analizarán sistemas para realizar esta tarea de la mejor forma posible.

2.1.3.2.3. JXME. JXME [17] es una implementación del protocolo Jxta para dispositivos móviles que funcionen con Java Micro Edition.

2.1.3.2.3.1. Con proxy. En la versión 2.1.3 de Jxme para dispositivos móviles con Java ME CLDC, que data del 2006, se hacía necesario operar con un relay que ejercía además de proxy para comunicarse con el resto de nodos de la red.

El uso de un proxy/relay en dispositivos móviles venía motivado por las capacidades de cómputo más limitadas que tenían los móviles hace años, que imponían las siguientes restricciones:

- La memoria necesaria para poder parsear los mensajes XML que constituyen el protocolo Jxta.
- La memoria necesaria para cachear el estado de la red.
- Los peers Jxta escuchan a nivel de socket y datagrama para encontrar información sobre las redes, sin embargo los dispositivos móviles sólo podían hacer uso del protocolo HTTP por defecto. Los sockets y conexiones datagrama de las que hace uso Jxta son opcionales.

Por lo tanto, dado Jxta excedía los requisitos mínimos de MIDP, era necesaria la figura del relay que se encargue de realizar esas tareas que los dispositivos no eran capaces de realizar.

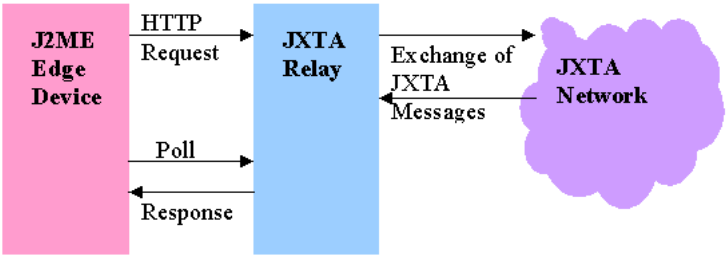


Figura 2.15.: Coordinación entre dispositivos J2ME y Relays JXTA.

2.1.3.2.3.2. Sin proxy. En la versión 2.5 de Jxme (2008) para dispositivos móviles, no se necesita hacer uso de relays que traduzcan los mensajes de los móviles a mensajes estándar de Jxta. En cualquier caso, los móviles deben comunicarse con un Rendezvous encargado de propagar sus mensajes al resto de nodos del grupo (por no tener Jxme implementado ningún mecanismo de multicast).

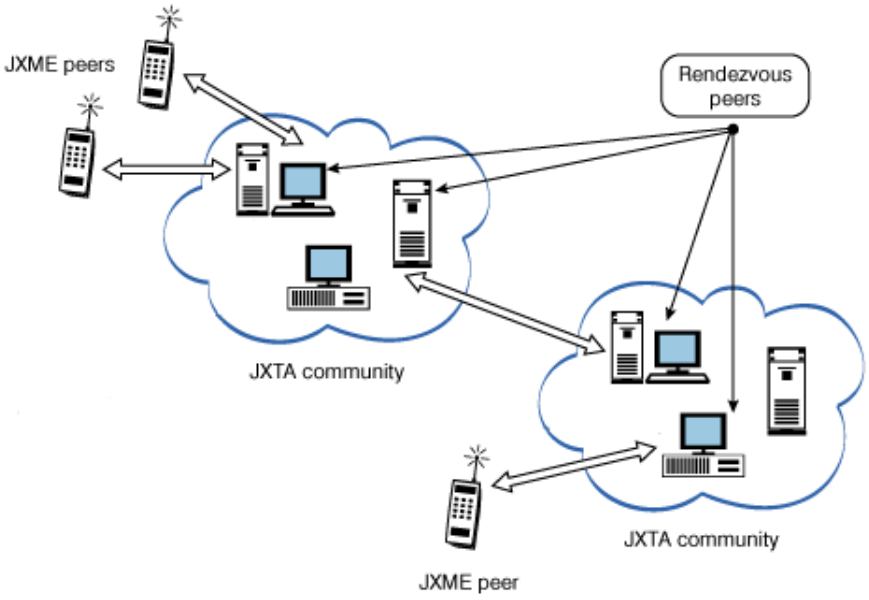


Figura 2.16.: Esquema del funcionamiento de peers JXME con el resto de peers de Jxta.

2.1.3.2.4. MobileSpaces. MobileSpaces [703] parece ser la única implementación de Javaspaces existente para dispositivos móviles. Esta librería trata de permitir a las aplicaciones para dispositivos móviles acceder a "tuple space", de la forma más semejante posible a como se realiza desde otras plataformas.

El sistema creado se basaba en tres componentes principales:

- Javaspace como implementación del paradigma tuple space.
- La tecnología Jini, que agrupa servicios (como se ha explicado previamente JavaSpace es considerado un servicio de Jini) y facilita el acceso a los mismos.
- La arquitectura Surrogate de Jini, que permite a dispositivos limitados acceder a los servicios de Jini.

Por su parte, esta librería está compuesta de dos elementos principales:

- Surrogate, que ejerce de proxy entre el dispositivo y JavaSpaces. Así, gestiona el acceso al espacio y envía y recibe objetos java, dado que Java ME es incapaz de ello debido a sus limitaciones.
- Manejar situaciones resultantes del funcionamiento común de los dispositivos móviles (como llamadas telefónicas, que pausarían la aplicación)

De esta forma, MobileSpaces permitía a los móviles convertirse en usuarios de JavaSpaces.

El aspecto negativo es la escasez de documentación que existe al respecto de MobileSpaces, y sobre todo, el hecho de que la implementación del sistema no parece haberse publicado en ningún lugar.

2.1.3.2.5. Persistencia de datos y RDF. Cómo se ha explicado en el epígrafe 2.1.2.8.6, los repositorios semánticos no sólo sirven para almacenar modelos semánticos de forma persistente (como podrían hacer los sistemas de gestión de bases de datos), sino que además permiten realizar inferencia sobre dichos modelos. Desgraciadamente, como se ha explicado anteriormente los motores de inferencia existentes para dispositivos embebidos son anecdóticos, por lo que es aún más complejo encontrar estos motores en conjunción con un mecanismo de persistencia. Es por ello, que no se ha tenido constancia de la existencia de ningún repositorio semántico.

Para suplir esta deficiencia, se podrían usar repositorios normales sin capacidades semánticas.

La solución más común bajo la plataforma Java ME CLDC es RecordStore, que es una base de datos de registro que se caracteriza por su gran consumo de tiempo y su capacidad para agotar la batería del dispositivo. Además, las búsquedas que se pueden realizar sobre él, son muy limitadas.

Existen numerosas abstracciones sobre RecordStore como son OpenBaseMovil, Perst Lite, Floggy, J2MEMicroDB o la capa de persistencia facilitada por la librería Java Polish. Además, existen distintos drivers que permiten acceder a servidores de bases de datos con funcionalidades no limitadas mediante una interfaz similar a JDBC, como puede ser Mimer.

Para CDC existen soluciones más sofisticadas como puede ser JavaDB (basado en apache Derby). JavaDB necesita de 256KBs de ROM y RAM, el perfil "Foundation Profile" (JSR-219) y el paquete opcional de JDBC para plataformas Java ME JSR-169.

2.1.4 Plataformas empotradas

A continuación se especificarán las capacidades, programabilidad, coste y capacidades de comunicación de las plataformas empotradas más relevantes susceptibles de ser usadas en el proyecto.

2.1.4.1 Gumstix.

Gumstix [13] es una empresa dedicada a la construcción de ordenadores miniaturizados completamente funcionales. La línea de productos de Gumstix está formada tanto por ordenadores empaquetados como por placas base sueltas.

Esta plataforma [14] está formada por una placa base, equipada con un procesador ARM, a la que se le puede añadir placas de expansión que ofrecen funcionalidades adicionales, tales como red Ethernet, wireless, lector de tarjetas, etc. Estos ordenadores funcionan con una distribución de GNU/Linux embebido llamada OpenEmbedded.

Actualmente existen las siguiente familias de placas base: Basix, Connex, Verdex, Verdex Pro y Overo Earth. Las diferencias entre todas estas placas base están especificadas en las siguientes tablas.

	Basix 200	Basix 400 xm	Basix 400 xm-bt
Procesador	Intel XScale PXA255 a 200 MHz	Intel XScale PXA255 a 400 MHz	Intel XScale PXA255 a 400 MHz
Mem. RAM	64 MB	64 MB	64 MB
Mem. Flash	64 MB	16 MB	16 MB
Tarjetas de memoria	MMC	MMC	MMC
Slots	1 x 60 pin	1 x 60 pin	1 x 60 pin
Conexión			Bluetooth
Precio	99 $	129 $	169 $

Tabla 2.10.: Comparativa de los distintos productos la línea Basix.

2.1.4.1.1. Programabilidad. Al disponer de un sistema operativo GNU/Linux, se pueden desarrollar aplicaciones para dispositivos Gumstix en diferentes lenguajes y plataformas.

Los lenguajes más interesantes en los que se pueden desarrollar aplicaciones para esta plataforma son los siguientes:

- C/C++: Se pueden desarrollar todo tipo de aplicaciones, librerías, módulos del núcleo, etc.
- Java: Es posible instalar JamVM y GNU Classpath. Dependiendo de la versión del sistema operativo, la versión de Java será 1.4 o 5.0
- Python: Es posible instalar Python 2.5
- Ruby: Es posible instalar Ruby 1.8

	Connex 200 xm	Connex 400 xm	Connex 400 xm-bt
Procesador	Intel XScale PXA255 a 200 MHz	Intel XScale PXA255 a 400 MHz	Intel XScale PXA255 a 400 MHz
Mem. RAM	64 MB	64 MB	64 MB
Mem. Flash	64 MB	16 MB	16 MB
Slots	1 x 60 pin, 1 x 92 pin	1 x 60 pin, 1 x 92 pin	1 x 60 pin, 1 x 92 pin
Conexión			Bluetooth
Precio	109 $	129 $	169 $

Tabla 2.11.: Comparativa de los distintos productos la línea Connex.

	Verdex xm4	Verdex xm4-bt	Verdex x16p
Procesador	Marvell PXA270 con XScale a 400 MHz	Marvell PXA270 con XScale a 400 MHz	Marvell PXA270 con XScale a 600 MHz
Mem. RAM	64 MB	64 MB	128 MB
Mem. Flash	16 MB	16 MB	32 MB
Slots	1 x 60 pin, 1 x 120 pin, 1 x 24 pin flexible	1 x 60 pin, 1 x 120 pin, 1 x 24 pin flexible	1 x 60 pin, 1 x 120 pin, 1 x 24 pin flexible
Conexión		Bluetooth	
Otras características	USB Host signals, CCD Camera signals	USB Host signals, CCD Camera signals	USB Host signals, CCD Camera signals
Precio	129 $	159 $	169 $

Tabla 2.12.: Comparativa de los distintos productos la línea Verdex.

	Verdex pro xm4	Verdex pro xm4-bt	Verdex pro x16p
Procesador	Marvell PXA270 con XScale a 400 MHz	Marvell PXA270 con XScale a 400 MHz	Marvell PXA270 con XScale a 600 MHz
Mem. RAM	64 MB	64 MB	128 MB
Mem. Flash	16 MB	16 MB	32 MB
Tarjetas de memoria	Micro SD	Micro SD	Micro SD
Slots	1 x 60 pin, 1 x 80 pin, 1 x 24 pin conector flexible y plano	1 x 60 pin, 1 x 80 pin, 1 x 24 pin conector flexible y plano	1 x 60 pin, 1 x 80 pin, 1 x 24 pin conector flexible y plano
Conexión		Bluetooth	
Otras características	USB Host signals, CCD Camera signals	USB Host signals, CCD Camera signals	USB Host signals, CCD Camera signals
Precio	129 $	159 $	169 $

Tabla 2.13.: Comparativa de los distintos productos de la línea Verdex pro.

	Overo Earth
Procesador	Procesador de aplicaciones OMAP 3503 con CPU ARM Cortex-A8 a 600 MHz
Mem. RAM	256 MB DDR de bajo consumo
Mem. Flash	256 MB NAND
Tarjetas de memoria	Micro SD
Slots	2 x 70 pin, 1 x 27 pin conector flexible y plano
Otras características	Soporte para placas base "OMAP 35x-based Overo", bus I2C, 6 x líneas PWM, 6 x A/D, 1-wire, UART, entrada de cámara, entrada de micrófono, salida de auriculares, entrada batería de reserva, USB OTG signal, USB HS host
Precio	149 $

Tabla 2.14.: Características de Overo Earth.

2.1.4.2 SunSpot.

Sun SPOT [24] es una mota de una red de sensores inalámbricos desarrollada por Sun Microsystems. Estas motas están desarrolladas para hacer uso del estándar IEEE 802.15.4, sobre el cual implementan la capa MAC, sobre la que se puede desplegar ZigBee.

A diferencia de otras motas de redes de sensores inalámbricas. Las Sun SPOT tienen soporte para aplicaciones Java debido a la máquina virtual que utilizan, la Squawk Virtual Machine. Las características de las Sun SPOT son las siguientes:

	Sun SPOT
Procesador	ARM920T de 32 bit a 180 MHz
Mem. RAM	512 KB
Mem. Flash	4 MB
Conectores	USB
Conexión	IEEE 802.15.4 con antena integrada
Sensores	Acelerómetros en 3 ejes 2G/6G, temperatura, luminosidad, 8 LED tricolor
Otras características	6 entradas analógicas, 2 botones, 5 pins de entrada/salida de propósito general, 4 pins de salida
Precio	630 € por Kit de desarrollo

Tabla 2.15.: Características de Sun SPOT.

2.1.4.2.1. Programabilidad. Las Sun SPOT únicamente pueden ser programadas en Java, más concretamente en Java ME. Actualmente, la forma más sencilla es mediante el IDE Netbeans con la ayuda del plugin de desarrollo para esta plataforma.

2.1.4.2.2. Squawk Virtual Machine. Squawk Virtual Machine [596] es una máquina virtual de Java (JVM) para sistemas embebidos. A diferencia de la mayoría de las JVMs, que están escritas en C/C++ y ensamblador, Squawk está desarrollada casi completamente en Java, lo cual aumenta su portabilidad.

Las características principales de esta JVM son: uso de memoria reducido, provee de un mecanismo de aislamiento por el cual una aplicación es representada como un objeto y es capaz de ejecutar más de una aplicación en una única instancia de la máquina virtual.

2.1.4.3 Millennial.

Millennial [19] ofrece hardware y software para el desarrollo de WSN de alta calidad y bajo consumo. El rango de aplicaciones que pretende abarcar es muy amplio, como por ejemplo la automatización de la industria, sistemas de seguridad y aplicaciones de baja transferencia de datos, en general.

La plataforma MeshScape proporciona un gran rendimiento en base a una gran escalabilidad, fiabilidad, capacidad de respuesta y eficiencia energética. Utiliza IEEE 802.15.4 como estándar de comunicación. Los módulos de los que se compone la plataforma MeshScape son:

- *MeshGate:* es el gateway de la red recibiendo información de ella, configurando sus parámetros y puede actuar como un monitor de red.
- *End nodes:* están integrados con sensores y actuadores para capturar la información del entorno.

- *Mesh nodes:* extiende la cobertura de la red encaminando los mensajes, proporcionando rutas de backups en caso de congestión de red o fallos. Los Mesh nodes se puede integrar directamente con los sensores y actuadores en configuraciones Mesh o de estrella.

Figura 2.17.: Módulo de Millenial.

2.1.4.3.1. Programabilidad. Proporciona un kit de desarrollo que incluye todo lo necesario (software, hardware y accesorios) para crear una sencilla red de sensores wireless.

2.1.4.4 Crossbow Motes.

Crossbow [9] nació en el año 1995 y ha sido, y sigue siendo, referencia en el desarrollo de sistemas de redes de sensores inalámbricas. Sus productos son conocidos como *Motes*, y en general son plataformas formadas por un microcontrolador y un transmisor. Utilizan un sistema operativo llamado *TinyOS*, de licencia libre, que permite la administración de tareas y eventos con un bajo consumo de energía.

Desde su creación, Crossbow ha desarrollado varios modelos de *Motes*: WeC, René, René2 o Mica. Actualmente, es posible adquirir varios modelos entre los que destacan:

- Micaz: plataforma wireless sobre IEEE 802.15.4/Zigbee de majo consumo, en la que todos los nodos tiene capacidad de enrutamiento. Trabaja sobre TinyOS.
- Mica2: plataforma wireless que utiliza un transceiver radio a 868/916 MHz, de bajo consumo y en la que todos los nodos tiene capacidad de enrutamiento. Trabaja sobre TinyOS.
- iMote2: esta plataforma trabaja sobre el Microframework de .NET, implementa comunicación IEEE 802.15.4/Zigbee y dispone de múltiples interfaces para comunicación con sensores.

2.1.4.4.1. Programabilidad. Tanto las motas Mica2 como Micaz, se pueden programar en nesC, mientras que iMote2 usa C# micro framework.

Además, cabe destacar que Crossbow ofrece kits de desarrollo de diferentes características, en función del desarrollo a realizar:

- **WSN Starters Kit:** es un kit orientado al desarrollo de redes simples de sensores. Incluye pocas unidades de motes.
- **WSN Profesional Kit:** es un kit con 8 motes, y se puede orientar al estudio de redes de sensores más complejas.
- **WSN OEMS Design Kit:** permite el desarrollo de prototipos de forma rápida.
- **WSN Imote2 .Builder Kit:** es uno de los últimos productos de Crossbow, basado en la plataforma .NET de Microsoft. Se reduce la complejidad a la vez que incrementa la productividad.

Figura 2.18.: Nodos sensores de Crossbow.

	Micaz	Mica 2	iMote2
Procesador	Atmel ATmega128L	Atmel ATmega128L	Intel PXA271 XScale® Processor at 13 – 416MHz
Mem. RAM	-	-	256kB SRAM 32MB SDRAM
Mem. Flash	Programa: 128 KB Medidas: 512 KB	Programa: 128 KB Medidas: 512 KB	32MB FLASH
Conectores	En él se pueden conectar una amplia variedad de sensores y placas que permiten la adquisición de datos.	Igual que en Micaz.	Igual que en Micaz.
Conexión	802.15.4 / ZigBee	802.15.4 / ZigBee	802.15.4 / ZigBee
Sensores	Conector de expansión de 51 pines.	Conector de expansión de 51 pines.	
Tamaño (mm3)	58x32x7	58x32x7	36x48x9
Precio	90 €	108 €	276 €

- **WSN Classroom Kit:** es un kit orientado a entornos académicos donde con muy pocos componentes, y ayudado por software y tutoriales, resulta muy sencillo hacer una pequeña red de sensores.

Todos los kits incluyen todos los componentes, tanto hardware como software, necesarios para el montaje y desarrollo de una red de sensores.

2.1.4.5 Dust Networks.

La familia de productos de Dust Networks [11] están orientadas a entornos industriales, aunque pueden ser utilizados dentro de otros campos. Proporcionan una solución wireless que permite extender la monitorización y control de forma sencilla, rápida y económica.

Dispone de 3 varias familias de productos: SmartMesh IA-500, SmartMesh-XD y SmartMesh-XT; y en todas ellas se trabaja en 2.4GHz sobre el estándar IEEE 802.15.4 (en la familia XT existen modelos a 900 MHz). Las capacidades de computación, sensorización o capacidades de energía difieren de unas a otras. Como ejemplo, se exponen a continuación las características del modelo M2510 de la familia SmartMesh AI-500.

Figura 2.19.: Modelo M2510 de la familia SmartMesh IA-500.

El modelo M2510 es un módulo de 22 pines de conexión que puede funcionar con dos pilas de tipo AA, puede funcionar como router o como nodo de forma que la implementación de una topología de tipo Mesh sea muy sencillo. Integra todos los componentes del circuito de radio incluida una pequeña antena.

Dispone de un kit de evaluación con 12 nodos de evaluación y todo el contenido necesario para la realización de diferentes pruebas con el kit.

2.1.4.6 SensiCast.

Su producto Sensinet [22] es una completa red de sensores inalámbrica orientada a entornos industriales y comerciales. El sistema incluye todo el hardware y software necesario para controlar procesos y datos ambientales.

Este sistema está orientado a su instalación rápida con el hardware y el software que se proporciona, y no la integración de nuevos nodos que no se proporcionen actualmente.

Figura 2.20.: Nodo de la red SensiNet.

2.1.4.7 **SquidBee Mote.**

SquidBee [23] es una mota realizado con hardware libre, que gracias a sus sensores de humedad, temperatura o luminosidad es capaz de captar datos del entorno y enviarlos de manera inalámbrica por la red utilizando el protocolo ZigBee. El objetivo del proyecto es la creación de un "open mote" para la creación de redes de sensores inalámbricas.

Figura 2.21.: Modelo de mota de SquidBee Mote.

Los desarrolladores de este sistema son los mismos que han desarrollado la plataforma Arduino, y en la página principal del proyecto se proporciona toda la información necesaria para su utilización.

	SquidBee Mote
Procesador	Atmel ATmega8 o ATmega168
Mem. RAM	1 KB SRAM
Mem. Flash	512 B EEPROM
Conectores	-
Conexión	IEEE 802.15.4 con antena integrada
Tamaño	70x60x30
Precio	-

2.1.4.8 **Comparativa.**

A modo de resumen, se ha realizado una tabla comparativa con las características anteriormente descritas, escogiendo los modelos superiores de cada línea de productos para aquellas en las que había más de uno. Lógicamente en ISMED se adquirirán los kits con unas especificaciones de procesador o memoria superiores.

	Procesador	Mem. RAM	Mem. Flash	Slots	Conexión	Sensores	Otras características	Precio
Sun SPOT	ARM920T de 32 bit a 180 MHz	512 KB	4 MB	6 entradas analógicas, 2 botones, 5 pins de entrada/-salida de propósito general, 4 pins de salida	IEEE 802.15.4 con antena integrada	Acelerómetros en 3 ejes 2G/6G, temperatura, luminosidad, 8 LED tricolor	Conector USB	630 €, con kit de desarrollo
Basix 400 xm-bt	Intel XScale PXA255 a 400 MHz	64 MB	16 MB	1 x 60 pin	Bluetooth	-	Permite conectar tarjetas de memoria MMC	169 $
Connex 400 xm-bt	Intel XScale PXA255 a 400 MHz	64 MB	16 MB	1 x 60 pin, 1 x 92 pin	Bluetooth	-		169 $
Verdex xl6p	Marvell PXA270 con XScale a 600 MHz	128 MB	32 MB	1 x 60 pin, 1 x 120 pin, 1 x 24 pin flexible	Bluetooth	-	USB Host signals, CCD Camera signals	169 $
Verdex pro xl6p	Marvell PXA270 con XScale a 600 MHz	128 MB	32 MB	1 x 60 pin, 1 x 80 pin, 1 x 24 pin conector flexible y plano	-	-	Tarjeta Micro SD USB Host signals, CCD Camera signals	169 $
Overo Earth	Procesador de aplicaciones OMAP 3503 con CPU ARM Cortex-A8 a 600 MHz	256 MB DDR de bajo consumo	256 MB NAND	2 x 70 pin, 1 x 27 pin conector flexible y plano	-	-	Tarjeta Micro SD; Soporte para placas base "OMAP 35x-based Overo", bus I2C, 6 x líneas PWM, 6 x A/D, 1-wire, UART, entrada de cámara, entrada de micrófono, salida de auriculares, entrada batería de reserva, USB OTG signal, USB HS host	149 $
SquidBee Mote	Atmel ATmega8 o ATmega168	1 KB SRAM	512 B EEPROM	-	IEEE 802.15.4 con antena integrada	-	-	-

2.2 Aprendizaje

2.2.1 Introducción

Los entornos inteligentes o entornos de inteligencia ambiental plantean un cambio de paradigma desde una visión techno-centred a otra human-centred donde la tecnología sería el que se adaptaría a las necesidades, preferencias, hábitos, costumbres,... de los usuarios. Así el entorno debe ser capaz de reconocer a las personas, aprender de su comportamiento e incluso anticiparse a sus necesidades. Estos entornos han sido definidos como "digital environments that proactively, but sensibly, supports people in their daily lives" [87].

Los entornos inteligentes, como paradigma, tienen potencial para hacer un gran impacto en la vida diaria de los usuarios, alterando positivamente las relaciones entre el usuario y las tecnologías.

Hay que destacar que hasta la fecha la AmI ha atraído a muchos investigadores y algunas aplicaciones han sido desarrolladas con diferente grado de éxito. Teniendo en cuenta la complejidad de dichos entornos, donde el hardware, software y sistemas de comunicación tienen que cooperar eficientemente, cada proyecto se ha centrado en diferentes aspectos de esa compleja estructura. En ese sentido, es comprensible e incluso lógico que los primeros desarrollos se hayan centrado en la parte de hardware y comunicaciones como forma de soporte para el resto de las aplicaciones. Esto ha supuesto que la inteligencia de dichas aplicaciones sea solo simples automatizaciones que actúan de forma reactiva en el entorno. Así, se ha identificado la necesidad de dotar de inteligencia a esos entornos como una de las prioridades para alcanzar el paradigma de entornos inteligentes. Aunque hasta la fecha han sido muchos los investigadores [76]-[662] que han destacado la importancia de este aspecto pocos trabajos han sido desarrollados en ese sentido, aunque con notables excepciones como se verá a continuación.

Una de las ideas subyacentes al paradigma de inteligencia ambiental es que a diferencia de los entornos actuales donde el usuario tiene que adaptarse al entorno y tiene que aprender cómo usar la tecnología, en entornos de AmI es el entorno mismo el que se adapta al usuario. Así, el usuario debería estar exento de alguna obligación de programar dispositivos [357]. El entorno automáticamente debería aprender cómo reaccionar a las acciones de los usuarios, todo ello de una forma no intrusiva. Así, la capacidad de aprendizaje se convierte en un aspecto indispensable de los entornos inteligentes. Aprendizaje, en entornos de AmI, significa que el entorno tiene que obtener conocimiento acerca de las preferencias, hábitos,... de los usuarios para poder acuerdo a ellos [317], [475]. El aprendizaje de patrones permite:

- Predecir las necesidades y acciones del usuario. Así, automatizando dichas acciones (por ejemplo, ajustando automáticamente la temperatura de acuerdo a las preferencias del usuario) se puede facilitar la vida diaria del usuario [225], [366].

- Hacer el sistema más eficiente en términos de eficiencia energética [579] (por ejemplo, apagando las luces si el usuario ha salido y no va a volver en 1 hora).

- Aumentar la seguridad [684]-[415] (por ejemplo emitiendo alarmas cuando se detectan situaciones fuera de los normal)

Como todo nuevo paradigma que impacta en la vida diaria de las personas, la AmI no ha quedado exenta de críticas. Hay suspicacias acerca de la perdida de la privacidad o el aumento de la individualidad en la sociedad. Estas críticas también fueron vertidas a la computación en general de modo que la AmI, como todo nuevo paradigma, tendrá que afrontar estos aspectos de forma adecuada, dándole una mayor importancia al aspecto de human-centric [475].

2.2.2 Características específicas de entornos AmI

Los entornos de inteligencia ambiental tienen ciertas características específicas, es decir, diferentes a otros entornos, que hay que tener en cuenta a la hora de llevar a cabo el aprendizaje. Estas características han sido agrupadas en cuatro grupos.

2.2.2.1 Importancia de los usuarios

La primera característica diferenciadoras de los entornos AmI es la importancia que se le da al usuario, por ejemplo Leake et al. [475] comentan que además de ser el centro de todo el desarrollo, el usuario tiene que tener el poder sobre todos los dispositivos. El usuario es el centro de la AmI. Al usuario no le tiene que suponer ningún esfuerzo añadido que el entorno se convierta inteligente. El cambio que debe notar el usuario es que el entorno realiza tareas de forma adecuada que antes los realizaba él y le suponían cierto esfuerzo. Si el entorno no realiza las tareas de forma adecuada el usuario deberá realizar correcciones, suponiendo un esfuerzo, que si es repetitivo puede ser un factor de no aceptación.

Así, Marzano [531] menciona que el hogar del futuro se asemejará más al hogar del ayer que al hogar actual ya que los dispositivos estarán integrados en los objetos cotidianos y no será necesario la inserción de nuevos objetos para dotar de inteligencia al entorno (Ver Figura 1). En este sentido menciona que la idea de equipar el hogar con dispositivos que aumenten la calidad de vida y reduzcan el trabajo no es novedosa, sino que la novedad que introduce la AmI es el de la transparencia y la posibilidad de interacción con el entorno.

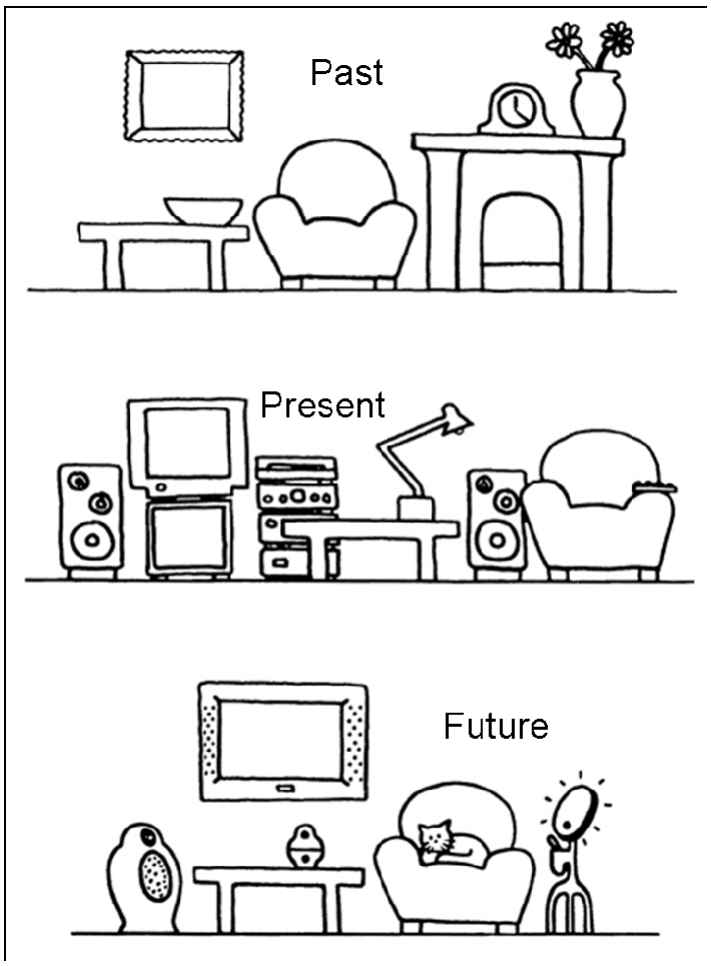


Figura 2.22.: Hogar del futuro.

Otro aspecto a tener en cuenta es la interacción entre el usuario y el entorno. Dicha interacción debe ser a través de dispositivos comunes. El usuario está acostumbrado al uso de ciertos dispositivos. Así, la información necesaria para la generación de reglas como el feedback posterior tienen que ser obtenidos a partir de sensores estándares. Introducir nuevos dispositivos o nuevos interfaces para la interacción ha sido criticado por autores como Mozer [577] o Rutishauser [698], ya que el usuario está acostumbrado a usar dispositivos normales. Además habrá muchos usuarios

que rechacen la idea de AmI si para ello hay que aprender a usar nuevos dispositivos o nuevos interfaces de usuario y ese aprendizaje no conlleva grandes beneficios.

Además los entornos AmI tienen que adaptarse a usuarios particulares y no a usuarios genéricos, aunque algunas reglas pueden ser relativas a usuarios genéricos o grupos de usuarios. Hay que destacar la complejidad de los comportamientos de los usuarios. Liao et al. [486] en su trabajo para reconocer el comportamiento destaca que 'Human behaviors involve many uncertain factors and are hard to model deterministically'. Así, Lialo considera que el 'probabilistic reasoning' parece un approach interesante. Otras características del comportamiento humano que Liao menciona en su trabajo son:

- Muchos factores que afectan al comportamiento (preferencias y capacidades) son difíciles de caracterizar.
- Existen grandes diferencias entre lo que miden los sensores (lo que observamos) y el comportamiento (lo que queremos saber)
- El comportamiento humano es muchas veces complejo. Kulkarni [453] menciona un ejemplo de este tipo de comportamiento. Imagina una habitación que tiene la regla de que si detecta que hay alguien en la habitación que está despierto encienda las luces. Esto en un principio puede parecer lógico, pero Kulkarni menciona que el ser humano tiene comportamientos complejos que no pueden ser reducidos a reglas tan simples. Por ejemplo hay personas que quieren ver una película en su habitación con las luces apagadas.
- Los usuarios cometen errores y reconocerlos es muy importante para no confundirlos con falsos deseos.

Debido a todas las posibles complicaciones expuestas anteriormente, además de los datos obtenidos de los sensores, conocimiento adquirido externamente será utilizado para llevar a cabo el aprendizaje. Este conocimiento externo puede ser:

- Información médica del paciente.
- Preferencias explicitas.
- Información acerca de actividades (por ejemplo, cuando se va de vacaciones).

2.2.2.2 Naturaleza de los datos

Los datos recogidos de los sensores influenciará de forma definitiva el proceso de aprendizaje, ya que éste estará basado en dichos datos. Esta dependencia es incluso mayor si consideramos la problemática de recoger e interpretar dichos datos de forma correcta.

Los datos de los sensores llegarán de forma continua de diferentes fuentes, de modo que la integración de dicha información conllevará ciertos retos a tener en cuenta. Diferentes formatos, diferentes estados de diferentes dispositivos incrementan la complejidad de dicha integración.

Otro aspecto a tener en cuenta es como se ha indicado anteriormente los datos necesarios para el aprendizaje se recogerán desde los sensores, por lo que habrá diversas fuentes de información [698], 61], lo que puede exigir la sincronización y la coordinación de los mismos.

Por otro lado, considerando la naturaleza de la información recogida, teniendo en cuenta que es referente a comportamientos del usuario, hay que considerar acciones contradictorias entre sí. En este sentido, como se ha indicado en la anterior sección, conocimiento adquirido desde el exterior podría ayudar a esclarecer dichos casos.

Finalmente tal como indican Rivera-Illingworth et al. [684], hay que tener en cuenta también la diferencia entre entornos de laboratorio y entornos reales, ya que por ejemplo cámaras y micrófonos pueden dar un resultado óptimo en entornos de laboratorio mientras que en entornos reales pueden ser fuente de muchos problemas.

2.2.2.3 Representación del conocimiento adquirido

Como el conocimiento adquirido, es decir, los patrones aprendidos serán utilizados dentro de un sistema más grande, habrá que considerar y analizar cómo representar el output del sistema de aprendizaje.

Si el conocimiento adquirido tiene que ser combinado con otro tipo de conocimiento sería preferible que dicho conocimiento sea comprensible. Además de combinarse con otro tipo de conocimiento, en ciertos entornos el entorno tendrá que dar "explicaciones" de por qué ha actuado de la forma en que lo ha hecho, por lo que en estos casos también sería preferible que la salida sea comprensible, para que, de esa forma el entorno pueda explicar al usuario su lógica de actuación [475].

Otro aspecto a considerar es la representación en si de los patrones, es decir, cómo, de una forma sencilla, representar las preferencias, hábitos,... de los usuarios. En este sentido, la utilización de secuencias para la representación de las actividades comunes de los usuarios parece prometedora.

2.2.2.4 Adaptación temporal de los patrones

Para adaptarse a las necesidades o preferencias del usuario, el entorno debe de tomar decisiones continuamente, y esas decisiones serán tomadas en base al conocimiento que tiene el sistema o cada agente. Pero este conocimiento se tiene que cambiar con el tiempo.

Se pueden definir diferentes etapas en el aprendizaje de los patrones:

- **Generación inicial de conocimiento:** La primera necesidad o reto que surge es la generación del conocimiento o "rulebase". Para adaptarse a las necesidades o preferencias del usuario, el entorno debe de tomar decisiones continuamente, y esas decisiones serán tomadas en base a un conocimiento (reglas, casos,...) que tiene el entorno. Ese conocimiento puede ser generada de distintas maneras. Una opción sencilla parece ser que el mismo usuario o un experto construya o defina ese conocimiento, pero teniendo en cuenta ciertas características de entornos AmI (no intrusivo, transparente,...) se trata de generarla automáticamente utilizando algoritmos de aprendizaje automático o machine learning [698]. Además no es efectivo ni en tiempo ni en esfuerzo implementar (y menos mantener) el conocimiento manualmente. La generación de conocimiento y por lo tanto la adaptación del entorno al usuario puede ser pensado en distintas etapas, ya que la generación de conocimiento (extracción de perfiles de usuario) puede llevar un tiempo considerable. Así, la primera adaptación del entorno (adaptación a corto plazo) puede ser llevado a cabo sin la extracción de perfiles de usuario. La adaptación (adaptación a largo plazo), llevado a cabo mediante la predicción y anticipación de las necesidades del usuario, necesita de perfiles de usuario extraídos a partir de un conjunto de datos extensos que necesita tiempo en recopilarlos, por lo que también es necesario la adaptación a corto plazo mencionado anteriormente.

- **Adaptación del conocimiento:** Una vez que se hayan definido el conocimiento, el entorno va a tomar decisiones y actuar en consecuencia. Al referirse a estos conocimientos hay que darse cuenta de que se refieren a gustos, preferencias, hábitos,... de los usuarios, y que pueden cambiar a lo largo del tiempo. La necesidad de adaptación del conocimiento puede venir por algunas de las siguientes situaciones:

  - Posible modificación de comportamientos, necesidades, costumbres, gustos o preferencias del usuario.
  - Posible modificación del contexto en el que se sitúa el usuario con la eliminación, modificación o inclusión de nuevos dispositivos.
  - Existencia de situaciones no previstas.
  - Conocimiento no optimizado.

  Así, hay una clara necesidad de que este conocimiento sea adaptada, refiriéndose con adaptación a la posible modificación (ajuste) de alguno de los valores, a la generación de un nuevo conocimiento e incluso a la eliminación de un conocimiento no válido.

- **Actuación inteligente sin aprendizaje:** Para extraer patrones de comportamiento a partir de los datos recogidos "Generación inicial de conocimiento" la primera tarea es la misma

recolección. Esta recolección implica la necesidad de estar observando las tareas o acciones que realiza el usuario en un periodo de tiempo. En ese periodo de tiempo el entorno no actuará, pudiendo ser ese periodo de 5 días, 1 semana, 2 semanas o más. Durante ese primer periodo lo ideal sería que el entorno también tratase de adaptarse al usuario. Esto podría ser realizado mediante técnicas de aprendizaje que no requieren entrenamiento, p.e. Case-Based Reasoning. Además esto podría proveer al sistema los primeros feedbacks del usuario.

Así, se puede concluir que primero habrá un periodo de actuación inteligente basado en técnicas que no necesita entrenamiento. A continuación otro periodo de aprendizaje realizado con datos recolectados durante un periodo de tiempo donde se extraerá el conocimiento, y finalmente habrá una adaptación continua de ese conocimiento tratando de adaptarse a los cambios de comportamiento que pueden existir.

2.2.3 Técnicas de Machine Learning

La utilización de técnicas de machine learning para el aprendizaje de patrones parece muy prometedor, debido a que históricamente han sido estas técnicas las usadas para llevar a cabo todo tipo de aprendizajes en todo tipo de entornos.

Lo que hay que destacar es que técnicas de machine learning que han sido útiles en otros entornos pueden no ser satisfactorios en entornos de inteligencia ambiental debido a las características especiales de dichos entornos. En este sentido, a continuación se muestra un análisis de diferentes técnicas de aprendizaje incidiendo en sus fortalezas (✓) y debilidades (***x***) para el aprendizaje de patrones en entornos AmI. No todas las técnicas de aprendizaje han sido consideradas sino que han sido seleccionadas las más prometedoras y las que han sido utilizadas por diferentes grupos para dicha tarea.

Árboles de decisión

✓ La salida puede ser traducida a un conjunto de reglas fácilmente interpretadas.

✓ Produce reglas que son no-ambiguas, de modo que no es relevante el orden en el que se ejecutan y no hay conflictos entre ellos.

✓ Funciona bien incluso cuando hay errores en los datos de entrenamiento o hay valores nulos.

x El añadir o eliminar reglas puede exigir la re-estructuración de todo la estructura del árbol.

x Las reglas que son extraídas directamente de los árboles de decisión pueden ser más complejas de lo necesario. En entornos complejos, esta característica puede ser una complicación innecesaria.

x Dificultades para representar secuencia temporales. Aunque investigaciones recientes indican que se pueden generar árboles de decisión temporales.

Aprendizaje de reglas

✓ Las reglas son la representación más sencillas para la interpretación del conocimiento.

✓ Es fácil introducir excepciones en ellas

✓ Añadir, modificar o borrar una regla de la estructure existente es fácil.

✓ En el proceso de aprendizaje se puede especificar el límite para su aceptación.

x Es posible que las reglas no cubran todas las opciones o casos. (✓) Pero esos casos se pueden utilizar para la adaptación de dicho conocimiento.

x Hay posibilidad de conflicto entre las reglas, es decir, en la misma situación puede haber dos reglas con consecuencias incompatibles.

x Como el conjunto de reglas es generado secuencialmente, un pequeño cambio puede hacer que todo el conjunto sea inconsistente.

Descubrimiento de secuencias

✓ El comportamiento humano es muchas veces mejor representado mediante secuencias que por medio de acciones independientes.

✓ El uso de secuencias permite usar tiempos relativos entre las distintas acciones.

✓ Añadir una nueva acción a una secuencia o una nueva secuencia al conjunto de secuencias es fácil.

✓ En el proceso de descubrimiento se puede establecer el límite para su aceptación.

x El descubrimiento de secuencias ha estado orientado hasta la fecha a descubrir eventos que suceden dentro de un periodo de tiempo.

Redes Neuronales Artificiales

✓ Los datos de entrenamiento pueden contener errores.

✓ Robusto a los errores de entrenamiento.

✓ Aplicable a problemas complejos tales como aprendizaje de comportamientos o asociaciones entre datos.

x La estructura no es comprensible. Sólo las entradas y las salidas pueden ser interpretadas.

x El tiempo de entrenamiento puede ser largo. Además, debido a su estructura estática, necesita ser re-entrenada cuando se presentan nuevos datos.

x Posibilidad de sobre-entrenamiento.

Instance based learning

✓ No necesita ningún periodo entrenamiento porque no estiman una solución para todo el espacio, sino que lo hace por cada nueva instancia.

✓ La adaptación es rápida. A diferencia de otras técnicas la adaptación a nuevos casos o conocimiento es rápida.

x No hace explicito la estructura del conocimiento adquirido. Pero la captura de casos similares puede ser utilizado para explicar porqué el sistema ha actuado de esa forma.

x Es lento para conjuntos de datos grandes. Este problema podría ser afrontado mediante estereotipos creados mediante la agrupación de casos similares.

x Establecer pesos es difícil.

Reinforcement Learning

✓ Útil para entornos que aprenden su comportamiento basándose en la continua interacción con el usuario, donde las acciones de los usuarios son interpretados como premio o castigo de las acciones realizadas por el entorno.

x Dificultad para interpretar el feedback del usuario. Además si el feedback es obtenido después de ciertas acciones es muy difícil estimarlo.

La Tabla 2.16 define las fortalezas y debilidades de cada técnica referente a los periodos de aprendizaje definidos anteriormente.

	Fortalezas / Debilidades	Árboles de Decisión	Aprendizaje de reglas	Descubrimiento de secuencias	Redes Neuronales	Instance Based Learning	Reinforcement Learning
Aprendizaje en un periodo corto	Fortalezas	Salida comprensible	Salida comprensible	Salida comprensible	-	No necesita entrenamiento.	-
	Debilidades	Necesidad de entrenamiento.	Necesidad de entrenamiento.	Necesidad de entrenamiento.	Necesidad de entrenamiento. Salida no comprensible	Sensible a instancias complicadas y erróneas	Necesidad de un modelo
Aprendizaje de patrones	Fortalezas	Salida comprensible	Salida comprensible. Posibilidad de excepciones.	Reglas que representan relaciones temporales	Capacidad de generalizar	Extracción de instancias generales.	-
	Debilidades	Dificultad para representar situaciones complejas (p.e. excepciones).	Posible conflictos entre reglas	Algoritmos no centrados en descubrir los lapsos de tiempo	Salida no comprensible	Lento con instancias complicadas	Necesidad de un modelo
Adaptación de patrones	Fortalezas	Facilidad para identificar el error	Facilidad para añadir, modificar o eliminar reglas.	Facilidad para añadir, modificar o eliminar reglas.	Posibilidad de introducir neuronas dinámicamente.	Posibilidad de clasificar instantes no conocidas.	Aprende interactuando con el usuario.
	Debilidades	Necesidad de reformular el árbol.	-	-	Necesidad de entrenar otra vez	Costes de computación y de tiempo.	Dificultades para interpretar los feedbacks.

Tabla 2.16.: Técnicas de Machine Learning con sus fortalezas y debilidades.

Actualmente es muy difícil establecer una aproximación global que pueda ser útil para todos los entornos. Teniendo en cuenta las necesidades, los objetivos, ... de cada entorno, diferentes técnicas pueden ser utilizados. También debido a las características de diferentes técnicas, la combinación de diferentes técnicas puede ser una buena solución. Si por alguna razón se decide que la combinación de diferentes técnicas es interesante, los interfaces entre ellos deben ser definidos claramente. Un ejemplo de dicha combinación puede ser la utilización de Instance based learning para actuar de forma inteligente mientras se recogen los datos y luego utilizar alguna técnica que necesite entrenamiento (p.e. aprendizaje de secuencias), utilizando reinforcement learning para la adaptación de dichos patrones.

2.2.4 Utilización de técnicas y grupos de investigación

2.2.4.1 Técnicas utilizadas

Aunque, como se ha indicado anteriormente, poco énfasis se ha hecho hasta la fecha en lo que se refiere al aprendizaje, ha habido excepciones que merecen ser mencionados. Teniendo en cuenta las técnicas definidas en la anterior sección, se van a analizar los trabajos realizados con dichas técnicas.

2.2.4.1.1. Redes Neuronales artificiales Chan et al. [183] y Mozer et al. [577], [387], [576] fueron de los primeros que trataron de inferir reglas sobre patrones de usuario para entornos inteligentes basándose en la vida diaria de los usuarios. Plantearon la utilización de redes neuronales artificiales para predecir el futuros estatus y controlar dispositivos tales como la calefacción, la televisión o las luces. A partir de ese momento muchos otros investigadores han utilizado las redes neuronales para el desarrollado de entornos inteligentes [684], [134]-[204]. Incluso se ha realizado una recopilación de dichos trabajos en un artículo [113].

2.2.4.1.2. Técnicas de clasificación Técnicas de clasificación, tales como árboles de decisión o aprendizaje de reglas también han sido utilizadas para el aprendizaje en entornos inteligentes. Un ejemplo de ello es el SmartOffice [315] donde el entorno trata de anticipar las intenciones de los usuarios. Teniendo en cuenta las pruebas realizadas Brdiczka et al. [145] definen que los árboles de decisión dan los mejores resultados comparado con las técnicas de Find-S y Candidate Elimination [558].

2.2.4.1.3. Aprendizaje de secuencias Una técnica que está adquiriendo importancia para la predicción de actividades es el aprendizaje de secuencias que permite definir secuencias de acciones del usuario. Así, en [396] Jakkula y Cook desarrollan un framework general para representar y razonar sobre relaciones temporales entre acciones en entornos inteligentes. Representan las relaciones temporales utilizando las relaciones definidas por Allen [362], [57] y luego utilizan técnicas de data mining para el descubrimiento de asociaciones frecuentes (usando el algoritmo del A priori [44]).

2.2.4.1.4. Instance Based Learning Case-Based Reasoning, el cual se puede considerar como una técnica incluida dentro del Instance based learning, ha sido utilizado para el aprendizaje de preferencias del usuario. Por ejemplo, en el proyecto de MyCampus [706] utilizaban dicha técnica para filtrar los mensajes en base al feedback del usuario. Almacenaban información acerca en forma de casos y cuando una nueva situación se presentaba lo clasificaban comparando con los casos previos. En este caso, las relaciones mediante atributos, para obtener las métricas de similitud, fueron definidos mediante redes bayesianas. Nuevas aproximaciones basadas en Case-based reasoning, tales como Temporal CBR [48] han sido sugeridas también.

2.2.4.1.5. Reinforcement learning La técnica de reinforcement learning puede ser considerado como una técnica de adaptación. Por ejemplo, Chidambaram et al. [577] utilizaban dicha técnica conjuntamente con otras técnicas, para, teniendo en cuenta los feedback negativos que daba el usuario mejorar la exactitud de las predicciones.

2.2.4.1.6. Otras técnicas Aparte de las técnicas definidas anteriormente otras técnicas también han sido utilizadas para llevar a cabo el aprendizaje en entornos inteligentes.

En el proyecto MavHome [242], [366],[224] los autores intentan modelar el comportamiento de los usuarios mediante cadenas de Markov para maximizar el confort de los mismos. Considerando las dificultades que tienen técnicas tradicionales como los árboles de decisión de utilizar información histórica para la predicción secuencial [668], sugieren la utilización de cadenas de Markov. Algunos investigadores [684], [280] también han mencionado las dificultades que pueden tener los Hidden Markov Models debido a que son incapaces de afrontar comportamientos complejos y su algoritmo de aprendizaje tiene dificultades cuando se trata de información proveniente de numerosos sensores de bajo nivel. Así, algunos autores han preferido utilizar Hidden Semi-Markov Model dado que generaliza la duración de los estados.

Investigadores de la universidad de Essex [357], [265] han trabajo con la lógica fuzzy para el aprendizaje de patrones. Así, basándose en los datos recogidos por los sensores, tratan de inferir las funciones de pertenencia y las reglas del sistema de lógica fuzzy, el cual es capaz de adaptarse.

2.2.4.2 Grupos de investigación y/o entornos

Aunque la investigación del aprendizaje no haya sido hasta la fecha una de las prioridades entre los investigadores de AmI, hay notables excepciones. Así, hay grupos de investigación que viendo la necesidad de dotar de inteligencia el entorno, más allá de las necesidades de conectividad y comunicación, hayan empezado a trabajar en el aprendizaje.

A continuación se detallarán los grupos de investigación y entornos más activos.

2.2.4.2.1. University of Texas / MavHome El proyecto MavHome (Managing an adaptive versatile Home) trata de maximizar el confort del usuario, minimizando el consumo de recursos y manteniendo la seguridad y privacidad del usuario [224]. La idea de MavHome está siendo desarrollado e implementado en "MavLab" y en "MavPad" (un apartamento del campus habitado por estudiantes) (Ver Figura 2.23). En dicho entornos están automatizados el control de luces y electrodomésticos, así como el aire acondicionado, ventiladores y persianas.

Figura 2.23.: MavHome y MavLab.

2.2.4.2.2. University of Ulster La universidad de Ulster en su trabajo conjunto con el hospital del Ulster trabaja con un entorno de 30 apartamentos individuales que trata de proporcionar soluciones tecnológicas para una vida más independiente para las personas mayores [89].

2.2.4.2.3. University of Essex/iDorm La universidad de Essex cuenta con un banco de pruebas donde hace test de sus mecanismos de aprendizaje y adaptación [357]. El iDorm (Ver Figura 2.24) es un espacio multiuso que contiene áreas para dormir, trabajar o entretenimiento. En estos momentos los creadores de iDorm están desarrollando el iDorm2, un apartamento de 2 habitaciones que trata de mejorar las prestaciones del iDorm [604].

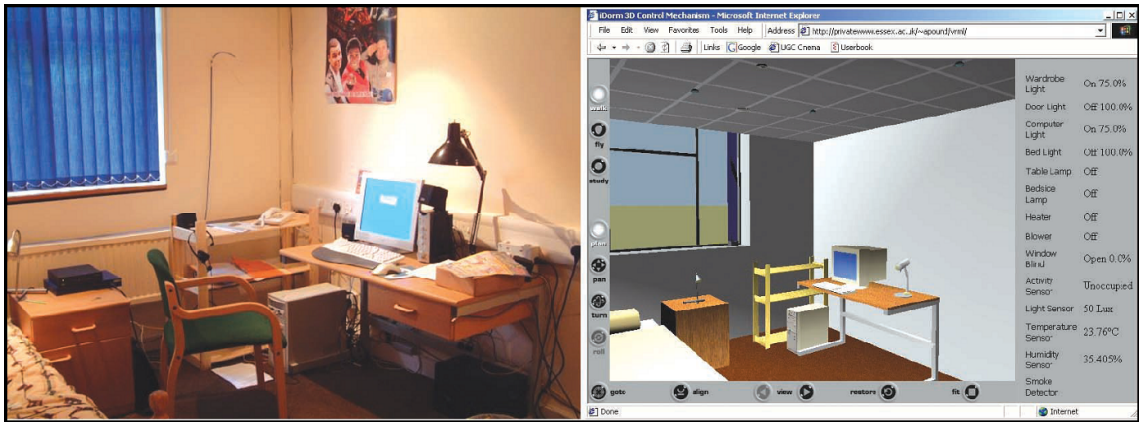


Figura 2.24.: iDorm.

2.2.4.2.4. Georgia Institute of Technology/AwareHome Esta casa cuenta dos espacios idénticos pero independientes entre sí. Cada espacio cuenta con dos habitaciones, dos cuartos de baño, una oficina, una cocina, sala de estar y cuarto para la limpieza. Han desarrolla algunas aplicaciones, por ejemplo una para encontrar objetos perdidos.

2.2.4.2.5. Philips/HomeLab Philips, con su HomeLab, ha sido una de las más activas entre las empresas del sector industrial. Sobre todo Philips se centra en la interacción y en cómo puede el hogar ayudar en la realización de tareas diarias. Además le da gran importancia al "social awareness" para que sea adecuado y aceptable para los usuarios. Algunos desarrollos concretos utilizados en HomeLab [26] son por ejemplo "AmI lightings Living light" donde se trata de crear el ambiente idóneo para ver la televisión o escuchar música mediante el control de las luces. Otra idea es "Sharing context and experiences" donde se trata de compartir espacios comunes aún cuando físicamente no se comparta ese espacio.

2.2.4.2.6. Siemens Siemens ha invertido en Smart Homes tratando de enriquecer el entretenimiento teniendo en cuenta la seguridad y la economía. Control de luces, ahorro de energía, pantallas táctiles, seguridad de dispositivos o monitorización de niños son algunas de los aspectos que trata Siemens.

2.2.4.2.7. Intel/Proactive Health Group Intel se centra en la investigación para aumentar la calidad de vida de los ancianos, dando importancia al "social network".

2.2.4.2.8. Otros entornos Aunque los arriba mencionados sean los grupos y los entornos más significativos, hay otros grupos que utilizando entornos más sencillos y más reducidos investigan aspectos concretos.

Uno de los aspectos donde los investigadores más se han centrado es la ayuda a personas de la 3a edad y discapacitados, teniendo en cuenta la perspectiva de las personas cuidadas como de los cuidadores.

- Muchos han sido los autores que han mencionado los centros de ayuda a las personas de 3a edad y discapacitados. Residencias, hospitales, casas medicalizadas y apartamentos de ancianos son algunos de los entornos mencionados [89]. 'Health Monitoring and Assistance' es otro de los ámbitos donde la AmI puede ser aplicada [224]. Los discapacitados son otro grupo de posibles beneficiarios de AmI [283].
- Martin et al. [529] definen un entorno de teleayuda para aumentar el bienestar de los ancianos comprobando si cambian los patrones de comportamiento. Lühr et al. [504] también se centran en detectar desviaciones del comportamiento habitual, y son conscientes de que para ello necesitan tener un modelo de comportamiento para poder compararlo.

- Chan et al. [183] y Rivera-Illinworth et al. [684] tratan de desarrollar un sistema capaz de monitorizar a personas de 3a edad y discapacitados para permitir una vida independiente. Para ello han diseñado un sistema de monitorización, que ha sido instalado en 12 habitaciones, para que la gente tenga más seguridad sin causarles ningún problema adicional.
- Demiris et al. [254] ven la necesidad de un sistema AmI para ayudarles a las personas mayores a seguir siendo independientes en lo posible. Para ello, además de a los ancianos, Demiris et al. ven la necesidad de involucrar a los cuidadores en el desarrollo de este sistema. En entorno donde están aplicando este sistema es un entorno real de una residencia de ancianos de 32 habitaciones.
- Pollack [642], al analizar la tendencia demográfica de USA y plantear la necesidad de sistemas que ayuden a la gente a vivir independientemente, menciona que con dichos sistemas los cuidadores no serán sustituidos, sino que la tecnología mejoraría la calidad de vida tanto de los pacientes como de los cuidadores. Kautz et al. [415] tratan con enfermos de Alzheimer ayudándoles con dos demostradores a orientarse y realizar múltiples tareas diarias.

2.2.5 Conclusiones

De lo expuesto en el capítulo se pueden extraer varias conclusiones:

- El aprendizaje automático de patrones de comportamiento de los usuarios es necesario si se quieren obtener entornos inteligentes de una manera no intrusiva.
- La adaptación a lo largo del tiempo de estos patrones también es necesario ya que estos patrones pueden cambiar.
- Hasta la fecha, la mayoría de las investigaciones llevadas a cabo en AmI no ha hecho mucho énfasis en la necesidad de aprendizaje y adaptación, ya que se han centrado más en temas relativos a sensores y redes, aunque con notables excepciones
- El aprendizaje tiene que llevarse a cabo sin que ello le suponga un esfuerzo añadido al usuario, o en su defecto, tratando que el usuario interactúe mediante interfaces estándares o interfaces multimodales de fácil uso. De la misma forma, el feedback del usuario, necesario para la adaptación del conocimiento, tiene que recogerse preferentemente a través de dispositivos estándares o interfaces multimodales de fácil uso.

La utilización de algoritmos de Machine Learning para el aprendizaje de tales patrones parece una opción lógica, pero la(s) técnica(s) de Machine Learning utilizada(s) tiene(n) que tener en cuenta las características específicas de entornos AmI, por ejemplo:

- Tipología de datos: Ruido en los datos, Diversas fuentes de información, etc.
- Objetivos que se persiguen: Confort del usuario, Ahorro de energía, etc.
- Capacidad de los dispositivos: Limitaciones de memoria y procesamiento, etc.
- Capacidad de razonar la decisión tomada, por lo que puede ser importante la interpretabilidad de lo aprendido.

Así, la elección de qué técnica de Machine Learning utilizar puede depender de las características arriba mencionadas, del entorno particular que tenemos, preprocesado que se ha hecho de los datos obtenidos, etc. Hasta la fecha, diferentes autores han utilizado diferentes técnicas dando cada uno da importancia a diferentes aspectos, por lo que no se ha podido establecer una técnica como la más usada o como la que da mejores resultados. Así como apunta Muller "In many research projects, great results were achieved... but the overall dilemma remains: there does not seem to be a system that learns quickly, is highly accurate, is nearly domain independent, does this from few examples with literally no bias, and delivers a user model that is understandable and contains breaking news about the user's characteristics. Each single problem favours a certain learning approach".

2.3 Composición de servicios

La composición de servicios consiste en conectar servicios entre sí para crear procesos de negocio más complejos o de alto nivel, facilitando de esta manera el desarrollo de nuevas aplicaciones reutilizando componentes ya existentes [660]. De esta manera dado un conjunto de servicios y con métodos eficaces de composición automática de servicios la visión de la programación en la que se especifica qué es lo que tiene que hacer el programa y no el cómo lo tiene que hacer puede llegar a ser una realidad [457].

En el área de los WS el enfoque de los SWS [63] es un paso importante hacia el descubrimiento y la composición dinámica [779], donde sistemas inteligentes tratan de componer servicios autónomamente, basándose en requisitos de usuario abstractos. Todo ello mediante SWS construidos basándose en técnicas de representación del conocimiento, con ontologías que describen el dominio de manera formal y técnicas de planificación de IA para hacer los sistemas de composición más autónomos [814], [816].

Fujii y otros [312] manifiestan que la composición dinámica de servicios es una de las tecnologías clave que facilitará el desarrollo de aplicaciones para la computación ubicua ya que su aplicabilidad es muy útil en entornos donde el número de componentes, servicios y usuarios disponibles es variable a lo largo del tiempo.

2.3.1 Módulo de composición de servicios

Los enfoques de composición pueden ser agrupados en función del grado de participación que tiene el usuario en el mismo, pero además de este factor la composición puede ser diferenciada en dos aspectos importantes, por una parte la síntesis y por otra la orquestación (término que no consideramos adecuado, consideramos más adecuado el término ejecución) [457]:

- La síntesis se refiere a la generación del plan que tiene como objetivo realizar el comportamiento deseado combinando múltiples servicios, es decir, se refiere al empleo de técnicas para crear la composición que posteriormente será ejecutada.
- La ejecución se refiere a la coordinación del control y los flujos de datos a lo largo de los diversos componentes cuando se está ejecutando el plan previamente definido, es decir, se refiere a las técnicas o enfoques empleados para ejecutar el plan.

En los últimos años se han desarrollado muchos proyectos y enfoques que tienen como objetivo la composición de servicios tanto semánticos como no semánticos. Consideramos que en vez de enumerar todos los enfoques es mejor describir cuáles son las características o tipos de enfoques que se han llevado a cabo describiéndolos en base a tres aspectos de composición (participación del usuario, síntesis y ejecución). A continuación se describirá en qué consiste cada uno de los aspectos.

2.3.1.1 Enfoques de composición en base a la participación del usuario

La composición de servicios viene derivada de modelos empresariales [761] y puede ser categorizada en dos tipos: la composición estática (proactiva) de servicios y la composición dinámica (reactiva) de servicios [176], [177]:

- La estática es un enfoque en el que los diseñadores de aplicaciones implementan una nueva aplicación manualmente diseñándola mediante herramientas de workflow o diagramas de estado, describiendo los patrones de interacción entre los componentes del mismo. Estas aplicaciones deben de ser manualmente diseñadas antes de desplegarlas, es decir, la composición de servicios se hace en tiempo de diseño. Este tipo de composición es adecuado para aplicaciones de tipo B2B, en donde la interacción entre los componentes es compleja pero estática [312]. Este tipo de composición es la empleada por la industria.
- En cambio en la composición dinámica de servicios el modelo de proceso se genera automáticamente cuando un usuario lo precise. La composición dinámica es más aconsejable para entornos ubicuos o móviles, donde el número de componentes varía a lo largo del tiempo [312]. Al contrario que en el caso de la composición estática, la composición de servicios se realiza

en tiempo de ejecución. Este tipo de composición ha sido planteado por grupos relacionados con la Web Semántica.

- Existe otro enfoque intermedio denominado composición semi-automática o interactiva que proponen métodos y herramientas para la interacción del usuario que capturen las intenciones de los usuarios de manera interactiva y continua durante el proceso de composición del plan a ejecutar [860].

2.3.1.2 Enfoques para la síntesis de la composición

En los últimos años se han publicado diversos artículos que tienen como objetivo realizar una clasificación de las diferentes técnicas de composición, centradas en la síntesis del plan a ejecutar. A continuación se van a describir cuales son las diferentes agrupaciones que realizan los autores.

Rao y Su [667] realizan una revisión bibliográfica centrándose en las técnicas empleadas para crear la composición identificando dos técnicas principales:

- Técnicas de workflow: Esta técnica tiene como objetivo crear un workflow para su posterior ejecución. Esta técnica puede ser dividida en dos, la estática en donde el peticionario además del modelo abstracto de proceso define también las tareas y las dependencias de datos entre las mismas, en este caso solamente la selección y el binding es realizado automáticamente por el programa; o bien la dinámica en donde se crea dinámicamente el modelo de proceso abstracto y se seleccionan dinámicamente los servicios, pero definiendo como entradas una serie de restricciones además de las preferencias y de las dependencias de los servicios atómicos.

- Técnicas de planificación de IA: La composición de servicios basada en planificación consiste en emplear un algoritmo que permita resolver un problema de planificación. De esta manera partiendo de un estado inicial se determinan los pasos a realizar para llegar a un estado final o meta, pero para ello es necesario realizar una representación de los estados posibles del entorno, incluyendo el estado inicial y el final, así como las operaciones permitidas en cada uno de ellos para realizar el cambio de estado. Los autores del artículo además dividen este tipo de técnicas en 5 subgrupos, como son las de cálculo situacional, planificación basada en reglas, PDDL (Planning Domain Definition Language), prueba de teoremas y otras (HTN, etc.). Algunas de estas técnicas serán explicadas en el punto 2.3.1.3.

Peer [630] en cambio solamente se centra en las técnicas de planning de IA empleadas para la composición de Servicios Web, tras el análisis de los diferentes enfoques realiza una comparativa de los planificadores más comunes evaluándolos en base a unos criterios definidos en el artículo. Del mismo modo, May Chan y otros [182] realizan un análisis centrándose solamente en las técnicas de planning de inteligencia artificial y para ello se basan en la categorización de técnicas realizada por Ghallab y otros [323] sin profundizar en las técnicas de neo-planning y en sus extensiones, ya que no son adecuadas para entornos de Servicios Web. Del informe se extrae como conclusión que la planificación basada en HTN puede ser considerada como la más completa y adecuada para la composición automática de Servicios Web.

Existe otro análisis de Küster y otros [457] en el que la clasificación no es realizada en base a la técnica empleada para la composición, sino en los diferentes tipos de aplicaciones de composición de servicios, es decir, realizan una clasificación en base al problema que se tiene entre manos a la hora de la composición, para ello identifican 3 clases de problemas que a su vez están divididos en varios grupos: Cumpliendo las precondiciones: El servicio que ofrece el efecto deseado existe, sin embargo, no todas sus precondiciones son cumplidas. Generando múltiples efectos: La petición de servicio dispone de múltiples efectos que están relacionados, pero no pueden ser realizados por un único servicio. Superando la carencia de conocimiento: Es necesaria disponer de conocimiento adicional para satisfacer correctamente una petición.

Alamri y otros [54] presentan otro análisis que agrupa los enfoques de composición en 6 tipos diferentes: reconfiguración en tiempo de ejecución empleando wrappers, adaptación de servicios en tiempo de ejecución, lenguajes de composición, técnicas de composición basadas en workflows, técnicas de composición basadas en ontologías y técnicas de composición declarativa. Tras describir cada uno de los tipos se detalla una tabla en donde se describen las ventajas y las limitaciones de cada uno de los grupos.

Por último Ter Beek [786]-[787] y otros realizan una clasificación basada en los lenguajes existentes para composición. Por una parte describen los lenguajes para composición estática como pueden ser BPEL para orquestación y WS-CDL para coreografía y por otra parte los lenguajes dinámicos (Servicios Web semánticos) destacando OWL-S y WSMO.

2.3.1.2.1. Modelos de composición. Basándose en la expresividad del lenguaje subyacente para la especificación de tareas de usuario y los servicios del entorno es posible describir otro enfoque de agrupación, que se cree que es más adecuado de cara al presente proyecto de investigación y que son descritos a continuación en seis modelos de composición:

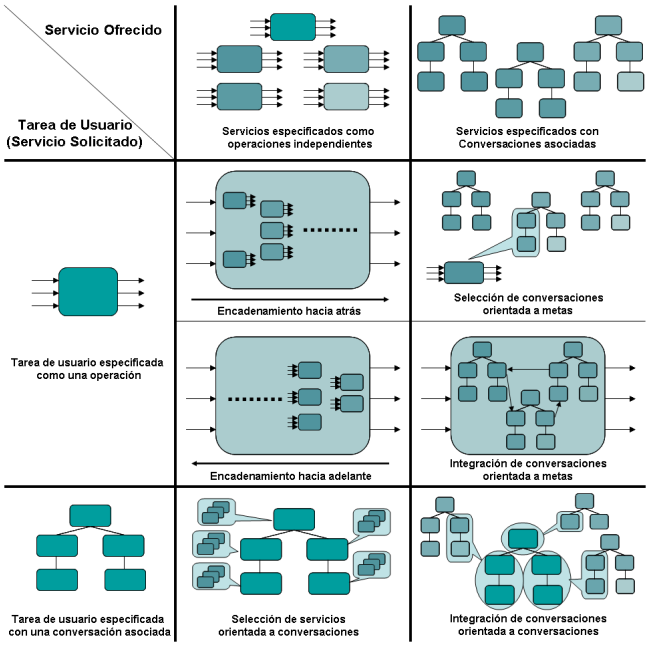


Figura 2.25.: Modelos de Composición.

Los primeros dos modelos de composición denominados encadenamiento hacia atrás y encadenamiento hacia adelante son empleados cuando los servicios de la red y la tarea del usuario están descritas mediante operaciones individuales, sin tener asociadas conversaciones. En estos modelos de composición las operaciones de los servicios del entorno son combinadas basándose en el emparejamiento entre signaturas. El objetivo de esta combinación es obtener un servicio compuesto que sea capaz de realizar la tarea del usuario en términos de especificación de signaturas. El encadenamiento hacia adelante consiste en ir seleccionando los servicios empezando por las entradas del servicio requerido (y precondiciones) y continuar así hasta que la salida (y efectos) del servicio requerido son generados [644]. Por el contrario el método de encadenamiento hacia atrás comenzaría por las salidas (y efectos) e iría generando el flujo hasta llegar a las entradas (y precondiciones) [739]. Aunque estos enfoques permitan combinar servicios sin conocimiento previo en el modelo de combinación (por ejemplo en qué orden ser van a encadenar los servicios), su complejidad es muy alta, ya que todas las posibilidades de encadenamiento tiene que ser investigadas, por lo que el coste computacional puede ser muy elevado. Además, como el proceso es ciego (las operaciones son encadenadas solo en base a la compatibilidad de las signaturas), existe cierta incertidumbre en lo relativo a como es manipulada la información del usuario. Sin embargo, existen algunos enfoques que asumen un conocimiento previo descrito en forma de reglas de descomposición, con el objetivo de orientar el proceso de encadenamiento [870].

Los tercer y cuarto modelos de composición denominados selección de conversaciones orientada a metas e integración de conversaciones orientada a metas, investigados respectivamente en [431] y en [148] que está basado en [99], asumen que los servicios del entorno están descritos mediante conversaciones y las tareas de usuarios están descritas mediante operaciones individuales. En el

primero de los modelos el denominado Process Query Language es empleado para encontrar aquellas conversaciones que contienen fragmentos que satisfacen la tarea del usuario requerida. Por lo tanto, se asume implícitamente que la tarea del usuario puede ser realizada por un único servicio al contrario que el segundo modelo de composición, que tiene como objetivo integrar un conjunto de conversaciones de servicios para realizar la tarea del usuario (que está descrita como una operación simple). En el segundo modelo de composición las conversaciones de un conjunto de servicios seleccionados son integradas para dar como resultado una composición que verifica ciertas propiedades como puede ser la ausencia de bloqueos además de conformar la tarea del usuario consumiendo todas sus entradas y generando todas las salidas requeridas. En los dos modelos de composición sigue existiendo un nivel de incertidumbre debido a la manera de combinar los servicios de la red. De hecho, verificando que la composición resultante está libre de bloqueos no quiere decir que los datos del usuario no hayan sido transformados de manera inadecuada empleando operaciones no apropiadas (por ejemplo empleando operaciones que un usuario no hubiera empleado para cumplir el objetivo) para cumplir con la especificación de la tarea del usuario.

El quinto modelo de composición, denominado selección de servicios orientada a conversaciones, donde la tarea a realizar está descrita como una conversación y los servicios del entorno son descritos mediante conjuntos de operaciones independientes. Este modelo ha sido muy empleado en entornos internet [42], [514] y sobre todo en la composición dinámica de servicios en entornos de computación ubicua [533], [664]. En este modelo, las operaciones de los servicios ofrecidos son emparejadas contra las operaciones requeridas en la tarea del usuario. Los diferentes enfoques que emplean este modelo se diferencian en la expresividad del lenguaje empleado para la descripción de servicios además del algoritmo de emparejamiento. Como este enfoque es capaz de satisfacer la conversación descrita en la tarea del usuario consigue que la composición resultante sea más ajustada a las necesidades del usuario todo ello siendo el flujo de información controlado, ya que es el descrito por el usuario.

El último modelo de composición, denominado integración de conversaciones orientada a conversaciones, donde tanto los servicios del entorno como la tarea del usuario están descritos mediante conversiones o comportamientos complejos. En este modelo, investigado en [117], las conversaciones de los servicios de red son integradas con el objetivo de satisfacer y cumplir la conversación de la tarea requerida por el usuario. Este modelo naturalmente incluye al modelo anteriormente descrito, ya que en caso de que existan servicios de red descritos de manera simple (mediante operaciones) el algoritmo los usará en combinación con las conversaciones de otros servicios para construir la tarea del usuario. Además del trabajo de Berardi en este mismo enfoque se encuentra también el trabajo que está realizando Ben Mokhtar en INRIA Roquencourt [564], [563].

2.3.1.3 Técnicas de planificación de IA

En este punto se van a explicar algunas de las técnicas de planificación de IA comentadas anteriormente.

2.3.1.3.1. Planificación HTN. La planificación HTN (Hierarchical Task Network) [591] permite descomponer una tarea compleja en un conjunto de tareas más simples. Esta planificación define el concepto método. Un método define las condiciones que se deben cumplir por una tarea para poderla descomponer en un conjunto de sub-tareas. Las tareas que pueden ser descompuestas se llaman tareas compuestas. En cambio, las tareas que no pueden ser descompuestas se llaman tareas primitivas. Para poder ejecutar la planificación HTN se debe crear una teoría de dominio. Esta teoría consiste en un conjunto de métodos y operadores que habilitan la creación de planes. Un método establece la relación entre una tarea compleja, sus precondiciones y las sub-tareas en las que puede descomponerse. Cuando el algoritmo tiene que descomponer una tarea, las precondiciones son comprobadas para probar que pueden ser aplicadas al estado actual del entorno.

Esta planificación es utilizada en [656] junto con un razonamiento LCW [211]. El uso de LCW permite evitar la redundancia en el acceso a la información durante la composición de un servicio. La base de conocimiento almacena en conocimiento que el necesita para empezar la planificación y la información que es producida en el proceso. Cuando un agente, que utiliza planificación HTN, quiere descomponer una tarea primero debe comprobar sus precondiciones. El uso de razonamiento LCW redice el número de peticiones a la base de conocimiento gracias al uso de la suposición de

que la información está completa.

El planificador tiene un conjunto de acciones especiales que permiten obtener información durante el proceso de planificación. Esta información no está incluida en el plan pero es obtenida por el planificador usando un mediador semántico. La adicción de este mediador semántico reduce el tiempo de planificación debido a que el número de acciones que el planificador tiene que realizar para obtener la solución es reducido. Esta solución utiliza OWL-S para realizar la descripción de las acciones del entorno.

Una solución similar puede ser encontrada en [870]. Este trabajo propone el uso de un planificador HTN llamado SHOP2 (Simple Hierarchical Order Planner) que ha sido seleccionado porque planifica las acciones de una tarea en el mismo orden que la ejecución. Esto permite conocer el estado del entorno en cualquier momento del proceso de la planificación y ofrece la posibilidad de incluir razonamiento e inferencia durante el proceso de evaluación de precondiciones.

2.3.1.3.1.1. Planificación utilizando un lenguaje de programación lógica. Otras propuestas para la composición de servicios usan lenguajes de programación lógica. La composición es realizada mediante un conjunto de procedimientos reutilizables descritos este tipo de lenguajes.

Un ejemplo de aproximación a esta es mostrado en [546]. donde los autores proponen la utilización de GOLOG [481]. GOLOG es un lenguaje para agentes inteligentes cuyo objetivo es el razonamiento de las acciones y los cambios del entorno. Este lenguaje fue desarrollado sin tener en cuenta las tareas relacionadas con la recolección de información. Aun así, los autores del trabajo consideran que estas operaciones son necesarias para la composición de servicios, es más, han propuesto una extensión de GOLOG que las incluye.

El fin de esta solución es conseguir crear procedimientos reusables que puedan ser compartidos y utilizados por todos los usuarios del entorno. Los autores proponen compartir estos procedimientos dentro de la descripción de servicios en OWL-S. Cuando un agente recupera uno de estos procedimientos genéricos, este lo debe amoldar a las necesidades de los usuarios teniendo en cuenta el estado y de los servicios que están disponibles en el entorno. Cuando el procedimiento esta creado y personalizado este podrá ser ejecutado. En este punto existe una necesidad de obtener la información del entorno mediante la evaluación de precondiciones y de los efectos de la composición del servicio.

Por otro lado, en el momento de ejecución del servicio es donde aparecen la mayoría de los problemas. Los autores de esta proposición utilizan una solución que mezcla un intérprete online y otro offline. Los intérpretes online son los que realizan la adquisición de la información durante el proceso de razonamiento. En cambio, los intérpretes offline únicamente utilizan la información de la que disponen en su base de conocimiento. El intérprete propuesto obtiene la información del entorno cuando es necesitada; aunque, aquellas acciones que afectan al entorno son simuladas y únicamente son ejecutadas en el plan final. Esta solución reduce el espacio de búsqueda cuando se razona y mantiene la posibilidad de obtener información mientras se ejecuta y razona el plan.

La principal desventaja de esta solución es que el plan está pensado para ser inalterable, por lo que, los cambios que se produzcan en el entorno pueden afectar a la validez de este y pueden convertirlo en inutilizable.

2.3.1.3.1.2. Planificación utilizando un Motor de Reglas Otra solución para la composición de servicios puede ser encontrada en [739]. En este caso, se utiliza una herramienta para la construcción de reglas basada en sistemas expertos. La herramienta utilizada es JESS [373] la cual utiliza el algoritmo RETE [7] para realizar la unión entre reglas y hecho de la base de conocimiento. Este algoritmo es mucho más rápido que otros basados en sentencias if-else en un bucle.

Los autores identifican los pasos que son necesarios para realizar la composición de servicios usando un motor de inferencia de este tipo. El primer paso es la realización de la traducción entre descripciones semánticas y las tripletas verbo-sujeto-objeto utilizadas por JESS. Las tripletas generadas en este paso son introducidas en la base de conocimiento como nuevos hechos. El siguiente paso es la construcción de operadores que correspondan con servicios atómicos. Estos operadores son esenciales para el uso del algoritmo RETE.

La composición de servicios es un proceso iterativo formado por dos pasos: el primero, búsqueda de aquellos servicios que satisfagan el objetivo existente y añadirlos a la composición; el segundo, conversión de las entradas y las precondiciones de todos los servicios añadidos a la composición.

Este proceso será repetido mientras sigan existiendo más servicios. La composición será considerada exitosa cuando únicamente los objetivos que corresponden a entradas y precondiciones externas no han sido resueltos. Las entradas y precondiciones externas son aquellas que pueden ser resueltas por el usuario o agente utilizando la información disponible. En cambio, si cuando la composición termina existen objetivos que dependen de precondiciones y/o entradas consideradas internas, aquellas que solo pueden ser resueltas por otros servicios, se considerará que la composición está incompleta y no podrá ser ejecutada.

La solución para la composición de servicios da por hecho que todas las descripciones de servicios han sido obtenidas y almacenadas en la base de conocimiento. Sin embargo, puede ser necesario el descubrimiento de nuevos servicios durante el proceso de planificación. Una posible solución consiste en la división del proceso de composición en diferentes partes. Inicialmente se deberían descubrir los servicios teniendo en cuenta sus entradas y salidas. Los dispositivos semánticos encontrados son unidos utilizado encaminamiento hacia delante (salidas unidas con entradas) o mediante encadenamiento hacia atrás (entradas unidas con salidas). Los grafos que son producidos como resultados del el encadenamiento hacia adelante empiezan desde las entradas provistas por el usuario, mientras que, en los grafos formados mediante el encadenamiento hacia atrás la composición será iniciada desde las salidas que desea el usuario. Además, como resultado del encadenamiento hacia adelante un conjunto de grafos parciales serán construidos, que serán utilizados para crear el grafo final utilizando técnicas de planificación de IA.

2.3.1.4 Enfoques para la ejecución de la composición

Actualmente existen dos enfoques principales para la ejecución de composiciones de servicios: Por una parte está la orquestación y por otra la coreografía. Estos conceptos definen en cierta medida cómo se establecen las relaciones entre los diferentes servicios para obtener un comportamiento coordinado de los mismos. Las diferencias se pueden resumir del siguiente modo:

2.3.1.5 Orquestación

Un proceso se puede considerar una orquestación de servicios cuando es controlado totalmente por una única entidad. Este proceso define completamente las interacciones con los servicios componentes y la lógica requerida para conducir correctamente esas interacciones (ver Ilustración 2.26). Este tipo de proceso puede entenderse como privado y ejecutable ya que sólo la entidad que está orquestando el proceso conoce el flujo de control e información que sigue el proceso que se está orquestando. De esta manera se crea un proceso que utiliza diferentes servicios manipulando la información que fluye entre ellos, convirtiendo, por ejemplo, los datos de salida de algunos servicios en datos de entrada de otro. Aquí, cada entidad que participa implementa y controla su propio proceso [631], [632].

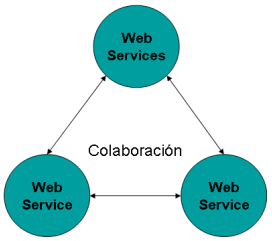


Figura 2.26.: Orquestación de Servicios Web [631].

La descripción anterior concuerda con una orquestación centralizada en la que una entidad es la que controla al resto y ordena cuando deben ejecutarse, pasando todo el flujo de información por él. Pero, en los últimos años se ha propuesto otro tipo de orquestación, la denominada orquestación distribuida o descentralizada [175], [131] que en vez de disponer de un nodo central o director que orquesta al resto de servicios se pretende dividir el workflow en diferentes partes para así ejecutarlo de manera distribuida. Algunas de las ventajas de la orquestación descentralizada son las siguientes:

- No hay coordinación central que puede llegar a ser el cuello de botella del sistema
- Distribuyendo los datos se reduce el tráfico de red y se mejora el tiempo de transferencia.
- Distribuyendo el control se mejora la concurrencia.

2.3.1.5.1. Coreografía Un proceso es una coreografía de servicios cuando define las colaboraciones entre cualquier tipo de aplicaciones componentes, independientemente del lenguaje o plataforma en el que estén definidas las mismas (ver Ilustración 2.27). Un proceso de coreografía no es controlado por uno solo de los participantes. A diferencia de la orquestación, la coreografía puede verse como un proceso público y no ejecutable. Público porque define un comportamiento común que todas las entidades participantes deben conocer y no ejecutable porque está pensado para verse más bien como un protocolo de negocio que dicta las reglas para que dichas entidades puedan interactuar entre sí [631], [632].

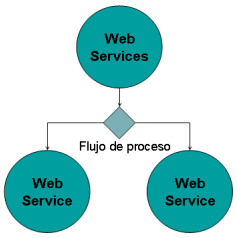


Figura 2.27.: Coreografía de Servicios Web [631].

2.3.2 Análisis de los principales enfoques de composición de servicios en entornos de computación ubicua

En las siguientes páginas se describen los principales enfoques de composición en computación ubicua, pero previamente se establecen las características a analizar en cada uno de ellos.

2.3.2.1 Criterios para la evaluación de enfoque de composición para entornos de computación ubicua

A continuación son descritos los cuatro grupos de características que se han definido para la clasificación de los enfoques existentes, así como los resultados de las comparativas de los mismos:

2.3.2.1.1. Expresividad del Lenguaje. En este grupo se encuentran aquellas características relacionadas con la expresividad del lenguaje empleado para describir los servicios así como las tareas definidas por el usuario (ver Tabla 2.17):

Lenguaje de modelado de servicios (mod): Lenguaje empleado para describir las características funcionales como no-funcionales de los servicios.

Semántica (sem): Esta característica indica si el lenguaje empleado para describir los servicios ofrece soporte semántico para describir tanto los atributos funcionales como no-funcionales de los servicios para su posterior empleo en las fases de descubrimiento y composición.

Mecanismo para representar la tarea del usuario (task): Este criterio representa el mecanismo empleado para describir el servicio o tarea requerido por el usuario. Por ejemplo, este puede ser descrito como una operación, una conversación, como la meta a conseguir, etc.

Mecanismo para representar los servicios publicados (pub): Este criterio representa el mecanismo empleado para describir el servicio publicado. Por ejemplo, operaciones, conversaciones, etc.

Approach	mod	sem	task	pub
[860], [131]	WSDL, UPnP	Syntactic	Workflow	Simple Services
[167]	OWL-S	Semantic	Composite FSM	FSM adaptive behaviour
[235]	WSDL	Syntactic	Workflow	Simple Services
[833]	WSDL	Syntactic	Context information + Goal	Simple Services
[140]	Key-Value pairs	Syntactic	Workflow	Simple Services
[657]	OWL-SC	Semantic	Simple Service	Simple Services
[437]	Key-Value pairs	Syntactic	Workflow	Simple Services
[372]	OWL-S	Semantic	Workflow	Simple Services
[557]	OWL-S	Semantic	Workflow	Simple Services
[656]	OWL-S	Semantic	Workflow	Simple Services
[404]	UPnP	Syntactic	Workflow	Simple Services
[645]	OWL-S	Semantic	Workflow (3rd party service providers)	Simple Services
[479]	Not specified	Semantic	Simple Service	Simple Services
[510]	WSDL	Syntactic	Workflow (State Chart Diagrams)	Simple Services
[115]	WSDL	Syntactic	Message Sequence Charts	Simple Services
[573]	WSDL	Syntactic	Simple Services	Simple Services
[740]	OWL-S	Semantic	Workflow	Simple Services
[563], [561]	COCOA-L (OWL-S)	Semantic	Workflow	Workflow
[176], [178]	DAML-S	Semantic	Workflow (Description-level Service Flow)	Simple Services
[533], [756]	OWL-S	Semantic	Workflow	Simple Services
[595], [594]	OWL-S	Semantic	Goal (desired state)	Simple Services
[666]	DAML+OIL	Syntactic	Goal	Simple Services
[664]	WSDL	Syntactic	Goal (converted to a Workflow)	Simple Services
[814], [815]	OWL-S	Semantic	Goal (converted to a abstract plan)	Simple Services

Tabla 2.17.: Resultados de la comparativa del grupo de expresividad del Lenguaje.

2.3.2.1.2. Modelo de Composición En este grupo se encuentran aquellas características relativas al método empleado para la creación de la composición o plan (síntesis) para su posterior ejecución (ver Tabla 2.18):

Técnica de creación de la composición (tech): Técnica empleada para conseguir el objetivo/meta definido por el usuario. Por ejemplo, técnicas de planificación de IA, de Workflow, de Data-Mining, etc.

Lenguaje de composición (lang): Lenguaje empleado para representar y ejecutar la composición que previamente ha sido creada mediante la técnica de composición. Por ejemplo, scripts, bpel4ws, etc.

Sensibilidad al contexto (cont): Indica si la técnica de composición tiene en cuenta la información contextual (por ejemplo: localización del usuario, perfil del usuario, localización de los dispositivos, etc.) del entorno durante el proceso de creación del plan.

Sensibilidad a la calidad de servicio (QoS): Indica si la técnica de composición tiene en cuenta la calidad de servicio (por ejemplo: latencia, memoria utilizada, etc.) durante la creación del plan.

Intervención del usuario (user): Indica si el usuario puede participar durante el proceso de creación de la composición, por ejemplo seleccionando los servicios en caso de conflicto, etc.

2.3.2.1.3. Modelo de Ejecución. En este grupo son descritas aquellas características relacionadas con el mecanismo empleado para la ejecución del plan creado en el grupo anterior (ver Tabla 2.19):

Orquestación/Coreografía (comp): Este criterio especifica si la composición es ejecutada de manera centralizada (orquestación) o bien de manera distribuida (coreografía)

Replanificación en la ejecución (repla): Este criterio especifica si el mecanismo de ejecución soporta la replanificación (por ejemplo debido a un cambio contextual) de la composición durante la ejecución de la misma.

Binding dinámico o estático (bind): Este criterio especifica si los servicios empleados en la composición están definidos de manera estática o bien dinámica, es decir, si los servicios de la composición están definidos previamente o pueden variar.

2.3.2.1.4. Entorno de Ejecución. En este grupo son descritas aquellas características relacionadas con el entorno de ejecución en el que operan los servicios, se realiza la composición y es ejecutada la misma (ver Tabla 2.20):

Arquitectura (arch): Este criterio especifica la arquitectura empleada en el enfoque de composición. Ésta puede estar basada en una arquitectura existente o puede ser una nueva.

Mecanismo de descubrimiento de servicios (disc): Tecnología o protocolo empleado para el descubrimiento de los servicios publicados y para la fase de creación de la síntesis de proceso.

Dispositivos que soportan la ejecución de la composición (device): Este parámetro representa que tipo de dispositivos son los que soportan la ejecución de la composición.

2.3.2.2 Análisis de los enfoques de composición

Tomando como base los resultados de la comparativa a continuación se realiza un breve resumen destacando las características más importantes de los enfoques planteados.

Propuesto en [533], [756], Task Computing orienta la composición de servicios hacia lo que quiere hacer el usuario, en lugar de hacer que el usuario se adecue a los servicios del entorno. De esta manera se ofrece un entorno completo orientado al usuario final basado en descripciones DAML-S. También Ni [595], [594] pretende abstraer al usuario de la complejidad en la composición por medio de OSOA defendiendo, como muchos otros autores [666], técnicas de AI planning para resolver el problema de la composición de servicios. En la propuesta presentada en [860], [131] por Nokia Research Center Cambridge, se propone una arquitectura en la cual se deja en manos de la intuición del usuario final la elección de los dispositivos que se quieren utilizar en la composición y se utiliza el algoritmo de composición DSF1 para buscar las posibles combinaciones entre los dispositivos seleccionados. Se trata de un sistema de composición semi-automático puesto que cuenta con la intervención del usuario durante la composición.

Approach	tech	lang	cont	QoS	user
[860], [131]	AI Planning - Depth First Search 1	Scripts	No	No	Yes
[167]	AI Planning - Finite State Machines	OWL-S Proccess	No	No	No
[235]	ECF+Mobile Agents	Executable Choreography Lang. (ECL)	Yes	No	No
[833]	AI Planning - HTN - SHOP2	BPEL4WS	Yes	No	No
[140]	Workflow (Service Selection)	Not specified	Yes	No	No
[657]	AI Planning - HTN	Directed Acyclic Graph (DAG)	Yes	Yes	No
[437]	Workflow (Service Selection)	XML	Yes	Yes	No
[372]	Workflow (Service Selection)	Not specified	Yes	No	Yes
[557]	Workflow (Service Selection)	XML	No	Yes	No
[656]	AI Planning - HTN + LCW	Not specified	Yes	No	No
[404]	Workflow	Functional Task Description (FTD)	Yes	Yes	Yes
[645]	Workflow (Service Selection)	OWL-S Proccess	Yes	No	No
[479]	Data-Mining	Not specified	Yes	No	No
[510]	Agent Conversation based	State Chart Diagrams	Yes	Yes	No
[115]	Workflow (Service Selection)	BPEL4WS	Yes	Yes	No
[573]	Ubiquitous Coordination Model	Not specified	Yes	Yes	No
[740]	AI Planning - Backward chainning - STRIPS	OWL-S Proccess	Yes	No	No
[563], [561]	Workflow - Finite State Automatas	COCOA-L (OWL-S)	Yes	Yes	Yes
[176], [178]	Workflow (Service Selection)	Description-level Ser. Flow (DAML-S)	No	No	No
[533], [756]	Workflow (Service Selection)	Not specified	Yes	No	Yes
[595], [594]	AI Planning - STRIPS	Not specified	Yes	No	Yes
[666]	AI Planning - Blackbox planner - STRIPS	Lisp	Yes	No	Yes
[664]	Workflow (Service Selection)	BPEL4WS	Yes	No	Yes
[814], [815]	Worfklow + AI Planning - CSP (for validation)	OWL-S Proccess	Yes	No	No

Tabla 2.18.: Resultados de la comparativa del grupo de Modelo de Composición.

Approach	comp	repla	bind
[860], [131]	Orchestration	No	Dynamic
[167]	Orchestration	Yes	Dynamic
[235]	Choreography	No	Dynamic
[833]	Orquestration	No	Dynamic
[140]	Orchestration	No	Dynamic
[657]	Orchestration	No	Dynamic
[437]	Orchestration	Yes	Dynamic
[372]	Orchestration	No	Dynamic
[557]	Orchestration	No	Dynamic
[656]	Orchestration	No	Dynamic
[404]	Orchestration	No	Dynamic
[645]	Orchestration	No	Dynamic
[479]	Orchestration	No	Dynamic
[510]	Choreography (Agent based)	No	Dynamic
[115]	Orchestration	Yes	Dynamic
[573]	Orchestration	No	Dynamic
[740]	Orchestration	No	Dynamic
[563], [561]	Orchestration	No	Dynamic
[176], [178]	Distributed Orchestration	Yes	Dynamic
[533], [756]	Orchestration	No	Dynamic
[595], [594]	Orchestration	No	Dynamic
[666]	Orchestration	Yes	Dynamic
[664]	Orchestration	No	Dynamic
[814], [815]	Orchestration	No	Dynamic

Tabla 2.19.: Resultados de la comparativa del grupo de Modelo de Ejecución.

Approach	arch	disc	device
[860], [131]	End-User Programming Framework (EUP)	UPnP, WS, Jini	Mobile Phones
[167]	Not specified	Not specified	Not specified
[235]	Executable Choreography Framework (SOAP)	Not specified	Embedded Systems
[833]	SOAP	UDDI	Not specified
[140]	IPOJO over OSGI	OSGI	Gateway (Home)
[657]	SOAP	Not specified	Not specified
[437]	Smart Space Middleware over OSGI	OSGI	Gateway
[372]	Jini	Reggie (Jini lookup service)	Not specified
[557]	Home Appliances Integration Unit (HAIU)	Not specified	Home entertainment syst.
[656]	OntoPlan	Not specified	Not specified
[404]	UPnP	UPnP	Resource Constrained devi.
[645]	OSGI (UPnP & Jini)	OSGI	Gateway (Home)
[479]	Service Provisioning Middleware (COSEP)	Not specified	Not specified
[510]	SOAP	UDDI	Not specified
[115]	SOAP	Not specified	Not specified
[573]	CB-Sec Framework	LW-UDDI	Embedded PCs
[740]	Multi-agent architecture and SOAP	Not specified	Not specified
[563], [561]	SOAP	Not specified	Not specified
[176], [178]	Architecture developed for the approach	Group Based Service Disc.	Mobile Devices
[533], [756]	Task Computing Environment (SOAP)	UPnP	PDA
[595], [594]	SOAP	WS-Discovery, UPnP	Mobile Devices
[666]	GAIA	Discovery module of GAIA	Not specified
[664]	SOAP	UDDI, Multicast DNS	iPaq
[814], [815]	SOAP	Not specified	Not specified

Tabla 2.20.: Resultados de la comparativa del grupo Entorno de Ejecución.

En [404] el objetivo se define como end-to-end, ej."link (video-source) with (display)z el sistema busca los servicios apropiados para unir los dos extremos con un algoritmo de búsqueda simple. Entre las posibles soluciones se elige la más adecuada basándose en policies y en la disponibilidad de los recursos. Vukovic y Robinson [833], plantean la composición como un problema de planificación usando HTN con SHOP2. La descripción de la composición se define usando BPEL4WS. Esta descripción se traduce a objetivos SHOP2 y el planificador resuelve un plan para estos objetivos. Finalmente, este plan es traducido de nuevo a BPEL4WS. Cambios en el contexto pueden provocar la replanificación de la composición durante su ejecución, haciendo que la aplicación se adapte a estos cambios contextuales, el mismo objetivo perseguido en [573].

Qiu y otros proponen en [657] utilizar OWL-SC, una extensión de OWL-S para incluir información contextual en la descripción del servicio, de manera similar a la propuesta en [774]. También proponen HTN como método de planificación, pero introducen la idea de "librería de planes"para poder almacenar planes y no tener que recalcular más de una vez un plan determinado. Ontoplan [656] también emplea HTN, aunque lo combina con Local Closed World reasoning (LCW) para evitar redundancias en la composición.

El uso de workflows como herramienta para componer servicios también es defendido por muchos autores como Ranganathan y McFaddin [664], Bellur y Narendra [115], etc. Por ejemplo Ben Mokhtar y otros [564], [561] proponen el sistema COCOA donde las tareas de los usuarios están descritas mediante procesos abstractos OWL-S en donde la composición final se consigue mediante un algoritmo de emparejamiento que construye la composición en base a fragmentos de las conversaciones ofrecidas por los servicios utilizando Finite State Machines (FSM) para modelizarlas. Vallee y otros [814], [815] y Maamar y otros [510] presentan un enfoque que combina técnicas de sistemas multi-agente con Servicios Web Semánticos para proporcionar composición dinámica de servicios sensible al contexto. En [479], se emplean técnicas de data-mining considerando información de contexto y el historial de uso de los servicios, para definir nuevas composiciones de servicios.

La representación de los servicios por medio de ontologías es defendida también en [372], [557], [740]. Otros como Lee y otros en [437], emplean OSGi como especificación de los servicios de forma que una composición expresada en forma de grafo y representada en formato XML, se construye como un nuevo servicio OSGi, utilizando servicios existentes. También en [140] y en [645] se emplea OSGi, proponiendo como novedad en este último dejar en manos de terceros más expertos, en lugar del usuario, la composición de los servicios.

A diferencia de las propuestas estudiadas hasta ahora, en la que la responsabilidad de la composición era centralizada, en cambio, Chakraborty y otros [176], [178] proponen protocolos descentralizados para componer servicios expresados en DAML-S. De forma similar, la gran mayora de los trabajos analizados utilizan orquestación centralizada para la ejecución de la composición en la que un coordinador es el encargado de dirigir el control del flujo.

Existen varios trabajos que describen algoritmos para la composición que utilizan descripciones semánticas de los servicios destacan los siguientes: En [167] Carey y otros proponen extender la planificación IA, criticada por necesitar replanificación para adaptarse a los cambios en el contexto, por medio de máquinas de estados finitos (FSM) que modelizan el comportamiento adaptativo (no funcional) que debe tener la composición resultante, de forma que estas FSM se encargan de adaptarse a los cambios en el contexto sin necesidad de replanificación. sEl ECL (Executable Choreography Language) descrito en [235] permite inyectar modificaciones en el flujo de control de un servicio sobre el ECF (Executable Choreograhy Framework) de forma no invasiva, esto es, sin alterar manualmente el flujo original del servicio.

2.3.3 Conclusiones

Del análisis de los sistemas para entornos ubicuos anteriormente mencionados y de los diferentes métodos para composición expuestos se pueden extraer las siguientes conclusiones:

Los sistemas o métodos de workflow son mayoritariamente empleados en los casos en donde el peticionario ya tiene definido un modelo de proceso, pero en donde es necesario que un programa encuentre automáticamente los servicios atómicos para rellenar dicho proceso. Y en cambio las técnicas de planificación IA son empleadas cuando el peticionario no tiene un modelo de proceso pero se tiene un conjunto de preferencias y restricciones, de esta manera el planificador basándose

en dichas preferencias y restricciones realiza la planificación. De estos dos métodos es difícil decir cual es más adecuado a un entorno inteligente, ya que puede variar en función del tipo de sistema que se desee desarrollar y en función de los casos de uso a emplear.

Casi en la práctica totalidad de enfoques estudiados en el presente capítulo la ejecución de la composición es centralizada es decir, la composición la realiza un dispositivo y no es una composición distribuida realizada de manera colaborativa. Una composición distribuida y colaborativa, en donde el número de dispositivos y servicios es dinámico se asemeja más a la visión de la computación ubicua planteada por Weiser [852].

Además de ser centralizados, varios de los sistemas emplean dispositivos con capacidades avanzadas de computación para la composición, por ejemplo, empleando PDAs y PCs. Pero en entornos ubicuos la mayoría de dispositivos que se encuentran en el mismo no tiene gran capacidad de procesamiento, ya que deben ser dispositivos pequeños y por lo tanto con una capacidad de procesamiento adecuada al mismo. Ello trae consigo un problema, ya que si se pretende emplear una composición distribuida con dispositivos con recursos limitados es posible que las tecnologías y estándares actuales no puedan ser empleados, por lo que tal vez tengan que ser adaptados para ser aplicables a dicha casuística.

En muchos de los sistemas la interacción con el usuario es muy alta y necesaria, pero creemos que esto no debe de ser así, el entorno debe de conocer el perfil y el comportamiento que el usuario desea que adopte dicho entorno y adaptarse a l, siendo prácticamente inexistente la interacción con el usuario. Para ello es necesario que el usuario previamente defina que tareas o que comportamientos quiere realizar, definiendo las tareas mediante un lenguaje especialmente definido para ello. De esta manera es posible, por ejemplo, crear un dispositivo inyecte en el entorno las tareas que desea realizar el usuario cuando este entre en una habitación para hacer conscientes a los dispositivos de cuáles son las necesidades del usuario y para que los dispositivos cooperen para componer tareas de alto nivel para satisfacer las necesidades del usuario.

En las propuestas planteadas los lenguajes empleados mayoritariamente son BPEL4WS y OWL-S, en cambio, no se han realizado trabajos relativos a composición para computación ubicua con propuestas como WSMO, SWSF, WSDL-S y SAWSDL, esto puede ser debido a que estas últimas propuestas son de mas reciente creación y OWL-S y BPEL4WS son tecnologías más maduras y desarrolladas con mas herramientas de desarrollo y APIs. Por lo que hay una va abierta en el posible empleo de estas nuevas propuestas de Servicios Web Semánticos en los entornos de computación ubicua o su posible adaptación para con recursos limitados (sistemas embebidos).

2.3.3.1 Áreas que requieren profundización técnica

Los dispositivos, aplicaciones y sistemas actuales carecen de los niveles de reactividad y sensibilidad al contexto descritos en los escenarios propuestos por Lassila [469], Weiser [852], Aarts [27] o el ISTAG [340], ya que son sistemas con dispositivos simples que tienen pautas de comportamiento básicos, como pueden ser abrir unas puertas cuando se detecte la presencia cercana de una persona, encender una luz cuando se detecta la proximidad de una persona, redireccionar una llamada al teléfono más cercano que se encuentra una persona, etc., o bien son sistemas centralizados en donde un ordenador central es quien tiene toda la inteligencia y los dispositivos que se encuentran alrededor solamente se encargan de recibir órdenes y actuar, sin tener iniciativa propia [828].

Para conseguir comportamientos más inteligentes, avanzados, espontáneos y con capacidad de razonamiento y aprendizaje autónoma es necesario que los dispositivos de los entornos inteligentes tengan cierta inteligencia, pero también es indispensable que sean ligeros, es decir, pequeños y que consuman poca energía, para así embeberlos en los objetos cotidianos. Nosotros pensamos que los entornos inteligentes sólo pueden ser construidos sobre la base de sistemas embebidos y no sobre sistemas en donde son necesarios grandes dispositivos como PCs, que crean escenarios irreales y artificiales alejados de la visión de la computación ubicua.

El problema de estas dos variables es que son inversamente proporcionales, ya que si se pretende que los dispositivos sean ligeros y pequeños, nos encontramos con que tenemos que emplear sistemas con muy poca capacidad de procesamiento, lo que hace que sea muy difícil desarrollar dispositivos inteligentes, y por el contrario si queremos que el dispositivo sea inteligente necesitamos de elementos con mayor capacidad de procesamiento y consumo de energía que hacen que tengamos un dispositivo grande, perdiendo por tanto la ligereza. Es por ello que nos encontramos en el punto

en el que hay que encontrar el equilibrio adecuado entre ambas variables, o bien decantarse por alguna de las dos sacrificando a la otra.

Creemos que es posible un término medio, en donde el sistema sea ligero, es decir, sea pequeño y consuma poca energía, pero a la vez tenga inteligencia, es decir, realice razonamiento sobre el contexto del usuario, los inputs percibidos y los objetivos deseados y sea capaz de comunicarse y coordinarse con otros dispositivos que se encuentren en el entorno para componer tareas más complejas. Y para ello creemos necesario establecer una cadena de valor que proporcione inteligencia al entorno, para que actúe con reactividad y sensibilidad al contexto basándose en las preferencias de los usuarios. La cadena de valor resultante es la siguiente: Sensorización y perfiles de usuario->Percepción de contexto (basada en ontologías)->Razonamiento (basado en reglas)->Composición (basada en estrategias)->Operación->Dispositivos y aparatos. Paralelo a esta cadena de valor estará el aprendizaje inicial que determinara las reglas iníciales y la adaptación posterior, es decir, el aprendizaje permitirá crear el contenido de las reglas, los perfiles de usuario, etc.

La posibilidad de instalar las capacidades de la Web Semántica en plataformas embebidas supone un desafío importante. Ya que pese a que se hayan realizado intentos para aplicar estas tecnologías para obtener entornos más reactivos y conscientes del usuario, los resultados obtenidos han sido desiguales, por lo que la adaptación de las nuevas tecnologías de la web semántica para su uso en plataformas embebidas para entornos inteligentes supone un reto.

Consideramos que los conceptos claves empleados en las tecnologías de Servicios Web Semánticos para la interoperabilidad y composición automática de procesos empresariales pueden ser extrapolados para obtener el mismo efecto en procesos ambientales, es decir, colaboración automática de servicios ambientales para obtener un comportamiento combinado del entorno por el bien del usuario.

Por ejemplo, si un usuario se encuentra viendo la TV con la luz ambiente apagada y suena el teléfono, se podrá efectuar un comportamiento coordinado del entorno y sus dispositivos que resulten en que el número llamante aparece superpuesto en la pantalla del TV, se afecta una disminución del volumen del altavoz del mismo y un encendido de las luces para facilitar la atención de la llamada y la subsiguiente conversación. En este caso existen varios servicios en los dispositivos del entorno (superposición de texto, ajuste de volumen, encendido y apagado de la luz) que deben coordinarse y colaborar como respuesta al evento de llamada entrante en el teléfono.

Una infraestructura de Servicios Web Semánticos Ambientales permitirá diseñar y desplegar este tipo de escenarios y configurarlos a la medida del usuario proporcionándole un modelo más avanzado y asistivo de interacción con el entorno. Por ello consideramos el estudio y experimentación en este campo como un desafío tecnológico a abordar y puede requerir:

- Adaptación de estándares existentes de composición (orquestación / coreografía) para la creación y ejecución de procesos empresariales distribuidos en entornos de computación ubicua (servicios ambientales) donde los dispositivos tienen restricciones respecto a su capacidad de procesamiento y comunicación, ya que los dispositivos están distribuidos en el entorno.

- Desarrollo de especificaciones nuevas para la composición distribuida de servicios ambientales y así cumplir ciertos requisitos particulares que se dan en este tipo de escenarios.

2.4 Descubrimiento de recursos

2.4.1 Introducción

La presente sección tiene como objetivo realizar un análisis crítico del estado del arte relativo al descubrimiento y emparejamiento de servicios en entornos de computación ubicua. Para ello inicialmente se describen cuáles tienen que ser las características principales y los componentes de un mecanismo de este tipo, para después describir los diferentes problemas que hay que abordar en el descubrimiento de servicios. Tras ello se realiza un análisis de los diferentes tipos de emparejamiento semántico y posteriormente se hace un repaso de los dos principales enfoques para descubrimiento de servicios eficiente en entornos de computación ubicua.

2.4.2 Componentes de un mecanismo de descubrimiento de servicios

En general, cualquier mecanismo de descubrimiento de servicios tiene que disponer los siguientes componentes [338]:

- Descripción de servicio: Lenguaje empleado para describir tanto la semántica funcional como la no-funcional de un servicio.
- Selección de servicios: Mecanismo de razonamiento para el emparejamiento (matching, retrieval, matchmaking, etc.) de servicios que consiste en la comparación de pares de descripciones de servicios en términos de la semántica funcional y no-funcional, así como la clasificación de los resultados en base a criterios de emparejamiento y preferencias.
- Arquitectura de descubrimiento: Arquitectura empleada como base para el descubrimiento, haciendo hincapié en la topología de red (centralizada, descentralizada), almacenamiento de la información de servicios (distribución de los servicios, ontologías, registros) y servicios de localización así como la funcionalidad de los agentes implicados (proveedor de servicios, consumidor de servicios, middle agent).

2.4.3 Arquitectura de descubrimiento

El enfoque ideal de descubrimiento para entornos pervasivos tiene que abarcar un amplio rango de factores relativos a interoperabilidad, debido a la heterogeneidad de estos entornos y debido al hecho de que los servicios pervasivos y los clientes potenciales (clientes que realizan las peticiones de servicios) son diseñados, desarrollados y desplegados independientemente. Los protocolos de descubrimiento de servicios permiten que los servicios puedan ser descubiertos en la red, permitiendo de esta manera oportunidades para la colaboración, para así componerse con otros servicios para crear servicios más complejos, cumpliendo las necesidades de las aplicaciones. Muchos protocolos de descubrimiento de servicios han sido propuestos tanto por la industria como por la comunidad científica, como pueden ser UDDI o el Trading Service de CORBA para Internet o SLP y Jini para redes de área local o redes ad-hoc.

En el artículo [891] se realiza una clasificación de los diferentes protocolos de descubrimiento de servicios que realiza una distinción entre los protocolos pull-based y los protocolos push-based. En los protocolos pull-based, el cliente envía la petición al proveedor de servicios (distributed pull-based mode) o a un repositorio de un tercero (centralized pull-based mode) con el objetivo de conocer la lista de servicios compatibles con la petición realizada. En los protocolos push-based, los proveedores de servicio proveen las descripciones de los servicios a los clientes que localmente mantienen la lista de los servicios disponibles en la red.

Los principales protocolos de descubrimiento de servicios, como Jini y SSDP, emplean el enfoque pull-based para el descubrimiento, soportando algunas veces tanto el modo centralizado como modelos de interacción distribuidos (SLP, WS-Discovery). En los protocolos de descubrimiento centralizado tipo pull-based uno o varios repositorios almacenan las descripciones de los servicios disponibles en la red y la localización de los repositorios puede ser conocida (UDDI) o dinámicamente descubierta (Jini). Los repositorios se mantienen actualizados gracias a que los proveedores se dan de baja explícitamente o bien gracias a que dan de baja a los servicios periódicamente debido a que ha expirado la vigencia del mismo. Si existen múltiples repositorios, estos cooperan entre ellos para distribuirse los registros de los nuevos servicios o también para encaminar las peticiones de servicios al repositorio relevante de acuerdo a relaciones preestablecidas.

Pese a que en la actualidad existen muchas soluciones relativas a arquitecturas de descubrimiento de servicios todavía hay retos y problemas a solucionar en los entornos de computación pervasiva, como se verá a continuación:

2.4.3.1 Descubrimiento de servicios multiprotocolo

La heterogeneidad de middlewares hace que surjan problemas de interoperabilidad entre los diferentes protocolos de descubrimiento de servicios (por ejemplo: SLP, SSDP, UDDI). Los protocolos de descubrimiento de servicios actuales no son capaces de interactuar con otros protocolos, ya que emplean formatos y protocolos incompatibles para la descripción de servicios o para las peticiones

de descubrimiento, e incluso tipos de datos o modelos de comunicación incompatibles. En cualquier caso, las características de los entornos inteligentes y los estándares de-facto de algunos de los protocolos existentes hace imposible que surja un nuevo y único protocolo de descubrimiento de servicios.

Diversos proyectos han tratado de buscar soluciones de interoperabilidad [589]-[446], ya que requerir a los clientes y proveedores de servicios que sean capaces de soportar múltiples protocolos de descubrimiento de servicios no es real. Los protocolos de descubrimiento de servicios interoperables típicamente han empleado un modelo común para la representación de los elementos de descubrimiento de servicios (por ejemplo: descripción del servicio, petición de descubrimiento) [149] en vez de realizar mapeos directos entre los protocolos [446], no pudiendo ser este último enfoque muy escalable con un número elevado de protocolos. Más allá, la capa de interoperabilidad debe de estar alojada cerca de la capa de red [149] y traducir eficientemente y transparentemente los mensajes de red entre los diferentes protocolos, o bien proveer de un interface explicito [669] a los clientes o servicios para así ampliar los protocolos existentes introduciendo características avanzadas tales como la gestión del contexto.

2.4.3.2 Descubrimiento de servicios multi-red

La heterogeneidad de la red hace que muchas redes independientes (que se pueden interconectar libremente con los dispositivos multi-radio móviles actuales) estén disponibles para los usuarios en una localización, por lo tanto soluciones innovadoras son necesarias para la diseminación, filtrado y selección de peticiones de descubrimiento servicios y el anuncio de los mismos [669]. Diversos proyectos han investigado el empleo del enrutamiento entre pares (gateways) o a nivel de aplicación (P2P) combinado con filtros inteligentes que proveen de descubrimiento de servicios eficiente y escalable en entornos multi-red.

El protocolo mSLP [890] mejora la eficiencia y la escalabilidad del protocolo de descubrimiento SLP introduciendo mejoras en la malla así como filtros de preferencia para el emparejamiento de los registros. INS/Twine [96] propone una arquitectura P2P escalable donde los servicios de directorio colaboran como pares para distribuir la información de los recursos y para resolver las consultas. GloServ [65] es una arquitectura global de descubrimiento de servicios basada en ontologías que opera tanto en redes de área extensas como en redes de área local empleando una arquitectura híbrida (jerárquica y peer to peer). MUSDAC [669] permite unir dinámicamente redes que se encuentran cerca mediante componentes específicos que proveen de enrutamiento a nivel de aplicación en dispositivos multi-radio, lo que permite la diseminación de peticiones de descubrimiento en el entorno. Un factor clave en el descubrimiento de servicios multi-red es divulgar cambios dinámicos en la red sin que ello suponga compromiso alguno en el procesamiento ni en los recursos de la red debido a la considerable cantidad de información que es procesada e intercambiada.

2.4.3.3 Movilidad en el descubrimiento

El intercambio de descripciones y las peticiones de descubrimiento debe ser seguro, ya que es crucial en entornos inteligentes, especialmente cuando la información está relacionada con información contextual relativa al usuario o al servicio que puede ayudar a conocer información privada del usuario [893]. Muchos mecanismos para asegurar el acceso a la información del servicio han sido propuestos centrándose bien en la encriptación y los derechos de acceso [239] o en la necesidad de equilibrar el acceso y la privacidad [893].

2.4.3.4 Seguridad y privacidad en el descubrimiento

El intercambio de descripciones y las peticiones de descubrimiento debe ser seguro, ya que es crucial en entornos inteligentes, especialmente cuando la información está relacionada con información contextual relativa al usuario o al servicio que puede ayudar a conocer información privada del usuario [893]. Muchos mecanismos para asegurar el acceso a la información del servicio han sido propuestos centrándose bien en la encriptación y los derechos de acceso [239] o en la necesidad de equilibrar el acceso y la privacidad [893].

2.4.4 Descripción de servicio

Además de la arquitectura que permite descubrir los dispositivos del entorno, es necesario definir el lenguaje en el que se van a representar las características funcionales como no-funcionales de los servicios del entorno. Los enfoques de representación basados en semántica ofrecen mayor flexibilidad y expresividad para representar los servicios de entornos ubicuos, por lo tanto en las siguiente líneas se describirán y analizaran las principales iniciativas para la representación de servicio semánticos [805].

2.4.4.1 OWL-S

OWL-S [64] anteriormente denominada DAML-S [446], [621], es una ontología para WS realizada en OWL [544], [754], un lenguaje recomendado por el W3C para representar ontologías y que sean fácilmente procesables por máquinas. El lenguaje OWL proporciona un conjunto de construcciones para representar las relaciones presentes en una ontología. Estas construcciones permiten representar conceptos, herencia, relaciones entre los conceptos, y restricciones sobre los valores que pueden tomar las propiedades de un concepto. OWL proporciona mayores facilidades a la hora de representar el significado y la semántica que otros lenguajes como pueden ser XML y RDF. OWL-S fue la primera propuesta enviada al W3C relativa a SWS, y se remonta al 2 de noviembre del 2004.

OWL-S es, por tanto, una ontología creada a partir de OWL que define los conceptos y relaciones necesarias para describir semánticamente un WS. Las descripciones de servicios se realizan mediante axiomas OWL-DL (uno de los tres sublenguajes de OWL), pero se ha visto que no aporta la expresividad necesaria para la descripción de los servicios, por lo que posteriormente OWL-S ha sido mejorado para que pueda permitir expresiones lógicas de otros lenguajes como pueden ser: DRS [542], KIF [321], SWRL [382] y PDDL [540]. El entorno de ejecución de OWL-S se denomina OWL-S VM [620].

OWL-S constituye una ontología de alto nivel con la cual se construye la descripción semántica de cada servicio. Esta ontología proporciona una clase principal definida como *Service* desde la que se puede acceder al resto de la descripción del servicio. A partir de esta clase se puede obtener el perfil del servicio (*ServiceProfile*), el modelo del servicio (*ServiceModel*) y el soporte del servicio (*ServiceGrounding*):

- **ServiceProfile:** La clase *Service* presenta un perfil de servicio (*Profile*). Este perfil describe qué es lo que el servicio hace. Proporciona información sobre el proveedor, un conjunto de atributos que permiten describir características del servicio y su descripción funcional. Los datos sobre el proveedor del servicio hacen referencia a nombre, administrador, información de contacto, etc. En cuanto a los atributos que permiten describir el servicio, éstos incluyen conceptos como la categoría a la que pertenece el servicio dentro del UNSPSC, su fiabilidad (buena, muy buena, aceptable, etc.) o la puntación del servicio en algún sistema de valoración de WS. La descripción funcional incluye la especificación de que funcionalidad proporciona el servicio y que precondiciones deben satisfacerse para obtener el resultado deseado. OWL-S presenta dos aspectos de la funcionalidad del servicio: la transformación de la información, representada por las entradas y salidas, y por otro lado el cambio de estado producido por la ejecución del servicio, es decir, que precondiciones son necesarias para su ejecución y que efectos se producen tras ella. OWL-S proporciona las clases necesarias para describir los parámetros, los efectos, las precondiciones y todos aquellos conceptos necesarios para modelar la descripción funcional del servicio. Para realizar estas descripciones funcionales se utiliza la jerarquía de clases definida para el modelado del comportamiento del servicio y que se corresponden con el ServiceModel explicado a continuación.

- **ServiceModel:** Después de haber seleccionado un servicio las tareas de invocación, composición e interoperación se realizan mediante el modelo del servicio. Existen tres tipos de procesos, que a su vez son subclases del la clase *Process*. El proceso define las maneras en las que el cliente puede interactuar con el servicio. OWL-S hace una diferenciación entre tres tipos de procesos: los procesos simples *SimpleProcess*, proceso no invocable utilizado como elemento de abstracción), los procesos atómicos (*AtomicProcess*) y los procesos compuestos (*CompositeProcess*). Un proceso atómico es una descripción de un servicio que espera un mensaje de entrada y produce un mensaje de salida. Un proceso compuesto es aquel que

mantiene un estado interno, de tal forma que cada mensaje que envía el cliente produce un cambio en ese estado. Un proceso compuesto necesita el envío de varios mensajes para producir el resultado deseado. Para cada proceso es posible especificar las precondiciones y los efectos. Un proceso puede tener una o más precondiciones que deben ser cumplidas para que el servicio produzca el efecto correcto. La ontología proporciona clases para representar parámetros, entradas y salidas, y las precondiciones y efectos del servicio. Los procesos compuestos se pueden descomponer en otros procesos que pueden ser simples o compuestos a su vez. En OWL-S su descomposición puede ser especificada usando estructuras como las secuencias (*Sequence*) o estructuras condicionales de tipo (*If-Then-Else*). Estas estructuras no representan cual es el comportamiento interno real del servicio, sino que explican el comportamiento que el servicio tiene tal como lo ve el usuario desde fuera. Además OWL-S incluye la posibilidad de describir el flujo de datos, pudiendo definir como la entrada de un subproceso esta relacionada con la entrada de otro, todo ello mediante las clases *InputBinding* y *OutputBinding*.

- **ServiceGrounding:** A través de esta rama de la ontología principal se puede conocer en detalle cómo acceder a un determinado servicio. El *ServiceProfile* y el *ServiceModel* son abstracciones de alto nivel de un servicio, solo el *ServiceGrounding* conoce las especificaciones reales del mismo a bajo nivel. Su función es determinar cómo las entradas y salidas que han sido especificadas en un nivel más alto son convertidas en mensajes concretos que transportan esas entradas y salidas en un formato específico. Los desarrolladores de OWL-S han elaborado esta correspondencia para WSDL y para SOAP. Cada mensaje WSDL es derivado de una especificación realizada en OWL-S a través del uso de transformaciones XSLT.

OWL-S presenta dos aspectos de la funcionalidad del servicio: por un lado la transformación de la información, representada por las entradas y salidas, y por otro lado el cambio de estado producido por la ejecución del servicio, es decir, que precondiciones son necesarias para su ejecución y que efectos se producen tras ella. OWL-S proporciona las clases necesarias para describir los parámetros, los efectos, las precondiciones y todos aquellos conceptos necesarios para modelar la descripción funcional del servicio.

A pesar de que OWL-S no define un lenguaje específico para representar las reglas de los efectos y precondiciones, recomienda y proporciona algunas facilidades para trabajar con el lenguaje Semantic Web Rule Language (SWRL) y ofrece mecanismos para representar dichas fórmulas en otros lenguajes.

Los Inputs, Outputs, Preconditionss y Effects se conocen como IOPEs. En la última versión de OWL-S, los efectos son definidos como parte de un Result. Utilizamos el término Result para referirnos al conjunto formado por un Output y un Effect. El esquema para describir los IOPEs está definido en el Process.

Los Inputs y Outputs especifican la transformación de datos producida por el proceso bajo unas condiciones. Describen la información que se requiere para la ejecución del proceso y la información que produce el servicio. En el caso de procesos atómicos la información se recibirá del cliente y en el caso de los compuestos, algunos Inputs se recibirán del cliente y otros serán los Outputs recibidos de procesos anteriores. El número de Inputs y Outputs en un proceso es indefinido, pudiendo ser incluso cero.

En OWL-S, las precondiciones y los efectos se representan por fórmulas lógicas. El número de precondiciones y efectos también es indefinido. Las Precondiciones describen las condiciones del estado del mundo que deben ser ciertas para poder ejecutar el servicio satisfactoriamente. Se tratan de subclases de Expression. En OWL-S las expresiones especifican el lenguaje en el que la expresión está descrita (SWRL, KIF, DRS) y la propia expresión está codificada como literal (string o XML) dependiendo del lenguaje para expresiones. Los Efectos describen las condiciones del estado del mundo que son verdaderas después de la ejecución del servicio. Están definidos como parte de un Result.

El resultado de la invocación de un servicio se puede ver como una salida y un efecto. El resultado de un servicio se puede condicionar mediante las propiedades inCondition, hasResultVar, withOutput, hasEffect. La propiedad inCondition especifica la condición bajo la que se produce ese resultado y no otro. Las propiedades withOutput y hasEffect establecen qué sucede cuando la condición se satisface. La propiedad hasResultVar declara variables usadas en inCondition. Estas

variables, llamadas ResultVars, son análogas a las variables Locals, y sirven para un propósito similar, permitiendo ligar las variables a las precondiciones y usarlas en las declaraciones de salida y efectos.

2.4.4.2 WSMO

WSMO [687]-[289] es otra ontología para descripción de Servicios Web semánticos, realizada tomando como base el framework para WS llamado WSMF [291]. Al contrario que OWL-S, el proyecto WSMO no sólo pretende crear la especificación, sino que, también crear la arquitectura y un conjunto de herramientas para soportar la especificación [829]. El entorno de ejecución de WSMO se denomina WSMX [220] y es la implementación de referencia de WSMO para la integración de aplicaciones empresariales. WSMO fue enviado al W3C como propuesta el 4 de abril del 2005.

WSMO define cuatro conceptos principales: ontologías, WS, metas (goals) y mediadores (mediators). Además define su propio lenguaje para definir ontologías denominado WSML [247] que está basado en F-Logic [426], [427]. Hay cinco variantes de WSML en base a la expresividad y el alcance del lenguaje: Core, DL, Flight, Rule, Full. WSMO está definido utilizando un lenguaje llamado MOF [344]. MOF define una arquitectura de meta-datos consistente en cuatro capas: la capa de información que contiene la información que se quiere describir, la capa de modelo que contiene la meta-información que describe la información de la capa anterior, la capa del meta-modelo que contiene la información que describe la estructura y semántica de la meta-información, y por último, la capa de meta-meta-modelo que describe la información y estructura de la capa del meta-modelo. La ontología de modelado WSMO está definida en la capa de meta-meta-modelado.

En WSMO cada elemento está identificado mediante una URI o un identificador anónimo. Los identificadores anónimos se utilizan para denotar elementos que existen pero no necesitan un identificador específico. Las ontologías se definen como clases de MOF que contienen una serie de propiedades. Estas propiedades pueden ser descriptivas como el autor, la descripción de la ontología, la fecha, su formato, su identificador, el lenguaje, etc. O atributos funcionales: ontologías importadas a las que hace referencia esta ontología, mediadores utilizados para que las ontologías importadas puedan ser utilizadas entre sí, relaciones, funciones, instancias y axiomas.

Un concepto es el elemento básico de una ontología, las propiedades de un concepto son sus atributos, que definen campos que pueden ser rellenados con valores cuando se realice una instancia del concepto. Además, todo concepto puede tener un super-concepto que actúa como padre en una relación de jerarquía. Las propiedades del concepto padre son heredadas por el concepto hijo. Todo concepto puede poseer una expresión lógica que sirve para definir la semántica del concepto formalmente.

Las relaciones se utilizan para modelar las interdependencias entre conceptos, concretamente entre las instancias de los mismos. Toda relación puede tener una super-relación, lo que significa que heredará las restricciones definidas en la misma. Una relación tiene un número de parámetros que intervienen en ella. Las restricciones de una relación son especificadas mediante expresiones lógicas. Las funciones de una ontología son un tipo especial de relación. Pueden ser evaluadas especificando valores para los parámetros de la misma.

En WSMO un servicio queda definido, entre otras propiedades, por su capacidad, que define el servicio de acuerdo a su funcionalidad. La capacidad de un servicio queda definida por sus precondiciones, postcondiciones y efectos. Además un servicio puede tener interfaces. Una interfaz define como la funcionalidad del servicio puede ser conseguida. Los objetivos, por su parte, son descripciones de los problemas que deben ser resueltos por los servicios. Un objetivo queda descrito por las capacidades de los servicios que el usuario querría utilizar y por la interfaz que el usuario querría que tuviera el servicio con el que interactuar. WSMO también proporciona una serie de propiedades para definir a los mediadores.

El mecanismo para describir coreografías y orquestación en WSMO está basado en la metodología de ASM [688], [732]. El modelo implementado sigue estos principios: está basado en estados, representa los estados mediante álgebra y modeliza los cambios de estado mediante reglas de transición almacenadas que cambian los valores de las funciones y las relaciones definidas en el álgebra.

En WSMO un Servicio Web (WS) se define, entre otros parámetros, mediante las propiedades Interface y Capability. La propiedad Interface especifica detalles operacionales del WS mientras que Capability se refiere a sus capacidades funcionales.

En WSMO se especifica que cada Web Service sólo puede tener una Capability (multiplicity = single-valued). Una Capability puede tener Preconditions, Assumptions, Postconditions y Effects, que son objetos de tipo Axiom, expresiones lógicas compuestas de reglas escritas en WSML. Desde un punto de vista funcional, si se cumplen los axiomas definidos en Preconditions y Assumptions, se puede asumir que el servicio dispone de la Capability, y por tanto, se pueden ejecutar los axiomas que procedan de entre los definidos en Postconditions y Effects.

Las Preconditions describen los estados válidos del espacio de información previos a la ejecución del WS. El espacio de información se refiere a los Input de las operaciones para la ejecución del servicio en cuestión. Del mismo modo, las Postconditions describen los estados válidos de información que se garantiza tras la ejecución. Es decir, describen los Outputs obtenidos tras la ejecución del WS.

Por contra, los Effects describen los efectos que la ejecución del servicio provoca en su entorno; o dicho de otro modo, los cambios que ocurren en el ambiente. En cambio, las Assumptions describen los estados válidos y necesarios del entorno o ambiente para la ejecución del WS.

2.4.4.3 SWSO

WSF [108] es una propuesta de la iniciativa SWSI para el desarrollo de SWS. Esta iniciativa agrupa a investigadores del programa DAML y de proyectos europeos relativos a la Web Semántica. El trabajo del SWSI está estrechamente relacionado con OWL-S y WSMO, que son tomadas como referencia por los desarrolladores de SWSF. SWSF fue enviado al W3C como propuesta el 9 de mayo del 2005.

El modelo SWSF incluye 2 componentes principales:

- **SWSL** [118] es un lenguaje basado en lógica para especificar caracterizaciones formales de conceptos y descripciones de Servicios Web. Su principal objetivo es actuar como base para la ontología SWSO, el modelo conceptual de SWSI para los SWS. SWSL tiene dos variantes. La primera de ellas es SWSL-FOL, que tiene como objetivo la definición de la ontología del proceso y la otra es SWSL-Rules, un sublenguaje basado en reglas para soportar razonamiento sobre ontologías de lógica de orden uno. SWSL-FOL y SWSL-Rules son subconjuntos diferentes y separados de SWSL, siendo empleados para diferentes tareas, aunque el conocimiento expresado en los dos lenguajes puede ser combinado.

- **SWSO** [107] define el modelo conceptual para la definición de SWS en SWSL. La ontología SWSO, expresada en SWSL-FOL, también denominada FLOWS [119], tiene como objetivo la declaración de SWS y el razonamiento en base a éstos. Para conseguir este objetivo, incluye mecanismos de la familia de lenguajes PSL [346]-[725] para la descripción de procesos de negocio. SWSO puede ser visto como una extensión o refinamiento de OWL-S, pero aunque tenga muchas similitudes con las ontologías de OWL-S una gran diferencia es la expresividad del lenguaje subyacente.

La estructura de FLOWS es muy parecida a la propuesta por OWL-S, la cual, consiste en tres componentes principales:

- **Service Descriptors** provee información básica acerca del web service (nombre, autor, información de contacto, etc.) para que sea utilizado en el descubrimiento y emparejamiento de servicios en base a las preferencias del cliente.

- **Process Model** adapta la ontología genérica de los procesos PSL para proporcionar un *framework* para la descripción de WS. El Process Model es creado como un conjunto de seis modulos de ontologia del PSL-OuterCore. Este fundamento modular del modelo de proceso de SWSO hace que los desarrolladores de WS puedan definir y utilizar sus propias extensiones a FLOWS-Core. De hecho un objetivo principal de SWSO es facilitar la integración con otros lenguajes para el modelado de *workflows*, como puede ser BPEL4WS. SWSL permite traducir parcialmente SWSL-FOL a SWSL-Rules. Consiguiendo de esta manera expresar FLOWS mediante reglas. Éste es el objetivo de ROWS, una traducción de FLOWS a SWSL-Rules, que permite describir el *Process Model* mediante programación lógica.

- **Grounding** permite declarar detalles de bajo-nivel de las definiciones de los Servicios Web, como pueden ser: formato de los mensajes, protocolos de transporte, direccionamiento de red, etc. y tiene como objetivo el mapeo de descripciones de alto nivel de SWSO con WSDL.

FLOWS-Core incluye las primitivas de modelado de servicios básicas: servicio, proceso atómico e IOPEs, composición de actividades, mensaje y canal. Un bloque fundamental del modelo de proceso para Web Services de FLOWS es el concepto de proceso atómico. Un proceso atómico es una actividad PSL que generalmente es una subactividad de la actividad asociada con un servicio. Un proceso atómico es directamente invocable, no tiene subprocesos y se ejecuta en un sólo paso. Dicha actividad puede ser empleada para describir un Web Service simple o bien puede ser parte de una composición más compleja junto con otros procesos atómicos o compuestos.

Asociados con cada proceso atómico hay múltiples parámetros de entrada, salida, precondiciones y efectos (condicionales). Las entradas y las salidas son las entradas y salidas del programa que realiza el proceso atómico. Las precondiciones son aquellas condiciones que tienen que ser ciertas en el mundo para que el proceso atómico pueda ser ejecutado. En la especificación se dice que en muchos casos no habrá parámetros de precondición ya que la mayoría del software no tiene precondiciones físicas de ejecución, al menos al nivel en el que se pretende modelizar SWSO. Los efectos del proceso atómico representan las condiciones del mundo que son ciertas tras la ejecución del proceso atómico. Las precondiciones y efectos de un proceso atómico son expresados mediante una fórmula lógica de primer orden.

Además de las IOPEs de los procesos atómicos existen, las IOPEs asociadas a los servicios que están asociadas al comportamiento complejo del servicio. Pero éstas no son formalmente definidas ya que las entradas, salidas, precondiciones y efectos solamente pueden ser especificadas en los procesos atómicos. Sin embargo, para ciertas tareas automáticas, como el descubrimiento de servicios y la composición es necesario considerar las entradas, salidas, precondiciones y efectos del comportamiento complejo que describe el modelo de proceso completo del servicio. Y para ello, en algunos casos, estas propiedades pueden ser inferidas de los IOPEs de los procesos atómicos que constituyen el modelo de proceso completo.

2.4.4.4 WSDL-S

WSDL-S [48] es un pequeño conjunto de extensiones propuestas para WSDL, que tiene como objetivo asociar anotaciones semánticas a elementos WSDL como interfaces y tipos de esquemas de operaciones: precondiciones, efectos, salidas y entradas. Para ello aumentan la expresividad de las descripciones WSDL asociando ciertos elementos clave con conceptos expresados en ontologías. Estos Servicios Web Semánticos pueden ser además publicados en un registro UDDI y para ser descubiertos dinámicamente utilizando conceptos expresados en ontologías. Una de las grandes ventajas de WSDL-S respecto al resto de propuestas es que está basada estándares de Servicios Web ya existentes, cosa que el resto no lo hace.

En la especificación de la propuesta se resalta que se ha aumentado la expresividad de WSDL con semántica utilizando conceptos análogos a los que utiliza OWL-S, siendo agnósticos al lenguaje de representación semántico empleado en el mismo. El modo en el que WSDL-S permite especificar la correspondencia entre elementos WSDL y conceptos semánticos es muy similar a como lo hace el ServiceGrounding.

Lo que pretende esta propuesta es externalizar los modelos de dominio semánticos, es por ello que WSDL-S no está atado a ningún lenguaje de representación de ontologías, facilitando de esta manera la reutilización de modelos de dominio existentes, pudiendo elegir el lenguaje que más nos interese y permitiendo la anotación utilizando múltiples ontologías (del mismo o diferente dominio). En la propuesta ser resalta que es posible emplear WSDL-S enriquecido con anotaciones semánticas de OWL y WSMO.

Kopecký y otros [445] plantean una propuesta para extender WSDL-S para que cubra todas las características de WSMO, pero sin comprometer el objetivo principal de WSDL-S: la simplicidad y la independencia y resaltan que no tienen constancia de que se esté realizando trabajo alguno en lo referente a la integración entre WSDL-S y OWL-S [445].

WSDL-S está asociada con un entorno de ejecución y con un conjunto de herramientas denominado METEOR-S [651] llevado a cabo por el grupo LSDIS de la Universidad de Georgia. WSDL-S fue enviado al W3C como propuesta el 1 de octubre del 2005.

El proyecto METEOR-S define un amplio framework semántico que cubre todas las fases del ciclo de vida de los procesos Web [741]. Y muestra de ello son los resultados obtenidos por esta propuesta como pueden ser en la agregación de QoS [164], anotación semántica de Servicios Web [628], descubrimiento de Servicios Web [831], emparejamiento semántico [829], [605] y composición [514], [751], [752]. WSDL-S ha sido tomado como entrada por el grupo de trabajo del W3C para la nueva recomendación SAWSDL que pretende añadir semántica a las descripciones WSDL.

La especificación de WSDL-S enviada al W3C define que una precondición es el conjunto de statements (o expresiones representadas empleando conceptos de un modelo semántico) que se tienen que cumplir para que la operación sea satisfactoriamente invocada. Un efecto es el conjunto de statements (o expresiones representadas empleando conceptos de un modelo semántico) que son necesarios que se cumplan después de que una operación haya finalizado su ejecución tras ser invocada. Los efectos pueden ser diferentes en caso de que la operación se haya completado satisfactoriamente o no.

El elemento PreCondition se define en base al atributo Name en conjunto con ModelReference o Expression (el atributo ModelReference y el atributo Expression son mutuamente exclusivos). El atributo Name específica un identificador único para representar a una precondición dentro del conjunto de precondiciones en el documento WSDL. El atributo ModelReference específica la URI de la parte del modelo semántico que describe la precondición. El atributo Expression representa a la expresión que define la precondición. El formato de la expresión viene definido por el lenguaje semántico de representación empleado para expresar el modelo semántico. El elemento Effect se define de la misma manera que el PreCondition, pero además, un Effect puede estar asociado a un Fault, para ello basta con añadir al elemento Fault de la operación el elemento Effect correspondiente.

Las precondiciones y efectos son especificados como elementos hijo de la operación Element. Pese a que los esquemas especifican que solamente se puede tener una precondición y tantos efectos posibles como se quieran, en la especificación de dice que cada operación puede tener como máximo una precondición y un efecto, para así mantener la especificación simple. La especificación detalla que las precondiciones y los efectos complejos y condicionales deberían ser expresados en el modelo dominio/semántico. Por ejemplo, un conjunto de precondiciones (efectos) debería ser definido mediante expresiones lógicas que faciliten operaciones del tipo ánd', ór', 'xor', etc. al evaluar las expresiones. En la especificación, se presupone que el lenguaje empleado para la representación semántica soporta la representación de múltiples precondiciones en una única precondición de alto nivel. Ésta puede ser referenciada en el elemento Operation dentro de la especificación WSDL.

Respecto a lenguajes de representación de efectos y precondiciones, en la especificación no se establece cuál hay que emplear sino que hacen referencia a varias propuestas, tales como, SWRL (comunicad de la Web Semántica) y OCL (comunidad de modelado).

2.4.4.5 SAWSDL

SAWSDL [471] tiene como objetivo añadir semántica a las descripciones de los Servicios Web. Ya que la especificación de WSDL 2.0 [203] no incluye semántica en la descripción, por ejemplo dos servicios pueden tener una descripción similar pero tener significados completamente diferentes. Para cubrir dicha carencia SAWSDL define cómo añadir anotaciones semánticas a varias partes de documentos WSDL, como pueden ser estructuras de mensajes de entrada y salida, interfaces y operaciones. Las extensiones definidas en la especificación encajan en el marco de extensibilidad de WSDL 2.0. SAWSDL tiene dos tipos básicos de anotación denominados *ModelReference* y *SchemaMapping*:

- La anotación ModelReference, al igual que el ModelReference de WSDL-S, es utilizado para asociar tipos de interfaces/puertos, operaciones, salidas, entradas y elementos y atributos de XML schema con conceptos semánticos.

- La anotación SchemaMapping es empleada para transformar la representación de un dato en otro, como también lo hace WSDL-S.

En la recomendación del W3C no se detalla en ningún lugar que puedan representarse efectos y precondiciones en los servicios SAWSDL, pero en el Working Group Note del 28 de Agosto del

2007 [49] se describe cómo se pueden representar restricciones relativas al comportamiento de los servicios, es decir algo similar a los efectos y las precondiciones.

Los efectos y precondiciones son descritos dentro de cada operación, y más concretamente, referenciando las reglas que representan las precondiciones en el Input y las que representan los efectos en el Output. En los ejemplos que ofrecen las reglas están descritas en SWRL y WSML.

2.4.4.6 AmIGO-S

AmIGO-S [650] es una propuesta que extiende OWL-S integrando características relativas a la heterogeneidad y riqueza de los entornos ubicuos, añadiendo nuevas clases para definir propiedades funcionales (especifican cómo es posible el descubrimiento del servicio descubrimiento, comunicación y protocolos de red) y no funcionales (permitiendo modelizar el contexto del entorno ubicuo y también los atributos de calidad de servicio - QoS) para los servicios Amigo y permitiendo la definición de varios Groundings para un servicio.

Un servicio AmIGO se describe utilizando la clase Profile de OWL-S, que se utiliza para definir las propiedades globales del servicio. Una funcionalidad específica que ofrece un servicio Amigo es la Capability. El servicio se define utilizando tantas clases Capability como sean necesarias. Las propiedades funcionales de una Capability de un servicio están descritas por sus precondiciones, efectos y sus entradas y salidas. Las propiedades no funcionales de un servicio están descritas a nivel global o para cada Capability del servicio mediante el contexto del servicio y los parámetros QoS del servicio. En AmIGO-S, un servicio es descrito por uno o más profiles (clase Profile) de OWL-S. Define propiedades (algunas nuevas y otras de OWL-S) globales del servicio, comunes para todas las capabilities proporcionadas del servicio. Se reutilizan todas las propiedades definidas en el Profile de OWL-S, excepto la descripción funcional, que será fijada a nivel de Capability. Las propiedades funcionales de un servicio presentadas en el Profile se especifican con las capabilities del servicio, las conversaciones del servicio, y el Middleware subyacente.

En lugar de especificar la funcionalidad proporcionada por el servicio a nivel de Profile, como en OWL-S, en AmIGO-S se asocia un Profile con una o más Capabilities. Las ProvidedCapabilities permiten describir las Capabilities que son ofrecidas por el servicio y las RequiredCapabilities son las Capabilities que tendrían que ser ofrecidas por servicios externos. Si la RequiredCapability no está disponible, el servicio no pude ofrecer las ProvidedCapabilities. Por lo tanto, se pueden describir varias funcionalidades ofrecidas y requeridas por un dispositivo Amigo. Las propiedades comunes de todas las funcionalidades se especificarán en el Profile, mientras que las especificaciones de cada funcionalidad se describirán en la definición de Capability. La descripción funcional del Profile de OWL-S es inútil ya que las funcionalidades se especifican en la Capability.

Una conversación (descrita utilizando Atomic o Composite Process) ofrece la forma de interactuar con el servicio en términos de una secuencia de intercambio de mensajes. En OWL-S, cada Profile puede tener una descripción de la conversación descrita en el Model asociado con el Profile mediante la propiedad hasProcess. Un servicio Amigo proporciona (o requiere) varias capabilities que cada una describe una funcionalidad diferente. Por lo tanto, se introduce la propiedad hasConversation para definir conversaciones asociadas a cada Capability. La propiedad hasConversation es una propiedad funcional que declara que cada Capability tiene como mucho una conversación. La conversación global soportada por el servicio Amigo será la unión de todas las conversaciones de todas las capabilities proporcionadas. Los procesos pueden ser reutilizados en varias conversaciones que implementan interacciones similares.

La funcionalidad del servicio es ofrecida por Capability, utilizando la clase Capability del lenguaje AmIGO-S, relacionada con la clase Profile de OWL-S mediante la propiedad hasCapability. Al igual que la descripción funcional de la clase Profile de OWL-S, la descripción de cada Capability está dada por los Inputs y Outputs del servicio, las Preconditions que deben cumplirse para la ejecución del servicio y los Effects (Results) producidos en el mundo por la ejecución del servicio. Los inputs y Outputs corresponden a los mensajes que serán enviados y recibidos desde y hacia el servicio y se expresan en cualquier tipo de sistema con XML literales. Los Effects y las Preconditions se dan en una fórmula lógica, según lo estipulado por la clase Expresión de OWL-S como las expresiones DPR, KIF o SWRL. Por lo tanto la estructura, propiedades y forma de definir es la misma.

2.4.4.7 Análisis de los enfoques

En las siguientes líneas son descritas las conclusiones que se han obtenido tras el análisis de los principales modelos para representación de servicios semánticos (ver Tabla 2.21):

	OWL-S	WSMO	SWSO	WSDL-S	SAWSDL	AMIGO-S
Input	Input	PreCondition	Input	Input	Input	Input
Output	Output	PostCondition	Output	Output	Output	Output
PreCondtion	PreCondition	Assumption	PreCondition	PreCondition	Input Condition	PreCondition
Effect	Result	Effect	Effect	Effect	Output Condition	Effect
Service Implementation Technology	Web Services. Extensible by means of Grounding	Web Services	Web Services. Extensible by means of Grounding	Web Services	Web Services	Web Services. Extensible by means of Grounding
Language for ontology definition	OWL DL	WSML	SWSL-FOL	Agnostic	Agnostic	Same as OWL
Framework for service definition and development	Protégé	WSMO Studio	Not specified	Semantic Tools for Web Services, Radian, WSMO Studio	Radiant, WSMO Studio	Not specified
Maximum number of Preconditions	Multiple	Multiple	Multiple	One for operation (recommended), it can be a complex Boolean expression	Multiple	Same as OWL
Maximum number of effects	Multiple	Multiple	Multiple	One for operation (recommended)	Multiple	Same as OWL
Effect and Pre-Condition definition level in the service	Process	Capability	Atomic Process	Operation	Operation	Capability
Effect and Pre-Condition rules definition language	SWRL, KIF, DRS	SWML	SWSL-Rules	SWRL, OCL	SWRL,WSML	Same as OWL
Development level	High	High	Low	Low	Medium	Low
API implementation	OWL-S API	SWMO4J	Not specified	WSDL4J API	Woden4SAWSDL, SWMO4J, SAWSDL4J	Not specified

Tabla 2.21.: Lenguajes de representación semántica de servicios vs Efectos y precondiciones.

- Pese a que cada propuesta emplea diferente terminología a la hora de nombrar las entradas, salidas, efectos y precondiciones (IOPEs), todos tienen la misma base. Aun así, ninguno de los modelos define un modelo ontológico para la representación de efectos y precondiciones, y simplemente mencionan los lenguajes que pueden ser empleados para representarlos: SWRL, WSML y SWSL.
- Todos los lenguajes excepto WSDL-S permiten representar múltiples precondiciones por cada unidad invocable. Sin embargo, la única precondición de WSDL-S puede ser el resultado de un conjunto complejo de expresiones booleanas. A pesar de que todos los lenguajes permiten expresar múltiples efectos. WSDL-S recomienda definir un único Effect por Operation.
- WSDL-S y SAWSDL permiten añadir información semántica a descripciones WSDL, manteniendo así la estructura y simplicidad de WSDL. Estos dos enfoques son agnósticos del lenguaje empleado para representar las ontologías, siendo posible emplear OWL, WSML u otro modelo para anotar los conceptos del servicio.
- OWL-S, WSMO, SWSO y AmIGO-S son lenguajes más complejos y expresivos. Prueba de ello es que permiten definir procesos complejos, compuestos por operaciones simples, cosa que no es posible con WSDL-S y SAWSDL.
- En OWL-S y SWSO las IOPEs son definidas a nivel de Atomic Process. Algunas tareas automáticas como el descubrimiento y la composición requieren considerar las IOPEs del comportamiento complejo que describe el proceso de modelo compuesto del servicio. Como dicho comportamiento no puede ser definido de manera formal, son inferidos desde los Atomic Processes. Pese a que AmiGO-S emplea como base OWL-S, difieren en que las IOPEs son definidas a nivel de Capability.
- En OWL-S y SWSO un servicio solo puede tener una única unidad invocable (Atomic Process o Composite Process), por lo tanto las IOPEs de esta unidad pasan a ser las IOPEs del servicio. El resultado de esto es un conjunto de IOPEs por servicio. Lo mismo ocurre en WSMO, pero en este caso las IOPEs son definidas para la única Capability que puede tener el servicio. Por el contrario, en WSDL-S y SAWSDL las IOPEs son definidas a nivel de Operation y como un servicio puede tener múltiples Operation, no existen las IOPEs de servicio. Sin embargo, en AmIGO-S un servicio puede tener varias Capability y a su vez también varios Process, se puede decir que AmIGO-S trata de integrar la expresividad de OWL-S con la simplicidad de WSDL en lo referente a las Capabilities.

Definir cuál de los lenguajes es mejor resultad difícil, ya que todo depende del entorno, problemática, tipos de servicios y arquitectura que se pretende emplear. Pero a rasgos generales, se pueden destacar los siguientes aspectos de cada uno de ellos:

- OWL-S tiene la madurez y el amplio trabajo que se ha hecho en el hasta la actualidad.
- La ventaja de SWSO es su gran expresividad, pero por el contrario la comunidad científica que está trabajando en este enfoque es muy escasa.
- WSMO también ofrece un marco completo para describir servicios, pero está más orientado a otros enfoques más empresariales que relativos a la computación ubicua.
- AmiGO-S, siendo basado en OWL-S, tiene toda su madurez y además ha sido desarrollada con la clara intención de que sea principalmente empleada en dispositivos de entornos de computación ubicua. Pero por el contrario, la comunidad que está trabajando en ella es escasa.
- WSDL-S y SAWSDL destacan por su simplicidad, ya que no soportan la definición de procesos. Sin embargo, son enfoques que están más cerca de los servicios web sintácticos, pero añaden semántica a estos de manera muy simple. Estos enfoques, y en especial SAWSDL, principalmente debido a que ha sido propuesto por el W3C como enfoque para la descripción de Servicios Web Semánticos, hace que sea un candidato ideal como enfoque como enfoque de Servicios Web Semánticos sin proceso asociado.

Del análisis se concluye que los lenguajes para describir servicios semánticos no detallan ni describen cómo tienen que ser modelados los efectos y las precondiciones de los servicios y deja en manos de los diseñadores/desarrolladores cómo representar los efectos y las precondiciones y la relación que tienen estos con el entorno. Por lo tanto, consideramos necesaria la creación de un modelo ontológico para representar los efectos y las precondiciones de servicios que será utilizado dentro de ISMED.

2.4.5 Selección de servicios

Los mecanismos de emparejamiento son dependientes del lenguaje o formalismo en el que se definen los servicios. Es por ello que en los últimos años se han definido diferentes mecanismos para el emparejamiento que han ido evolucionando desde simples servicios descritos en base a palabras clave, pasando por servicios descritos en base a Input/Ouput hasta llegar a servicios que pueden ser descritos teniendo en cuenta condiciones asociadas a las Input/Output además del cambio en el estado del mundo previo y posterior a la ejecución del servicio. Estos diferentes métodos de representación de servicios están directamente relacionados con diferentes enfoques de descubrimiento de servicios. A continuación se describen los tres principales enfoques de emparejamiento de servicios que han sido empleados en los últimos años (ver Ilustración 4.17):

- **Keyword-Based:** Es el enfoque más básico de descubrimiento de servicios, basado en el emparejamiento de términos simples empleando técnicas de Information Retrieval (emparejamiento sintáctico). De esta manera el conjunto de palabras clave que representa a la solicitud del servicio es emparejado con los conjuntos de palabras clave contenidas en las descripciones de servicios. Estas técnicas basadas en palabras clave son limitadas debido a la ambigüedad del lenguaje natural y su carencia de semántica. Sin embargo, su gran ventaja es la escalabilidad ya que pueden ser empleados con grandes conjuntos de servicios y además pueden emplear tecnologías maduras para el emparejamiento de palabras clave.

- **Lightweight Semantic:** Este es un enfoque más avanzado que el anterior en el que los servicios son descritos empleando conceptos abstractos que los representan. Por lo tanto, los servicios son descritos como un conjunto de objetos y el emparejamiento entre ellos se determina comprobando si los conjuntos de objetos están interrelacionados, por ejemplo, empleando para ello un marco teórico que determine las relaciones entre los objetos del servicio solicitado y los publicados, como puede ser el propuesto por Paolucci y otros [779], [621].

- **Heavyweight Semantic:** Esta basada en descripciones semánticamente más ricas que el enfoque Lightweight tomando en cuenta la relación existente entre Input, Output, Efectos y Precondiciones. Esto permite que los estados de las propiedades puedan ser tomadas en consideración tanto antes como después de la ejecución del servicio. Empleando este enfoque será por lo tanto posible, determinar cuáles pueden ser los efectos que pueden derivar tras la invocación de un servicio, siendo estos dependientes de los valores de entrada del Input y del estado del entorno en ese instante.

Basado en los formalismos descritos por los servicios semánticos, se han llevado a cabo diversos enfoques que tienen como objetivo el emparejamiento entre servicios con el objetivo de comparar la conveniencia entre el servicio requerido y el servicio publicado. La efectividad del emparejamiento de servicios depende de la expresividad de la descripción del servicio y en la determinación adecuada del grado de emparejamiento entre las descripciones de los servicios. Klusch [432] describe que los actuales enfoques de emparejamiento semánticos de servicios pueden ser clasificados en base a dos parámetros:

- **Parte del modelo de servicios es empleada para realizar el matching.** La mayoría de enfoques emplean el emparejamiento basado en el perfil del servicio (*Service Profile*). Este tipo de emparejamiento, denominado, emparejamiento de servicios "black-box" determina la correspondencia semántica entre los servicios en base a la descripción de sus perfiles. El perfil del servicio describe qué es lo que el servicio hace en términos de su signatura

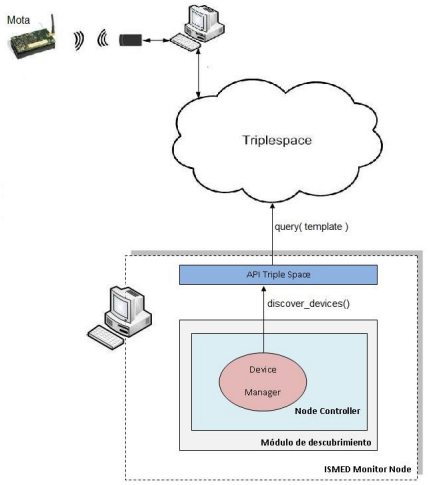


Figura 2.28.: Enfoques de descubrimiento sintácticos y semánticos, adaptado de [421]

(entradas, salidas, precondiciones y efectos) y aspectos no funcionales (categoría, nombre, QoS, privacidad, localización del servicio, etc.) En cambio el emparejamiento basado en el modelo de proceso (*Process Model*), denominado emparejamiento "glass-box", tiene como objetivo comparar el comportamiento operacional de los servicios en base al control de proceso y al flujo de datos de los mismos.

- **Método que se emplea para realizar el emparejamiento.** La mayoría de enfoques de emparejamiento de servicios realizan emparejamiento semántico de servicios basado en lógica, pero existen otra serie de enfoques denominados no-lógicos que no emplean ningún mecanismo de razonamiento basado en lógica para calcular el grado de emparejamiento entre descripciones, para ello se emplean técnicas de cálculo de similitud sintáctica, emparejamiento basado en grafos estructurados, o calculo numérico de distancia entre conceptos de una ontología. Además existe otro grupo que aglutina a los enfoques híbridos que mezclan tanto el razonamiento lógico como el no lógico para conseguir un grado de emparejamiento más exacto.

2.4.5.1 Emparejamiento basado en el modelo de servicio

En las siguientes líneas se ofrece una agrupación de los diferentes tipos de emparejamiento existentes (*Service Profile* y *Model Process*) en base al modelo de servicio empleado (ver Ilustración 2.29):

- **Emparejamiento basado en el modelo de perfil:** Aquí se engloban todos los enfoques relativos al emparejamiento basado en el modelo de perfil, es decir en las propiedades funcionales (entradas, salidas, efectos y precondiciones) y los no-funcionales (QoS, privacidad, etc.):
  - **Emparejamiento basado en propiedades no-funcionales:** En este tipo de emparejamiento se encontraría los enfoques que emplean las propiedades no-funcionales de los perfiles de servicios para el emparejamiento y selección de servicios. Entre las propiedades no-funcionales más empleadas destacan la calidad de servicios (QoS), tipo de interacción en el que el servicio puede ser empleado. Además en enfoques de emparejamiento para entornos de computación ubicua las propiedades no-funcionales que más se han empleado han sido las relativas al contexto y en muchas ocasiones relativas a la localización del servicio-usuario (ver apartado 2.4.5.2). Sin embargo, creemos que el contexto está estrechamente ligado con la funcionalidad ofrecida por el servicio, ya que depende de él para invocarse correctamente y los cambios que produce éste afectan directamente al entorno.

- **Emparejamiento basado en propiedades funcionales:** Aquí se engloban los enfoques que emplean tanto las salidas, entradas efectos y precondiciones (IOPE), clasificándolas en función de cuáles de ellas es empleada para el emparejamiento. El modelo de proceso de emparejamiento aceptado por la comunidad científica [621], [884]-[778] describe el emparejamiento como la suma de dos subprocesos: Emparejamiento= Emparejamiento de las Signaturas + Emparejamiento de las especificaciones. Donde se puede determinar que las signaturas serían el equivalente a las entradas y a las salidas y las especificaciones las equivalentes a los efectos y precondiciones. Todas estas propiedades formarían parte de las propiedades funcionales.
- **Emparejamiento Combinado:** Este tipo de emparejamiento surge de la combinación de las propiedades funcionales así como las no funcionales, pero ello no quiere decir que todos los enfoques empleen tanto las entradas, salidas, efectos, precondiciones y propiedades no-funcionales para el matching, ya que no existe en la actualidad enfoque destacable que emplee precondiciones y efectos combinado con atributos no-funcionales. Entre los enfoque que pueden encontrarse en este grupo destacan: GSD-MM de Chakraborty [162] un mecanismo de emparejamiento basado en lógica orientado a entornos de computación ubicua y distribuidos, iMatcher1 [123] y iMatcher2 [424] de Bernstein basados en OWL-S siendo el primero no-lógico y el segundo híbrido, FC-Match [129] de Bianchini realiza un emparejamiento híbrido basado en lógica y similitud textual (empleando WordNet y el coeficiente Dice) basado en perfiles OWL-S redefinidos como expresiones OWL-DL.

- **Emparejamiento basado en el modelo de proceso:** El emparejamiento de servicio semántico basado en el modelo de proceso, es en general, poco común. Y no es tomado en cuenta por lo actuales desarrolladores de lenguajes para describir servicios web semánticos. Además, los modelos de proceso de OWL-S y WSML no están actualmente definidos de manera formal y en el caso de SAWSDL y las descripciones de servicio monolíticas no ofrecen ni siquiera la posibilidad de definir el modelo de proceso. Sin embargo, este problema puede ser solventado parcialmente reescribiendo la descripción del modelo de proceso de manera que pueda ser descrito en un lenguaje lógico que tanga un sistema automático de verificación que disponga de herramientas para el correcto análisis del proceso [432]. Como este método de emparejamiento queda fuera de nuestro enfoque y no lo consideramos de interés no vamos a profundizar más en ello.

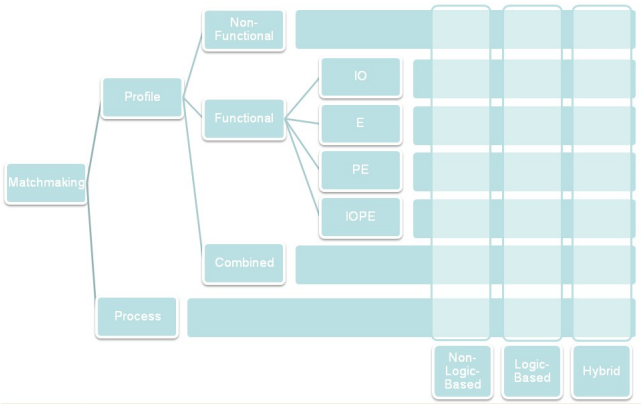


Figura 2.29.: Enfoques de emparejamiento semántico de servicios.

En las siguientes páginas se ofrece un análisis más detallado de los enfoques de emparejamiento basado en el perfil de servicio funcional, ya que es en estos enfoques en donde son empleados los efectos y condiciones. Pero además de lo anterior se mencionarán los diferentes métodos (lógico, no-lógico e híbrido) que se han empleado en cada uno de ellos.

2.4.5.1.1. Emparejamiento funcional basado en Entradas y Salidas. El tipo de emparejamiento basado en lógica trata la identificación de relaciones tipo subsumption entre conceptos que describen las operaciones que realizan los servicios. Esta relación permite relacionar conceptos con conceptos más genéricos en una ontología basándose en su definición formal descrita en lógica descriptiva. Tras el razonamiento tipo subsumption en la ontología, la estructura jerárquica resultante permite calcular los tipos de relación entre el concepto solicitado y los conceptos ofrecidos. A continuación se describirán varios enfoques que emplean lógica para el emparejamiento, seguido de varios que emplean un método no-lógico para el emparejamiento para acabar con un enfoque híbrido.

Varios esfuerzos han sido llevados a cabo en el área de emparejamiento (matching) de Servicios Web Semánticos basándose en las entradas/salidas de los servicios. Un algoritmo base para el emparejamiento basado en signaturas es el propuesto por Paolucci y otros [779], [621]. Este algoritmo permite el emparejamiento de una funcionalidad solicitada, siendo esta descrita como un conjunto de entradas ofrecidas y salidas requeridas, contra un número de funcionalidades expuestas, siendo descrita cada una como un conjunto de entradas requeridas y salidas ofrecidas. Las entradas y las salidas son semánticamente definidas empleando conceptos de ontologías. El algoritmo describe cuatro niveles de emparejamiento entre los conceptos ontológicos requeridos y los ofrecidos. Los cuatro niveles son los siguientes:

- **Exact:** Si la salida de la petición es igual a la salida del servicio registrado entonces se da este tipo de correspondencia. También existe una correspondencia exacta si el concepto de la petición es subclase del concepto del servicio registrado, pero sólo en el caso especial de que se pueda asegurar que la salida del servicio registrado proporciona todos los subtipos posibles.
- **Plug In:** Se da si el concepto de servicio registrado subsume al concepto de la petición del usuario. En este caso del concepto del servicio registrado puede usarse en el lugar del concepto de la petición.
- **Subsumes:** Si es la petición del usuario la que subsume el concepto del servicio registrado significa que el servicio no satisface completamente las necesidades del usuario. Debido a que un servicio de estas características proporciona resultados incompletos, el usuario debería realizar más operaciones para obtener los resultados deseados.
- **Fail:** Si no se da ningún caso de los anteriores el emparejamiento se considera un fallo. Es decir, la petición del usuario y la descripción del servicio registrado son incompatibles.

El grado de correspondencia total, entre las salidas de una petición de un usuario y las de la descripción semántica del servicio, se calcula como el peor de todos los resultados obtenidos al realizar las comparaciones individuales. En el caso de comparar las entradas del servicio, se realizará de la forma descrita, pero invirtiendo el orden de la comparación, es decir, se en ese caso se comparará la entrada de la notificación del servicio con la entrada de la petición del usuario. Posteriormente, el módulo de emparejamiento ordena los resultados obtenidos dando prioridad a aquellos servicios que tienen un grado de correspondencia mayor para las salidas, y solo si existe igualdad entre dos servicios utiliza el grado de correspondencia de las entradas.

Una modificación del algoritmo anteriormente descrito es planteado en [484]. La definición de grados de emparejamiento semántico es extendida para incluir un nuevo grado denominado Intersección. Por lo tanto, los grados existentes, ordenados de mayor a menor correspondencia semántica, quedan de la siguiente forma: *Exact, Plug In, Subsume, Intersection y Disjoint o Fail.* Mientras que en el algoritmo de Paolucci las comparaciones de los perfiles de servicio se realizan mediante la comparación semántica de cada uno de las propiedades descriptivas, en este artículo se propone la necesidad de permitir el cálculo de la correspondencia considerando el perfil de servicio como un solo concepto. Además de este algoritmo, existen otros enfoques que también se han basado en el método de emparejamiento descrito anteriormente [297], [794]. En el caso de [514] modifican el algoritmo en dos sentidos: por una parte, no es necesario emparejar todos los *Input/Output* del servicio solicitado con todos los *Input/Ouput* de los servicios disponibles. Solamente requieren que todos los *Input/Output* del servicio requerido sean emparejados con algunos o todos los *Input/Output* de los servicios disponibles. Esto hace posible que servicios que de otra manera serian

descartados puedan ser emparejados. Y por otra solamente calculan los matching tipo *Exact* y *Plug In* ya que creen que con Subsumes no se puede asegurar que el servicio ofrecido que hace este tipo de matching pueda ofrecer las funcionales requeridas, siendo así el algoritmo más rápido y eficiente.

Mandell y McIlraith [517], [518] emplean el sistema de emparejamiento SDS (Semantic Discovery Service) para la selección de servicios para su posterior uso en una composición creada mediante BPEL4WS, esta composición se realiza encadenando hacia atrás los servicios con el fin de obtener una composición con las entradas y salidas deseadas, empleando para ello un enfoque lógico.

Jaeger y otros [395] proponen un sistema de emparejamiento a 4 fases, en donde las tres primeras fases (entradas, salidas y categoría) emplean razonamiento de tipo subsumption empleando ontologías en OWL y la cuarta fase estaría orientada a un sistema que pueda emparejar requisitos que no son soportados mediante razonamiento basado en ontologías. Los autores se centran en describir las tres primeras fases detallando 4 tipos de emparejamiento: *Fail, Unknown, Subsumes y Equivalent.*

Constantinescu y otros [220] emplean en el sistema de emparejamiento HotBlu los grados de emparejamiento definidos por Paolucci y otros, pero además añaden uno nuevo denominado Overlap, que representa al Intersection del enfoque de Li y Horrocks [484]. Además para mejorar la eficiencia del sistema implementan una clasificación jerárquica de servicios codificados mediante intervalos, reduciendo de esta manera el problema del emparejamiento a la búsqueda de intersecciones entre estructuras rectangulares en hiperespacios. Gracias el proceso de indexación y codificación el proceso de emparejamiento pasa a ser mucho más rápido pero en cambio el proceso de inserción de servicios en los árboles con más de 10000 servicios pasa a ser un proceso muy pesado, necesitando hasta 3 segundos para publicar un nuevo servicio. Ben Mokhtar y otros [566], [560] en el sistema EASY también emplean un enfoque similar que en vez de emplear intervalos emplean codificaciones en base a números primos, por lo que en proceso de emparejamiento todavía resulta más rápido. Estos dos enfoques podrían ser considerados como enfoques de emparejamiento no-lógico ya que emplean codificaciones de intervalos y números primos para realizar el emparejamiento.

Otro enfoque de emparejamiento no basado en lógica es la infraestructura METEOR-SWSDI Discovery [831] y Lumina, el componente de búsqueda basado en UDDI. En Lumina la búsqueda es basada en palabras clave, mientras que en MWSDI el descubrimiento de servicios SAWSDL se basa en emparejamiento no-lógico empleando ello dos técnicas para el cálculo de la distancia semántica: por una parte algoritmo de emparejamiento estructural y por otra el algoritmo de emparejamiento a nivel de elemento. El algoritmo de emparejamiento de nivel de elemento calcula la similitud lingüística entre los conceptos, mientras que el algoritmo de emparejamiento estructural considera la similitud entre los sub-arboles de los conceptos para calcular la similitud. Además de Lumina existen otros dos enfoques principales que emplean descripciones de servicios basadas en SAWSDL, la primmera de ellas es SAWSDL-MX de Klusch y otros [435], el cual emplea un mecanismo híbrido de emparejamiento al igual que lo hacen sus antecesores OWLS-MX [434], [433] y WSMO-MX [413], [414]. Y la segunda es FUSION [447], de Kourtesis y Paraskakis, el cual es un sistema de emparejamiento lógico que emplea registros UDDI. En FUSION, cualquier descripción de servicio es clasificado en el momento de su publicación y después es mapeado a UDDI para así facilitar búsquedas rápidas.

OWLS-MX [434] es hasta la actualidad el único sistema de emparejamiento híbrido para OWL-S. OWLS-MX complementa el razonamiento basado en lógica con varias métricas de similitud de IR (*Information Retrieval*) basados en Token para calcular el emparejamiento semántico entre pares de servicios. Los autores [433] además argumentan que un emparejamiento solamente basado en razonamiento de Logica Descriptiva (como son la mayoría de enfoques actuales) es insuficiente en práctica ya que un enfoque híbrido. Argumentan que la calidad del proceso de descubrimiento semántico puede mejorar considerablemente combinando de manera adecuada tanto métodos de lógica descriptiva como técnicas tipo *Information Retrieval*. En un artículo posterior [433] realizan un análisis de la calidad de los resultados obtenidos, analizando los posibles falsos positivos y falsos negativos con el objetivo de analizar los beneficios y puntos flacos que tienen tanto el emparejamiento basado en lógica como el híbrido. De este artículo se extrae que los falsos positivos/negativos pueden ser mitigados mediante medidas de solapamiento sintáctico de las métricas IR y mediante un emparejamiento híbrido integrado en conceptos de bajo nivel. Los autores resaltan que el sistema es mejor que otros enfoques puramente lógicos, pero como comentan Küster y

otros en [456] no está claramente demostrado que sea mejor ya que el conjunto de servicios que se empleo para el test era completamente desarrollado por los autores y además resaltan lo siguiente: "OWLS-TC2 is suited to test and evaluate the features of this particular hybrid matchmaker...". Por lo tanto, se puede considerar que la validación es parcial y no completa.

2.4.5.1.2. Emparejamiento funcional basado en Efectos. El emparejamiento de descripciones de servicios basados en Lógica Descriptiva monolítica es llevado a cabo exclusivamente dentro de lo que se considera la teoría clásica del razonamiento en lógica descriptiva. Este tipo de emparejamiento de efectos basado en lógica es simple pero agnóstico para las estructuras impuestas por formatos de servicios web semánticos como OWL-S y WSMO. Klusch manifiesta en [414] que la investigación en este campo no ha hecho más que comenzar y que en parte esta investigación está relacionada con la investigación en razonamiento no monotónico que se emplea en lenguajes basados en regla de la Web Semántica. Como este método de emparejamiento queda fuera de nuestro enfoque y no lo consideramos de interés no vamos a profundizar en ello.

2.4.5.1.3. Emparejamiento funcional basado en Efectos y Precondiciones. El emparejamiento de especificaciones se basa en el emparejamiento de precondiciones y postcondiciones que describen la semántica operacional de los servicios. El primer trabajo de referencia relativo al emparejamiento funcional basado en efectos y precondiciones es el planteado por Zaremski y Wing en [885], donde el emparejamiento basado en especificaciones es llevado a cabo empleando técnicas de theorem proving (describiendo las precondiciones y postcondiciones mediante predicados de lógica de primer orden), por ejemplo: infiriendo reglas generales de tipo sumbsuption entre expresiones lógicas que describen las precondiciones y postcondiciones de los componentes. El autor describe varios tipos de relaciones Exact Pre/Post, Plug-in, Plug-in Post, Weak Post, Exact Predicate, Generalized y Specialized, donde las cuatro primeras son emparejamientos tipo Pre/Post y las últimas tres son emparejamientos basados en el predicado. A continuación de ofrece una descripción más amplia de cada uno de los tipos de relaciones:

- **Pre/Post Matches:** Tipo de emparejamiento en donde se relacionan por una parte precondiciones y por otra las postcondiciones de cada componente.
- **Predicate Matches:** Tipo de emparejamiento en donde se relaciona toda la especificación de predicados de los dos componentes.

Lin y Arpinar proponen en [884], [490] un mecanismo para el definición de relaciones semánticas entre Web Services, empleando para ello las precondiciones y postcondiciones de los mismos. Los tipos de relaciones que plantean están orientadas para su empleo en procesos de composición de servicios y no tanto a emparejamiento. Los autores definen dos grupos de relaciones, por una parte las relaciones entre Web Services y por otra las relaciones entre condiciones.

En este enfoque sin embargo, los autores no definen como representar las condiciones, sino que solamente mencionan que van a emplear tripletas RDF y además el mecanismo que emplean para determinar las relaciones entre condiciones solamente emplean acepta servicios con una sola precondición y una sola postcondición. Además en este caso, de manera similar a como lo hacen en LARKS [777], los autores resaltan que las precondiciones representan condiciones asociadas a los Input y las postcondiciones representan las condiciones asociadas a los Output, sin embargo, las condiciones no solamente están relacionadas con los IO, sino que también están relacionadas con la información contextual.

Un sistema de emparejamiento más reciente es el PCEM [139], el cual está basado en lógica. Los servicios semánticos de PCEM están descritos mediante OWL-S, las anotaciones de éstos están descritos mediante ontologías en OWL y los efectos y precondiciones están descritos mediante PDDL. Sin embargo, internamente el mecanismo de emparejamiento emplea Prolog como mecanismo de representación de servicios, ontologías y efectos y precondiciones así como para el razonamiento. PCEM realiza dos tipos de emparejamiento:

- **Exacto:** Comprueba que las precondiciones del servicio solicitada emparejen de manera exacta con el publicado y lo mismo con los efectos. Para ello, comprueba si hay una posible sustitución de una variable vacía que cuando sea aplicada a una o a las dos proposiciones

(representando tanto las dos precondiciones como los dos efectos) de lugar a dos expresiones equivalentes.

- **Basado en razonamiento:** El emparejamiento basado en razonamiento, es más complejo que el emparejamiento exacto. Para ello se emplean las reglas generales de inferencia de Prolog (por ejemplo, todas las reglas de deducción), además de reglas de inferencia específicas de dominio (por ejemplo, subPartOf). Para las primeras se emplea el mecanismo de razonamiento de Prolog empleando resolución y CWA (Closed World Assumption) y las últimas pueden ser específicas para ciertas precondiciones y efectos.

2.4.5.1.4. Emparejamiento funcional basado en Entradas, Salidas, Efectos y Precondiciones. Un método para el emparejamiento de IOPEs es empleando el denominado query containment [777], [778], [750], para ello tanto los servicios publicados como las peticiones de los clientes son modelados mediante consultas con ciertas restricciones. Empezando con las restricciones definidas los posibles valores de ambas consultas, tanto la del servicio solicitado el servicio publicado, son evaluadas y la posible inclusión entre los resultados de las consultas es inferida. Una consulta q1 es contenida en q2 si todas las respuestas de q1 están incluidas en q2. El sistema LARKS de Sycara y otros [777], [778], donde se comprueban las relaciones polinomiales tipo theta-subsumption de las precondiciones y postcondiciones descritas como clausulas def-Horn.

WSMO-MX [413], [414] es el único sistema híbrido de emparejamiento escrito basado en entradas, salidas, efectos y precondiciones junto con LARKS. Este sistema emplea una extensión LP (Logic Programming) de WSML-Rules, denominado WSML-MX para la descripción tanto de los servicios como metas a conseguir. El sistema adopta los principales elementos requeridos para el emparejamiento de servicios en WSML, que son, goal, service, capabilities, preconditions and post-conditions, pero no effect y assumption. El sistema toma como base ideas del algoritmo hibrido de emparejamiento de OWLS-MX [434], [433], el emparejamiento basado en grafos orientados a objetos del sistema DSD-MM [431], [430] y el concepto de matching intencional de servicios propuesto por Keller [414]. Además de basarse también en general en LARKS [777], [778], pero permitiendo una parametrización más afinada combinando tanto emparejamiento sintáctico como semántico.

2.4.5.2 Emparejamiento de servicios semánticos en dispositivos con recursos limitados.

Los métodos más comunes de selección SDP, Salutation, SLP, SSDP y Zeroconf ofrecen métodos de emparejamiento (sintácticas) similares, pero ninguna de ellas semántica. Muchas de estas emplean patrones o palabras clave para el emparejamiento, pero este tipo de enfoque no es adecuado para entornos ubicuos, ya que uno de los mayores problemas a la hora de afrontar el problema de la riqueza y heterogeneidad de los entornos de computación ubicua es el problema de utilizar la información contextual para inferir cuales son las necesidades de los usuarios que se mueven en el entorno y localizar automáticamente los servicios más apropiados [840],[477].

Los protocolos de descubrimiento de servicios sensibles al contexto tienen como objetivo proveer a los usuarios los mejores servicios de red, basándose en sus preferencias, necesidades y condiciones de ejecución [478], [187]. En muchos de los casos se centran en la sensibilidad relativa a la localización (location awareness) [648], [892] mientras en otras consideran la información del contexto para personalizar así el proceso de selección de servicios [154], [888]. En otras en cambio emplean información relativa a la calidad de servicio (QoS) [162].

La evaluación de las propiedades del contexto debe ser llevado a cabo mediante la evaluación de reglas de contexto [670] o mediante la maximización de una función de utilidad [162], pero normalmente los enfoques solamente se centran en el emparejamiento sintáctico entre la información proveniente del cliente y la que proviene del servicio.

Se da el hecho que la mayoría de estos protocolos de descubrimiento de servicios sensibles al contexto asumen que el cliente y el proveedor de servicios emplean la misma terminología para describir la información contextual, pero tal hecho no es real en entornos de computación ubicua debido a que los diferentes componentes del entorno son diseñados, desarrollados y desplegados independientemente, es por ello que es necesario que se empleen lenguajes que permitan describir la semántica operacional de los servicios de manera adecuada con el objetivo de realizar un descubrimiento de servicios centrado en las necesidades de los usuarios. Diversos estudios han trabajado en el descubrimiento semántico y sensible al contexto de servicios en entornos ubicuos [176], [178],

[565], [560], pero todavía ninguno de ellos ha empleado los efectos y precondiciones para la selección de servicios, simplemente han empleado las entradas y las salidas de los mismos e información relativa al contexto del usuario y su localización.

En la actualidad destacan dos propuestas basadas en el emparejamiento basado en signaturas (Input y Output) desarrolladas por investigadores del proyecto ARLES de INRIA Rocquencourt de reciente creación que tienen como objetivo el emparejamiento eficiente de Servicios Web Semánticos orientado a dispositivos con recursos limitados.

2.4.5.2.1. EASY. EASY (Efficient SemAntic Service DiscoverY in Pervasive Computing Environments with QoS and Context Support) [567], [565] que provee de un marco para el descubrimiento semántico de servicios eficiente en entornos de computación ubicua, factor que es clave para dispositivos de recursos limitados que pueden encontrarse en los entornos de computación ubicua. Para ello codifica los conceptos de la ontología en modo offline (es decir, antes del proceso de descubrimiento) y clasifica adecuadamente las descripciones de servicios en el repositorio basado en los conceptos codificados. El algoritmo de emparejamiento de EASY adopta (con una ligera modificación) el enfoque relativo a los niveles de emparejamiento de Paolucci y otros. Además, comparado con otros enfoques relativos al emparejamiento de signaturas el enfoque EASY no solamente soporta aspectos funcionales sino que además soporta aspectos no-funcionales como son la calidad de servicio (QoS) y las propiedades del contexto. Además ofrece medios para clasificar los servicios en función de las preferencias de los usuarios basándose en varias propiedades no-funcionales. El algoritmo resultante clasifica los servicios evaluando los niveles de emparejamiento entre los conceptos empleados en la petición de servicio y aquellos que están siendo publicados.

En el artículo de EASY realizan un estudio del tiempo necesario por los razonadores para clasificar los servicios anotados semánticamente y se concluye que el proceso es muy lento, llegando incluso a necesitar cerca de 5 segundos para parsear, clasificar y realizar el emparejamiento de conceptos. Por lo que consideran que son tiempos muy elevados para ser aplicados en entornos de computación ubicua y con el objeto de mejorar el rendimiento del proceso plantean una serie de mejoras:

- Desplazar el proceso de clasificación de online a offline.
- Codificar los conceptos de las ontologías para así reducir el problema a una simple comparación de códigos.
- Organizar y agrupar las descripciones de servicios en estructuras para aligerar el proceso de emparejamiento.

2.4.5.2.2. PerSeSyn. Hasta la actualidad el emparejamiento sintáctico y el semántico han sido considerados como problemas separados, más allá, los enfoques de descubrimiento de servicios relativos a la interoperabilidad (que tiene como objetivo facilitar el descubrimiento de servicios de diferentes tipos de protocolos) se han centrado en el descubrimiento sintáctico, dejando de lado el semántico. Y al mismo tiempo los protocolos de descubrimiento de red semánticos no se han preocupado nunca de los aspectos relativos a la heterogeneidad. Y los protocolos de descubrimiento sensibles al contexto han dado por hecho que todos los clientes y proveedores emplean la misma ontología, cosa que en la realidad no será así, ya que cada dispositivo tendrá la ontología que ha desarrollado el fabricante del mismo. Con el objetivo de integrar los problemas anteriormente descritos se plantea la plataforma de descubrimiento de servicios PerSeSyn (Pervasive Semantic Syntactic) [567], [569], [569] que tiene las siguientes características:

- Plantean un modelo y lenguaje para la descripción de servicios que abstrae y permite representar las diferentes tipos de lenguajes para describir servicios (UPnP, SLP, Servicios Web, Servicios Web Semánticos). El descubrimiento de servicios es interoperable, ya que permite integrar diversos tipos de servicios y protocolos diferentes, como son: UPnP, SLP, Servicios Web, Servicios Web Semánticos.
- El emparejamiento de servicios puede ser semántico, sintáctico o mixto (pueden existir servicios que tengan ciertas partes anotadas semánticamente y otras que no) dependiendo del tipo de servicio solicitado y de los servicios publicados.

- El emparejamiento puede ir desde un simple servicio descrito sintácticamente (SLP) hasta servicios semánticos con conversaciones asociadas (OWL-S).
- Plantean un mecanismo para la clasificación de los resultados de emparejamiento heterogéneo de servicios.
- El emparejamiento no solo tiene en cuenta los atributos funcionales sino que también los atributos no funcionales como son la calidad de servicios (Qos) y el contexto.

El algoritmo de emparejamiento empleado extiende el algoritmo del enfoque EASY mejorándolo y añadiendo funcionalidades para el emparejamiento semántico y de conversaciones. Para el prototipo se emplea el lenguaje SAWSDL para describir los servicios y BPEL para describir las conversaciones de los servicios y como resultados cabe manifestar que se obtiene un rendimiento del emparejamiento semántico igual o incluso menor en muchos de los casos, todo ello gracias a la codificación de las ontologías y a la agrupación de los servicios similares.

2.4.6 Conclusiones

De las dos últimas propuestas para emparejamiento semántico de servicios en entornos de computación ubicua se nota claramente la necesidad de disponer de algoritmos rápidos y eficaces que permitan realizar un emparejamiento lo más ajustado posible con el objeto de reducir el consumo de memoria, procesador y batería de los dispositivos con recursos limitados que conformarán el ecosistema de ISMED. Y como comenta Ben Mokhtar en [565] tanto si el emparejamiento de servicios semánticos es realizado mediante la signatura o bien mediante la especificación el factor clave en ambos (en clave de rendimiento) subyace en el rendimiento del motor de razonamiento empleado.

En los dos casos planteados el razonamiento y clasificación de ontologías es off-line, es decir, no es realizado en el momento del descubrimiento sino que es realizado al cargar los servicios, por lo que no es realmente aplicable en un entorno de computación ubicua real y dinámico donde los servicios anunciados cambian a lo largo del tiempo y mucho menos en dispositivos con recursos limitados. Además el enfoque de emparejamiento planteado en ambos es el basado en signatura, el cual no ofrece una respuesta real a los servicios existentes en entornos ubicuos, sería más adecuado poder describir el comportamiento de los servicios en base a las precondiciones que son necesarios para que pueda ejecutarse y en base a los efectos que genera la ejecución del mismo. De esta manera lo que se consigue es que tras la ejecución del servicio, cambie el estado del mundo, y no solo se ejecuten servicios que requieren y ofrecen información, como es el caso del emparejamiento basado en signaturas.

Del análisis de los enfoques de emparejamiento actuales [referencia 2.4.5.1] se extraen las siguientes conclusiones:

- El lenguaje más empleado para describir servicios es OWL-S, seguido de WSMO y de SAWSDL. Pero además de éstos existen otros lenguajes específicos y también mecanismos de emparejamiento que se basan en descripciones monolíticas de servicios descritas como un concepto simple en Lógica Descriptiva.
- El tipo de emparejamiento más empleado en base a la descripción del servicio es el basado en entradas y salidas seguido del basado en entradas, salidas, efectos y precondiciones y de los menos desarrollados el enfoque basado exclusivamente en Precondiciones y Efectos.
- El tipo de emparejamiento más empleado en base a la técnica de emparejamiento es el basado en lógica, pero estos últimos años se está trabajando también en mecanismos híbridos para el emparejamiento, que están demostrando que un enfoque híbrido ofrece resultados más exactos que el puramente lógico.

Tal y como comenta Klusch [414], en la actualidad no hay sistemas de emparejamiento basados exclusivamente en precondiciones y efectos que empleen enfoques híbridos o no basados en lógica y ninguno aplicado a entornos de computación ubicua, por lo tanto, supone un reto implementar un mecanismo de descubrimiento/emparejamiento de servicios basados en efectos y precondiciones aplicado a dispositivos con recursos limitados ya que como comenta Ben Mokhtar [560] las

soluciones y propuestas actuales de emparejamiento basado en efectos y precondiciones emplean costosas técnicas de prueba de teoremas [885], o bien emplean sistemas de consulta en bases de conocimiento centralizadas [777], [778], [413], [414] no siendo estas propuestas apropiadas para emplearlas en entornos distribuidos con dispositivos con recursos limitados. Sin embargo, antes de definir unos mecanismos de emparejamiento basado en efectos para dispositivos con recursos limitados y orientado a entornos de computación ubicua es necesario definir un mecanismo para representar los efectos y condiciones de los servicios de entornos de computación ubicua.

2.5 Razonamiento

2.5.1 Análisis de OWL

2.5.1.1 Comparativa de OWL Lite, OWL DL y OWL Full

	OWL Lite	OWL DL	OWL Full
Advantages	- Its inference capabilities are enough to do hierarchical classification using simple constraints. - Reasoning over OWL Lite is efficient (computationally simple and fast).	- Computationally complete and decidable, can be computed in finite time. - It models Description Logic, a decidable fragment of the first-order logic.	- Maximum expressiveness
Disadvantages	- Limited expressiveness	- Some restrictions against OWL Full: One class can not be an individual or property and one property can not be an individual or a class. Also there is a separation between object properties and data type properties	- No computational guarantees (computation has not to be finite)

Tabla 2.22.: Ventajas y desventajas de los distintos tipos de OWL.

2.5.1.2 Comparación de vocabulario

La figura 2.30 explica la relación del vocabulario de OWL Lite, DL y Full. Tanto OWL DL como OWL Full tienen el mismo vocabulario mientras que OWL Lite es un subconjunto de dicho vocabulario.

2.5.1.3 Comparativa de OWL 1 y OWL 2

La siguiente información ha sido sacada de [336].

2.5.1.3.1. Limitaciones en la expresividad. En lo que referente a la expresividad de OWL 1 sus inconvenientes han sido ampliamente reconocidos por la comunidad de DL, y se ha realizado una cantidad significativa de investigaciones tratando de buscar posibles soluciones a los mismos. Los resultados de dichos trabajos se incorporaron a DL SROIQ, que es estrictamente más expresivo que SHOIN.

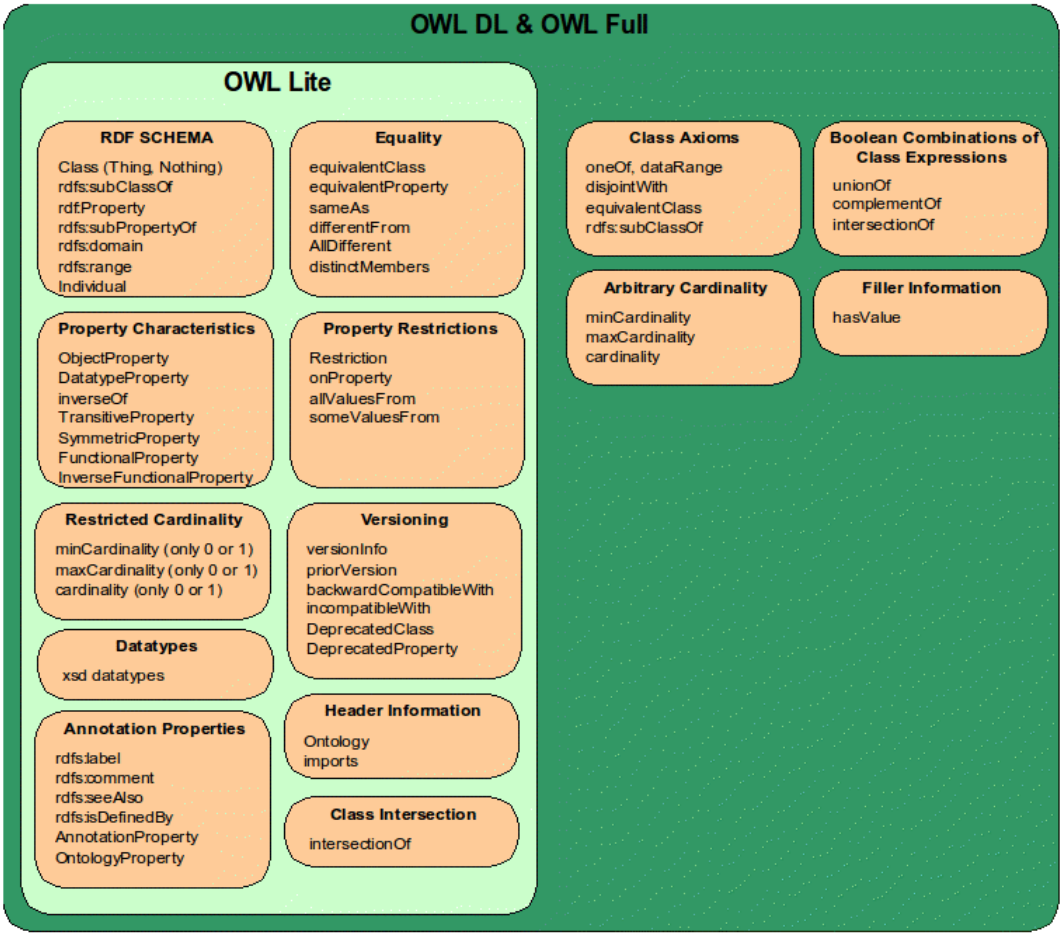


Figura 2.30.: Comparativa de los vocabularios.

Tabla 2.23.: Limitaciones en la expresividad de OWL.

Issue	Problem in OWL 1	Solution in OWL 2
Qualified Cardinality Restrictions	Cardinality restrictions cannot be qualified with classes other than owl:Thing (The following example cannot be modelled: "a medical oversight committee as a committee that consists of at least five members, of which two are medically qualified, one is a manager, and two are members of the public").	Even while OWL 1 was being designed, it was known that QCRs could have been added to the language without any theoretical or practical problems: the resulting logics are still decidable and have been successfully implemented in practical reasoning systems. Thus, qualified number restrictions were incorporated into OWL 2 in the obvious way.
Relational Expressivity: Transitive propagation of properties.	Applications commonly need to model interactions that are sometimes described as one property "propagating" or being "transitive across" another. E.g. "we might want to assert that an abnormality of a part of an anatomical structure constitutes an abnormality of the structure as a whole".	This is addressed in SROIQ and OWL 2 by the provision complex property inclusion axioms, which significantly increase the relational expressivity of the language. In particular, they provide for the propagation of one property along another. In addition to complex property inclusion axioms, properties in SROIQ and OWL 2 can be transitive, reflexive, irreflexive, symmetric, or asymmetric, and pairs of properties can be made disjoint.
Relational Expressivity: Properties of properties.	The partOf relation is often defined to be transitive (if x is a part of y and y is a part of z, then x is a part of z), reflexive (every object is a part of itself), and asymmetric (nothing is a part of one of its parts). It was not included in OWL 1 because its effects in the decidability of the ontologies where unknown.	
Continúa en la siguiente página		

Continuación de la tabla		
Issue	Problem in OWL 1	Solution in OWL 2
Datatype Expressivity	OWL 1 provides very limited expressive power for describing classes whose features have concrete values such as integers and strings. Is not possible to model assertions like: restrictions to a subset of datatype values (e.g., a gale as a wind whose speed is in the range from 34 to 40 knots). relationships between values of concrete features on one object (e.g., a square table is a table whose breadth equals its depth). relationships between values of concrete features on different objects (e.g., people who are older than their boss). aggregation functions (e.g., the duration of a process is the sum of the durations of its subprocesses). Another important limitation of the datatype support in OWL 1 is the lack of a suitable set of built-in datatypes. OWL 1 relies on XML Schema 11 for the list of built-in datatypes.	OWL 2 significantly extends the set of built-in datatypes of OWL 1 by reusing certain datatypes from XML Schema. Thus, OWL 2 now supports owl :boolean, owl :string, xsd:integer, xsd:dateTime, xsd:hexBinary, and a number of datatypes derived from these by placing various restrictions. In addition, xsd:decimal , xsd:double, and xsd:float will most likely be replaced with owl :real—a datatype interpreted as the set of all real numbers, providing a constant for all rational numbers. OWL 2 also provides a datatype restriction construct, which allows new datatypes to be defined by restricting the built-in datatypes in various ways.
Keys	OWL 1 does not provide means for expressing key constraints, which are a core feature of database technologies.	Extending DL-based languages such as OWL 2 with keys, however, poses both theoretical and practical problems. Therefore, the Working Group has decided to include a more restricted variant of keys that can be useful in practice as well as relatively easy to implement, commonly known as easy keys. Unlike the general keys, easy keys are not applied to individuals not known by name.
Issue	Problem in OWL 1	Solution in OWL 2

2.5.1.3.2. Aspectos sintácticos. Véase la tabla 2.24.

Tabla 2.24.: Aspectos sintácticos.

Issue	Problem in OWL 1	Solution in OWL 2
Frame-Based Paradigm	OWL 1 Abstract Syntax is heavily based in frames. The problem is that the frames are somewhat different to Description Logic axioms, been this a source of confusion for developers.	The structure of OWL 2 ontologies has been unambiguously specified using OMG'sMeta-Object Facility (MOF). The classes of the MOF metamodel describe the canonical structure of OWL 2 ontologies in a way that is independent of the syntax used to serialize the ontologies. The MOF metamodel for OWL 2 can be seen as analogous to the Document Object Model (DOM) specification 16 for XML. It unambiguously specifies what OWL 2 ontologies are in terms of their structure and thus makes the specification precise. In addition to the MOF metamodel, the OWL 2 specification defines a Functional-Style Syntax as a simple encoding of the metamodel. OWL 2 departs in its conceptual design and in syntax from the OWL 1 Abstract Syntax. The main difference with respect to the OWL 1 Abstract Syntax is that the Functional-Style Syntax of OWL 2 does not contain the frame-like syntactic constructs of OWL 1; instead, ontology entities are described at a more fine-grained level using axioms.
Alignment with DL Constructs	Even though OWL 1 is based on DLs, the constructs of OWL1 Abstract Syntax do not completely correspond to the constructs of DLs.	
Continúa en la siguiente página		

Continuación de la tabla		
Issue	Problem in OWL 1	Solution in OWL 2
Determining the Types of Ontology Entities	Neither Abstract Syntax nor OWL 1 RDF is fully context-fre, that is, an axiom containing a URI often does not contain sufficient information to disambiguate the type of ontology entity (i.e., a concept, a property, or an individual) that the URI refers to. To disambiguate the syntax, OWL 1 DL relies on a strict separation of the vocabulary into individuals, classes, and data and object properties. The specification of OWL 1 DL, however, does not precisely specify how to enforce this separation at the syntactic level. Thus, whereas the semantics of OWL 1 DL requires strict typing of all names, the syntax does not enforce it, which can prevent certain ontologies from being interpreted.	OWL 2 introduces the notion of declarations. All entities can, and sometimes even must, be declared in an OWL 2 ontology. OWL 2 imposes certain typing constraints on ontologies. Roughly speaking, the sets of object, data, and annotation properties, as well as the sets of classes and datatypes in an OWL 2 ontology must be mutually disjoint.
Problems with OWL 1 RDF	The vast majority of OWL 1 ontologies have been written in OWL1 RDF syntax; this syntax has, however, shown itself to be quite difficult to use in practice. The main difficulty is that RDF represents everything using triples, which specify relations hips between pairs of objects. In contrast, many OWL 1 constructs cannot be represented using triples without the introduction of new objects.	In order to provide an alternative to RDF, OWL 2 comes with a normative XML Syntax which presents a number of advantages for publishing ontologies on the Web. In particular, the XML Syntax is easy to parse and process; also, XML is a widely adopted format on the Web and it is supported by a variety of tools. Existing OWL 1 ontologies can still be used and further developed with OWL 2 implementations. Therefore, an important requirement in the development of OWL 2 was to maintain backwards compatibility with the RDF syntax of OWL 1. At the same time, a major design goal in OWL 2 was to clean up problems in the definition of OWL 1. Every OWL 1 DL ontology written in RDF is also a valid OWL 2 ontology.
Issue	Problem in OWL 1	Solution in OWL 2

2.5.1.3.3. Metamodelado Véase la tabla 2.25.

Tabla 2.25.: Problemas de OWL 1 en lo que a metamodelado se refiere.

Issue	Problem in OWL 1	Solution in OWL 2
Frame-Based Paradigm	OWL 1 Abstract Syntax is heavily based in frames. The problem is that the frames are somewhat different to Description Logic axioms, been this a source of confusion for developers.	The structure of OWL 2 ontologies has been unambiguously specified using OMG'sMeta-Object Facility (MOF). The classes of the MOF metamodel describe the canonical structure of OWL 2 ontologies in a way that is independent of the syntax used to serialize the ontologies. The MOF metamodel for OWL 2 can be seen as analogous to the Document Object Model (DOM) specification 16 for XML. It unambiguously specifies what OWL 2 ontologies are in terms of their structure and thus makes the specification precise. In addition to the MOF metamodel, the OWL 2 specification defines a Functional-Style Syntax as a simple encoding of the metamodel. OWL 2 departs in its conceptual design and in syntax from the OWL 1 Abstract Syntax. The main difference with respect to the OWL 1 Abstract Syntax is that the Functional-Style Syntax of OWL 2 does not contain the frame-like syntactic constructs of OWL 1; instead, ontology entities are described at a more fine-grained level using axioms.
Alignment with DL Constructs	Even though OWL 1 is based on DLs, the constructs of OWL1 Abstract Syntax do not completely correspond to the constructs of DLs.	
Continúa en la siguiente página		

Continuación de la tabla		
Issue	Problem in OWL 1	Solution in OWL 2
Determining the Types of Ontology Entities	Neither Abstract Syntax nor OWL 1 RDF is fully context-fre, that is, an axiom containing a URI often does not contain sufficient information to disambiguate the type of ontology entity (i.e., a concept, a property, or an individual) that the URI refers to. To disambiguate the syntax, OWL 1 DL relies on a strict separation of the vocabulary into individuals, classes, and data and object properties. The specification of OWL 1 DL, however, does not precisely specify how to enforce this separation at the syntactic level. Thus, whereas the semantics of OWL 1 DL requires strict typing of all names, the syntax does not enforce it, which can prevent certain ontologies from being interpreted.	OWL 2 introduces the notion of declarations. All entities can, and sometimes even must, be declared in an OWL 2 ontology. OWL 2 imposes certain typing constraints on ontologies. Roughly speaking, the sets of object, data, and annotation properties, as well as the sets of classes and datatypes in an OWL 2 ontology must be mutually disjoint.
Problems with OWL 1 RDF	The vast majority of OWL 1 ontologies have been written in OWL1 RDF syntax; this syntax has, however, shown itself to be quite difficult to use in practice. The main difficulty is that RDF represents everything using triples, which specify relations hips between pairs of objects. In contrast, many OWL 1 constructs cannot be represented using triples without the introduction of new objects.	In order to provide an alternative to RDF, OWL 2 comes with a normative XML Syntax which presents a number of advantages for publishing ontologies on the Web. In particular, the XML Syntax is easy to parse and process; also, XML is a widely adopted format on the Web and it is supported by a variety of tools. Existing OWL 1 ontologies can still be used and further developed with OWL 2 implementations. Therefore, an important requirement in the development of OWL 2 was to maintain backwards compatibility with the RDF syntax of OWL 1. At the same time, a major design goal in OWL 2 was to clean up problems in the definition of OWL 1. Every OWL 1 DL ontology written in RDF is also a valid OWL 2 ontology.
Issue	Problem in OWL 1	Solution in OWL 2

2.5.1.3.4. Versionado e importado Véase la tabla 2.26.

2.5.1.3.5. Anotaciones. Véase la tabla 2.27.

Issue	Problem in OWL 1	Solution in OWL 2
Naming problems	OWL 1 provides a basic mechanism that allows an ontology to import another ontology, and thus gain access to all entities and axioms in it. However, the two ontologies are kept physically separate. The importing ontology is required to contain a URI pointing to the location of the imported ontology, and this location must match with the name of it. However in many realistic use cases, the location of an ontology does not correspond with its name.	The import mechanism of OWL 2 Is also "by name and location." The main difference to OWL 1 is that OWL 2 allows implementations to provide a suitable redirection mechanism: when requested to access an ontology from a location u, a tool can redirect u to a different location v and retrieve the ontology from there. The accessed ontology should, however, be treated just as though it had been retrieved from u. For example, relative URIs in the ontology should be resolved as if the ontology had been retrieved from u, and the ontology URI, if present, should be equal to u. The design of concrete redirection mechanisms is left entirely to OWL 2 implementations. OWL 2 also provides a simple method for the management of ontology versions. In addition to an ontology URI, an OWL 2 ontology can contain a version URI, which identifies the version of the ontology. In such cases, the ontology URI identifies an ontology series—a set of all versions of a particular ontology—whereas the version URI identifies a particular version in the series.

Tabla 2.26.: Versionado e importado.

Issue	Problem in OWL 1	Solution in OWL 2
Inadequate annotation system	The annotation system of OWL 1 has been found to be inadequate for many applications. In particular, OWL 1 does not allow axioms to be annotated, which can be necessary	In OWL 2 it is possible to annotate axioms as well as ontologies and entities. This can be used to capture additional information about axioms, such as their provenance or certainly values.

Tabla 2.27.: Anotaciones.

2.5.1.3.6. Semánticas OWL La existencia de dos semánticas para OWL 1 (semántica directa de modelado teórico basada en la semántica de SHOIN(D) y el modelo teórico RDF) ha causado muchos problemas (tabla 2.28):

Issue	Problem in OWL 1	Solution in OWL 2
OWL 1 DL Semantics	The OWL 1 DL specification provides an explicit (and quite complex) definition of the semantics, which does not straightforwardly correspond to SHOIN(D). This caused a number of problems regarding presentation and understandability. A source of confusion is the ability to use unnamed individuals in OWL1 facts (this feature was inspired by RDF blank nodes). Unnamed individuals are not directly available in SHOIN(D), and it is not obvious how to simulate them.	OWL 2 defines a clean direct model theoretic semantics that clearly corresponds to the description logic SROIQ(D). At the moment, OWL 2 does not provide an RDF style semantics. Therefore, in OWL 2, the transformation of ontologies in Functional-Style Syntax into RDF graphs is a purely syntactic process: the triples obtained by the RDF serialization are not intended to be interpreted directly, and the meaning of the RDF semantics on them does not define the meaning of the corresponding OWL 2 ontology.
RDF-Compatible Semantics	OWL 1 ontologies written as RDF graphs can be interpreted using an extension of the RDF semantics. This semantics, however, has proved to be extremely complex to understand, use, and implement.	
Relating the DL and the RDF Semantics	A key idea behind the two semantics of OWL 1 was that they should be "equivalent" on OWL 1 DL and OWL 1 Lite ontologies. Maintaining this compatibility becomes increasingly problematical with increasing expressive power.	

Tabla 2.28.: Semánticas OWL

2.5.1.3.7. OWL 1 Full. Véase la tabla 2.29.

2.5.1.3.8. OWL 1 Lite. Véase la tabla 2.30.

2.5.1.3.9. Validación de tipos OWL. La validación de tipos OWL (species validation) Es el problema de determinar cuándo una ontología de OWL 1 es OWL 1 Lite, OWL 1 DL u OWL 1 Full.

2.5.2 Motor de inferencia

A la hora de diseñar el motor de inferencia que regirá el funcionamiento del módulo de razonamiento de ISMED se presentan tres opciones:

1. Tener un motor semántico y un motor de reglas separados (figura 2.31).

Issue	Problem in OWL 1	Solution in OWL 2
Undecidabilty of RDF graphs	Each RDF graph is a syntactically valid OWL 1 Full ontology. The basic reasoning problems for the RDF-compatible semantics are undecidable. no complete implementation of OWL 1 Full currently exists, and it is not clear whether OWL 1 Full can be implemented in practice.	See 3.8

Tabla 2.29.: OWL Full.

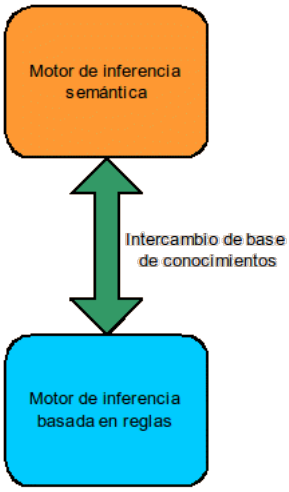


Figura 2.31.: Motor de reglas y semántico separados.

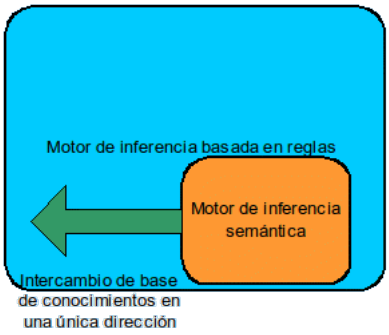


Figura 2.32.: Motor semántico embebido.

Issue	Problem in OWL 1	Solution in OWL 2
Computational complexity	Even though OWL 1 Lite seems to be much simpler than OWL 1 DL, most of the complexity of OWL1 DL can be captured due to implicit negations in axioms. In fact, of all the OWL 1 DL constructors, only nominals and cardinality restrictions with cardinality larger than one cannot be captured in OWL 1 Lite. From a computational perspective, basic reasoning problems are only slightly less complex for OWL 1 Lite than for OWL 1 DL.	The EL++ Profile has been designed to allow for efficient reasoning with large terminologies. The main reasoning service of interest is classification, the problem of computing the subclass relation between all the classes in an ontology. Reasoning in EL++ can be implemented in time that is polynomial in the size of the ontology. In order to achieve tractability, EL++ disallows the use of negation, disjunction, AllValuesFrom restrictions and cardinality restrictions. The DL-Lite Profile has been designed to allow for efficient reasoning with large amounts of data structured according to relatively simple schemata. The main reasoning service in the DL-Lite profile is conjunctive query answering. DLLite forbids the use of disjunction and AllValuesFrom restrictions, as well as certain other features that require recursive query evaluation. OWL-R is intended to be used by OWL 2 users who are willing to trade some of the available expressivity in OWL 2 for the ability to implement reasoning using rule systems based on forward chaining. OWL-R allows for most constructs of OWL 2; however, to allow for rule-based implementations of reasoning, the way these constructs can be used in axioms has been estricted. These restrictions ensure that a reasoning engine only needs to reason with the individuals that occur explicitly in the ontology. Thus, in OWL-R DL it is not possible to use SomeValuesFrom restrictions on the right-hand side of a subclass axiom, as this would imply the existence of an "anonymous" individual—that is, an individual that is not explicitly known by name. Similar principles have been followed in the design of Description Logic Programs (DLP).

Tabla 2.30.: OWL1 Lite.

Issue	Problem in OWL 1	Solution in OWL 2
Differentiation between OWL DL and OWL Lite	Determining whether an ontology in RDF is an OWL 1 Lite or an OWL 1 DL ontology becomes very hard in practice since it involves 'guessing' ontologies in Abstract Syntax and checking whether their normative transformation into RDF yields precisely the original ontology.	OWL 2 specification identifies several profiles, trimmed down versions of the full language that trade some expressive power for efficiency of reasoning, tool developers can easily use the corresponding grammars to create tools for checking which profile an ontology belongs to.
Species Validation and Imports	Imports can interact with OWL 1 species in a quite unpredictable and unintuitive way, changing the "species" of the importing ontology.	

Tabla 2.31.: Validación de tipos OWL.

2. Embeber el motor semántico dentro del motor de reglas (figura 2.32).
3. Expresar la semántica de RDF y OWL como reglas y sólo utilizar el motor de reglas (figura 2.33).

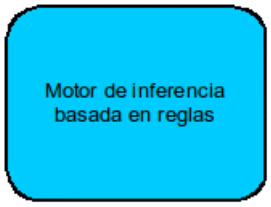


Figura 2.33.: Un único motor de reglas.

Para decidir entre una de las tres opciones se han analizado las ventajas e inconvenientes de cada una de ellas, que se representan en la siguiente tabla:

Analizando la tabla se ha llegado a la conclusión que el diseño más adecuado para el sistema es el tercero, ya que habrá cambios frecuentes en la base de conocimiento que serán provocados tanto por la inferencia de las reglas como por la inferencia semántica. Además el posible inconveniente de la tercera opción de diseño se convierte en este caso en una ventaja ya que permite seleccionar que reglas semánticas va a implementar el sistema. De esta manera no se implementaran aquellas reglas que la ontología diseñada no vaya a utilizar por lo que el número de comprobaciones que realizará el motor será menor.

El motor de reglas utilizado podría ser el propio motor de reglas incorporado en JENA, siempre que se pueda utilizar desde dispositivos empotrados y móviles, ya que tiene una serie de características que resultan necesarias para el sistema:

- Puede trabajar con una base de conocimientos expresada como una ontología.
- Permite expresar las reglas en formato de triples: (?a esUn Mamífero) ->(?a esUn Animal).
- Permite encaminamiento hacia delante, hacia atrás e híbrido.
- Permite introducir nuevos predicados en la ontología (la base de conocimiento) de manera sencilla.

El motor realizará tres labores de inferencia que se dividirán en dos fases. En una primera fase el motor extraerá el conocimiento implícito en la ontología y procesará sus relaciones semánticas para

Motor	Ventajas	Inconvenientes
Motores separados	El motor semántico puede ser un motor externo como Pellet [20] o Racer [21], por lo que la inferencia semántica sería rápida y eficiente.	Habría que implementar manualmente el intercambio de la base de conocimiento entre ambos motores. Este intercambio sería necesario cada vez que la base de conocimiento sea modificada de alguna manera, lo que disminuiría el rendimiento del sistema.
Motor embebido	El intercambio de la base de conocimiento se haría de manera automática.	El intercambio de conocimiento sólo podría hacerse en una dirección, por lo que el motor semántico no podría observar los cambios hechos en la base de conocimiento por el motor de reglas. La eficiencia baja al estar un motor embebido en otro.
Sólo motor de reglas	No es necesario intercambio de la base de conocimiento. Se puede especificar que reglas semánticas que se deben tener en cuenta, pudiendo eliminar las que no sean necesarias en el sistema y aumentando el rendimiento.	Deben de especificarse las reglas semánticas manualmente.

Tabla 2.32.: Comparación entre diseños de motores de inferencia.

expandir la base de conocimientos. Después, haciendo uso de esta base de conocimientos expandida inferirá las acciones a llevar a cabo por el sistema.

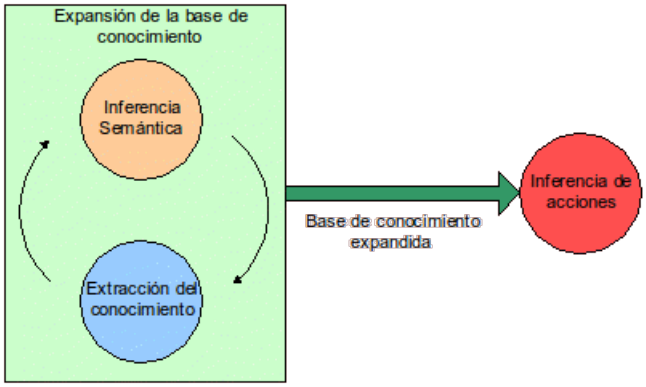


Figura 2.34.: Proceso de inferencia.

2.5.3 Comparativa de razonadores

En la siguiente tabla se muestra una comparación de los razonadores más usados del mercado. Para construir la tabla se han tenido en cuenta las siguientes características:

- Expresividad: la expresividad de la lógica de descripción permitida.
- OWL-DL Entailment: la capacidad para realizar el entailment lógico necesario para permitir la teoría de modelos OWL-DL.

- Revisión de la consistencia: la capacidad para verificar la consistencia de la ontología.
- Soporte DIG [10]: índica si provee una interfaz de acceso al razonador de lógica de descripciones.
- Soporte SPARQL: permite usar el lenguaje de consultas SPARQL.
- Soporte de reglas: permite ejecutar reglas usando la ontología como base del conocimiento.
- Documentación disponible: algunos de los razonadores analizados no tienen ninguna documentación disponible.
- Versión: versión analizada del razonador.
- Licencia: licencia del razonador.

	Pellet [20]	KAON2 [18]	RacerPro [21]	Jena [16]	FaCT++ [12]	SweetRules [25]	Bossam [8]	Hoolet [15]
Expressivity	SROIQ(D)	SHIQ(D)	SHIQ	Incomplete DL support	SROIQ(D)	Not Applicable	?	?
OWL-DL Entailment	Yes	Yes	Yes	Incomplete. Jena does not implement a complete OWL DL model	Yes	No	Partial (still under development)	Yes
Consistency checking	Yes	?	Yes	Incomplete. Jena does not implement a complete OWL DL model	?	No	?	Yes
Consistency checking	Yes	?	Yes	Incomplete. Jena does not implement a complete OWL DL model	?	No	?	Yes
SPARQL Support	Yes	Yes	No	Yes	No	No	No	No
Rule Support	Yes (SWRL – DL Safe Rules)	Yes (SWRL)	Yes (SWRL)	Yes (Own rule format)	No	Yes (SWRL, RuleML, Jess)	Yes (SWRL & own rule format)	Yes (SWRL)
Documentation available	Yes	?	Yes	Yes	?	Yes	Yes	?
Version	2.0 RC1	?	1.9.2	2.5.4	1.1.8	2.1	0.9b45	?
Licensing	Free/ open-source & Non-Free/ closed-source	Free/Closed Source	Non-Free/ closed-source	Free/ open-source	Free/ open-source	Free/ open-source	Free/ closed-source	Free/ open-source

Tabla 2.33.: Comparativa de razonadores.

2.5.3.1 Soporte SWRL

Algunas de las plataformas analizadas permiten el uso de SWRL [600] (Semantic Web Rule Language). SWRL es parte de una propuesta del W3C que combina OWL con RuleML. A pesar del soporte de SWRL, no todos los razonadores permiten todas las características que esté incorpora al usar reglas. Existen ocho tipos de built-ins en la especificación SWRL: Built-Ins para comparaciones, Built-Ins matemáticos, Built-Ins para valores boleanos, Built-Ins para cadenas de texto, Built-Ins para fechas, tiempo y duración, Built-Ins para URIs, Built-Ins para listas and Built-Ins personalizadas.

	Pellet [20]	KAON2 [18]	RacerPro [21]	SweetRules [25]	Bossam [8]	Hoolet [15]
SWRL Math Built-Ins	No	Yes	No	Yes	Partial	No
SWRL String Built-Ins	No	Yes	No	Yes	Partial	No
SWRL Comparison Built-Ins	Partial	Yes	No	Yes	No	No
SWRL Boolean Built-Ins	No	Yes	No	Yes	No	No
SWRL Date, Time and Duration Built-Ins	No	Yes	No	Yes	No	No
SWRL URI Built-Ins	No	Yes	No	Yes	No	No
SWRL Lists Built-Ins	No	Yes	No	No	No	No
SWRL custom Built-Ins	Partial	Yes (Programmed in Java)	No	?	Yes (Programmed in Java)	No

Tabla 2.34.: Comparativa de razonadores.

3. DISEÑO

Este capítulo describe el diseño de la plataforma ISMED a partir de las conclusiones derivadas del estado del arte efectuadas en el anterior capítulo.

3.1 Modelado y coordinación semántica

3.1.1 Alternativas seleccionadas

Dadas las características descritas en la sección 2.1.4, a continuación se enumerarán los dispositivos hardware y tecnologías software que se tomarán como punto de partida para desarrollar el módulo de modelado y coordinación semántica.

3.1.1.1 Plataformas empotradas elegidas

A continuación se enumeran las plataformas empotradas para las que se planteó desarrollar el middleware ISMED.

Además, cabe destacar que tal y como se adelantó en la memoria del 2008, se ha concluido que es inviable implementar en ellas todos los módulos funcionales que componen ISMED por falta de recursos computacionales.

Pese a ello, se ha creado para todas ellas una versión del módulo de coordinación semántica, de forma que puedan proveer los datos que dichas plataformas capturen al espacio Triple Space al que se encuentren unidas.

Para aquellas plataformas que no disponen capacidad de computo suficiente (SunSpot y XBee), se optó por utilizar una arquitectura híbrida en la que elementos de la red que pueden ofrecer de mayor capacidad computacional asistan al middleware desplegado en dichas plataformas.

- Sunspot [24]
- XBee [263]. De las tres alternativas enumeradas, esta es la única que no se incluyó en el estado del arte de la sección 2.1.4. Los sensores XBee utilizados se comunican de forma transparente para el desarrollador con un Gateway [262]. Dicho Gateway tiene conectividad IP (mediante wireless o Ethernet) y en él se pueden cargar programas hechos en Python que manejarán los datos aportados por los sensores. Este sensor se ha escogido por la sencillez y transparencia con la que permite capturar datos del entorno, permitiendo al desarrollador centrarse en el Gateway y no preocuparse por el protocolo de comunicación entre este y los sensores.
- Gumstix [13]

3.1.1.2 Dispositivos móviles

Aparte de las plataformas de sistemas empotrados mencionadas en el apartado anterior también se hará uso de teléfonos móviles. Éstos son dispositivos de uso común que formarán parte del ecosistema de dispositivos cooperativos que pretende coordinar el middleware de ISMED.

- Nokia Series 60 soportando CLCD 1.1 Java ME. Concretamente se hará uso de dispositivos Nokia N96.

3.1.1.3 Librerías y tecnologías

A continuación se enumeran varias librerías que se han usado para implementar el módulo de coordinación de ISMED.

- Jxta

Se ha escogido por tratarse de un protocolo bien definido con versiones para distintas plataformas, que permite gestionar redes P2P con un alto nivel de abstracción. En concreto, se usarán las siguientes versiones:

- JXTA-JXSE 2.5, se trata de la implementación principal de Jxta realizada en Java, que será usada en aquellas plataformas empotradas que lo permitan y en las máquinas que sirvan de intermediarias para dispositivos demasiado simples.
- JXTA-JXME 2.5, es la versión para dispositivos móviles que usen Java ME, tanto CDC como CLDC. La principal diferencia con la versión que en la memoria del 2008 se indicó que se iba a usar, es que la versión 2.5 permite a los dispositivos móviles CLDC no depender de Proxys que los comuniquen con el resto de nodos Jxta. Si bien es cierto que debido a las limitaciones impuestas por los dispositivos móviles no se soportan todos los protocolos de la versión JXSE, sí que implementa los suficientes como para que un nodo móvil pueda comunicarse con el resto de nodos de la red mediante un Rendezvous.

- Repositorios semánticos

  Se usa Sesame (con Owlim, opcionalmente), para aquellas plataformas empotradas que permitan usar Java y en las máquinas que sirvan de intermediarias para dispositivos demasiado simples.

- Jena

  Se usa para filtrar resultados de consultas complejas y para, apoyándose en su procesador de SPARQL ARQ, descomponer dichas consultas SPARQ en templates que todos los nodos pueden interpretar.

- Microjena

  Se usa para expresar la información semántica en los dispositivos con Java ME. Motores de inferencia limitados Se analizará la posibilidad de crearemos nuestro propio motor de inferencia, considerando la adopción de otro hipotético motor de inferencia más completo para entornos empotrados y programado en Java, siempre que se disponga de una versión del mismo antes del fin del proyecto.

3.1.1.4 Reutilización de proyectos analizados

De los proyectos analizados, TripCom tiene una arquitectura demasiado grande que no encaja nada bien con las dimensiones más reducidas y localizadas de la red que se establecerá entre los dispositivos de ISMED. Además, en el momento en el que se comenzó a implementar ISMED, no existía una versión lo suficientemente madura.

Por otro lado, tsc++ tiene un enfoque más sencillo que hace uso de los distintos super- nodos de las redes Jxta para comunicar entre si distintas redes locales, pero aún así parece consumir muchos recursos. El enfoque plenamente distribuido que utiliza, además, se adapta perfectamente a redes muy dinámicas donde diversos dispositivos entran y salen contínuamente y facilita el despliegue de nuevos dispositivos en el entorno.

Por ello, pese a no reutilizar el propio tsc++, si que resultó un buen punto de partida, y fue usado a modo de referencia, adaptándolo a las mayores limitaciones impuestas por la naturaleza de dispositivos y mecanismos de comunicación dentro de ISMED. Además, se valoró positivamente que los dispositivos empotrados gobernados por el middleware de ISMED pudieran cooperar con aquellos desarrollados usando tsc++.

3.1.1.5 Análisis de la aplicabilidad de las técnicas de TripCom a ISMED

Pese a que tal y como se ha comentado en el anterior epígrafe, el módulo de coordinación semántica está basado originalmente en la librería tsc++, a lo largo del año 2009 finalizó el proyecto TripCom (ver epígrafe 2.1.2.3.3), y con ello quedó patente la capacidad del mismo para resolver algunos de los problemas detectados en tsc++.

Así, a continuación se identificarán aquellas características o técnicas de TripCom que se pueden aplicar a ISMED para paliar alguna de las limitaciones detectadas en tsc++.

3.1.1.5.1. Diferencias entre TripCom e ISMED. En primer lugar, cabe comentar que existen tres **estrategias** principales a la hora de almacenar y devolver datos en sistemas **distribuidos**: centralización, flooding e índices distribuidos [599].

Así, mientras TripCom usa PGrid que está basado en **índices distribuidos**, una aproximación completamente descentralizada dónde los peers manejan referencias a otros peers y mapean la información que estos les ofrecen, ISMED está basado en tsc++ que a su vez usa Jxta que se ayuda del **flooding**, dónde los peers no almacenan información sobre los datos almacenados por otros peers y se propaga a todos los nodos participantes.

Concretamente tsc++ utiliza negative broadcast, dónde las operaciones de escritura son locales y las consultas son distribuidas (en positive broadcast las consultas son locales y la escritura se propaga a todos los nodos).

En segundo lugar, cabe destacar que si bien ISMED está pensado para ser ejecutado en un **entorno local** dónde los retrasos por comunicación entre dispositivos serán más reducidos y homogéneos entre sí, la filosofía de TripCom busca crear un "internet para máquinas" en el que dichas máquinas se ubican en redes WAN y que sea altamente escalable.

Bajo este entorno de redes locales en el que ISMED trabajará, no es necesario tener diferentes estrategias para optimizar la búsqueda del peer que debe resolver una respuesta concreta mediante índices distribuidos y permite utilizar un enfoque mucho más sencillo como el flooding en su lugar. En tercer lugar, los kernels del proyecto TripCom están compuestos de componentes que no tienen porque ejecutarse necesariamente en la misma máquina y que aíslan el proyecto en partes de alta complejidad. No todas las partes tienen que ser implementadas para ofrecer una API que siga el paradigma Triple Space, y en ese sentido el hecho de que ISMED deba ejecutarse en **dispositivos de una capacidad de cómputo reducida** hace interesante reducir al máximo el tamaño del kernel a utilizar dejando en el sólo lo estrictamente necesario para tener un sistema funcional.

En ese aspecto, cabe destacar también que el enfoque de TripCom aboga por unos **clientes** que son **ajenos** a toda la distribución y manejo de los datos, y sólo interactúan con un servidor (uno de los kernels) consultando y recibiendo las consultas a través del API que este les proporciona. Mientras tanto todo kernel de tsc++ es en sí un cliente con capacidad de hacer consultas sobre un espacio. En tsc++ el nodo que sigue el paradigma Triple Space es en sí quien pregunta al espacio, no un simple intermediario de los clientes con TripleSpace como en TripCom.

Finalmente, cabe destacar que los dispositivos pueden entrar y salir con relativa facilidad de la red de sensores que se formará en ISMED, mientras que en TripCom pese a que los kernels pueden incorporarse a la red y salir de ella, este hecho es más complejo y por lo tanto no es un proceso pensado para ser realizado con asiduidad.

3.1.1.5.2. Limitaciones de ISMED. Se han detectado las siguientes limitaciones a la hora de adaptar el paradigma del Triple Space Computing al proyecto ISMED:

- **Razonamiento distribuido.** El poder razonar sobre el conjunto de la información existente en un espacio completo es una problemática de alto nivel de complejidad que no ha sido abordado tampoco por TripCom.
- **Razonamiento en dispositivos móviles.** Punto que se tratará en el epígrafe correspondiente al módulo de razonamiento.
- **Limitación en la expresividad de las consultas.** Actualmente ISMED trabaja con consultas realizadas con patrones de tripletas individuales. Al igual que TripCom con su Core API y su Extended API, se ha planteado paliar esta solución en ISMED con una serie de medidas:
  - Aumentar la expresividad del lenguaje de consultas mediante la inclusión de primitivas para la comparación de cifras y arrays de caracteres.
  - Permitir usar consultas SPARQL en aquellos kernels que tengan capacidad para responderlas localmente.
- **Consultas distribuidas.** Se puede tratar de adaptar los métodos usados en el QPP de TripCom.

3.1.1.5.3. Adaptación de funcionalidades de TripCom. Siendo conscientes de las diferencias que existen entre TripCom y el middleware a desarrollar en ISMED, cabe hacer un análisis de las funcionalidades que ofrece TripCom pero no ISMED con su versión de tsc++, para pensar cuál de ellas sería conveniente implementar en ISMED y de qué forma se podría adaptar.

Los métodos para identificar kernels descritos en él para el DM de TripCom no son aplicables a ISMED por la diferencia en su estrategia de distribución vista anteriormente (ISMED no puede dirigir una consulta a un kernel concreto).

Sin embargo, alguna de las estrategias usadas en el QPP de TripCom podría ser usada para realizar consultas distribuidas. Así, de las dos partes principales del QPP [598]:

- **Inconsistency reasoner.** En una primera iteración no se planea implementar este módulo, dado que no soluciona ninguna de las limitaciones descritas anteriormente.
- **Query Optimizer.** De este componente no se implementará la estimación de coste y la reescritura de consultas (por ser más difíciles de implementar en un sistema de flooding). La descomposición de consultas y construcción de respuestas, sin embargo, parecen buenas estrategias para permitir las consultas distribuidas.

  Así, partiendo de una consulta SPARQL compleja, se puede descomponer en templates de tipo sujeto-predicado-objeto, que cualquier nodo de tsc++ sabrá responder.

El problema, una vez más, es que no cualquier dispositivo tendrá la capacidad suficiente como para realizar esa descomposición (que en TripCom se hace haciendo uso del procesador SPARQL de Jena ARQ). Para salvar esa limitación, y guiándonos por el diseño de TripCom, nos apoyaremos en dos niveles de APIs distintos.

El API más básico de todos ellos será el de TripleSpace básico. El extendido, tendrá aquellas primitivas que se apoyen en la descomposición de consultas.

Otra solución un poco más compleja, sería la de implementar ambos APIs en todos los dispositivos, pero hacer que aquellos que no sean capaces de descomponer las consultas las redirigiesen a aquellos dispositivos que si pudiesen hacerlo. De esta forma, estos últimos nodos capaces, serían los responsables de obtener las respuestas y reenviarlas al nodo limitado que redirigió la pregunta inicialmente.

En un primer momento, se descartará la segunda solución por la complejidad que implica, adoptando la de los dos APIs.

3.1.2 API de Triple Space

El desarrollador deberá interactuar con una API de Triple Space para coordinar los distintos módulos que cree. A pesar de que originalmente nos inspiramos en el API de tsc++, esta se adaptó para tener en cuenta la naturaleza y restricciones de entornos ubicuos. En la tabla 5.5, se mostrarán las primitivas de las que se compone este API.

Las primitivas de gestión de espacios permiten a los nodos compartir información con diferentes grupos de nodos. *Query* devuelve todas las tripletas que concuerdan con un patrón dentro del espacio facilitado, mientras *queryMultiple* divide una consulta SPARQL en patrones simples que se envían a todos los nodos, las respuestas obtenidas de esos nodos se unen y se procesa de nuevo la consulta sobre ellos para poder obtener resultados nuevos. *Read* devuelve un único grafo que contiene al menos una tripleta que concuerda con el patrón dado y *take* hace lo mismo, pero eliminando dicho grafo del repositorio semántico del nodo que lo contenía. Tanto *read* como *take* están sobrecargados para devolver un grafo que es identificado por su URI.

A pesa de que la primitiva *write* se usaba para escribir tripletas en el repositorio local de cada nodo, devolviendo la URI que identificaba el nuevo grafo creado y almacenado, su implementación ha sido modificada para no seguir siempre la estrategia de *negative broadcasting* (escritura en local y consulta propagada al resto de miembros de un espacio). *Demand* permite al desarrollador anunciar que otro nodo que puede ser responsable de los grafos que concuerdan con el un patrón dado por un periodo de tiempo definido por el parámetro *timeout*. La solución se explicará en detalle en la sección 3.1.4.4.

La primitiva *subscribe* se usa para mantener a un nodo al corriente de las notificaciones que se han realizado sobre un patrón determinado o la particularización del mismo. *Advertise* permite a

Figura 3.1.: Funcionamiento esquemático de la primitiva query.

Figura 3.2.: Funcionamiento esquemático de la primitiva queryMultiple.

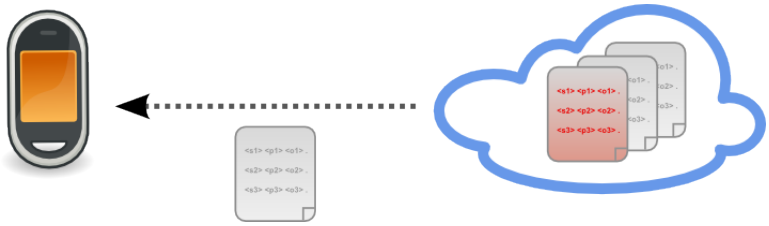


Figura 3.3.: Funcionamiento esquemático de la primitiva read.

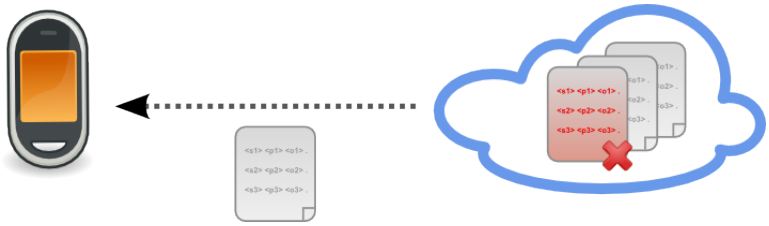


Figura 3.4.: Funcionamiento esquemático de la primitiva take.

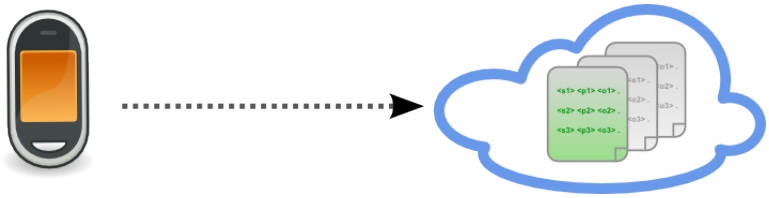


Figura 3.5.: Funcionamiento esquemático de la primitiva write.

Space management	createSpace(space) joinSpace(space) leaveSpace(space)
Querying	query(space,template): triples read(space,graph): triples read(space,template): triples take(space,graph): triples take(space,template): triples queryMultiple(space,sparql): triples
Writing	write(space,triples): URI demand(space,template,timeout)
Subscriptions	subscribe(space,template): URI unsubcribe(space,URI) advertise(space,template): URI unadvertise(space,URI)
Services	register(space,service) unregister(space,service) invoke(space,invocation,listener): URI

Tabla 3.1.: Primitivas del middleware basado en Triple Space diseñado agrupadas por su naturaleza. *space* es la URI que identifica el conocimiento del grupo de nodos, *graph* es una URI que identifica un conjunto de tripletas escritas en un espacio, *triples* es un conjunto de triples y *template* expresa una secuencia de patrones de tripletas adyacente que especifican la clausula WHERE-clauses de una consulta SPARQL. *Service* e *invocation* son interfaces que permiten definir los datos necesarios para definir un servicio y su invocación.

un nodo propagar la notificación a todos los nodos del espacio suscritos a ese tipo de patrón para avisarles de algo. Estas primitivas se pueden usar para construir un sistema de notificación sobre el espacio.

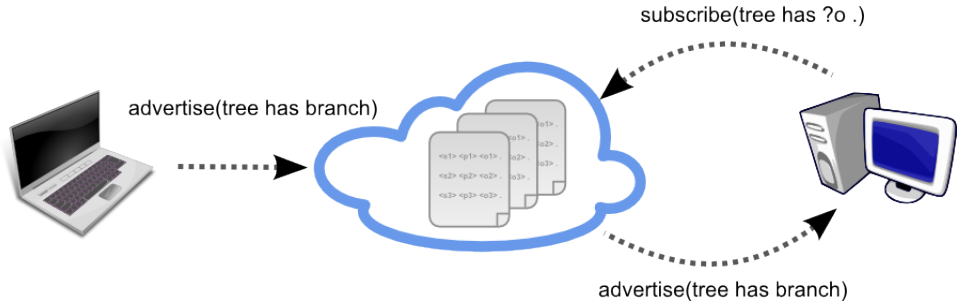


Figura 3.6.: Funcionamiento esquemático de las primitivas subscribe y advertise.

3.1.2.1 API de Servicios

La necesidad de crear esta API de Servicios surgió a raíz de ciertos problemas encontrados a la hora de modificar actuadores:

- Seguridad: al no implementar tsc++ ningún tipo de control de acceso, cualquiera puede modificar, por error, más conocimiento del que desea.
- Concurrencia: si dos nodos diferentes modifican la misma información al mismo tiempo, no se puede determinar qué información se va a tener en cuenta.
- Localización de la información: por la naturaleza de tsc++, cuando una información que inicialmente pertenece al *nodo a* es modificada por el *nodo b*, se almacena en el *nodo b* en

lugar de en el *nodo a*. Si el *nodo b* abandona el espacio esta información desaparecerá, por lo que sería lógico que la información acerca de un actuador fuese almacenada en el nodo que lo gestiona.

Para solucionar esta problemática se ha creado una simple invocación de Servicios, independiente de la manera en la que los servicios semánticos son definidos (para este proyecto se ha creado su propio lenguaje de definición, pero podrían utilizarse otros). Este proceso se realiza en tres pasos:

- Primero, el proveedor de servicios registra su servicio en el espacio, como puede verse en la figura 3.12a).
- En segundo lugar, el consumidor descubre el servicio realizando una *query* en el espacio, creando una invocación a través del patrón *master-worker* y anunciándolo. Una invocación suele estar compuesta por una URI y ciertos parámetros de entrada.
- Por último, el proveedor de servicios, que está suscrito a las plantillas de invocación de servicios que puede atender, notificará el evento determinado y recuperará la información de entrada ejecutando el servicio (usualmente, la modificación del entorno implica la utilización de un actuador). Cuando se completa la invocación, el proveedor de servicios escribe en el espacio las tripletas con el resultado de la acción, para notificar al consumidor los resultados de su invocación.

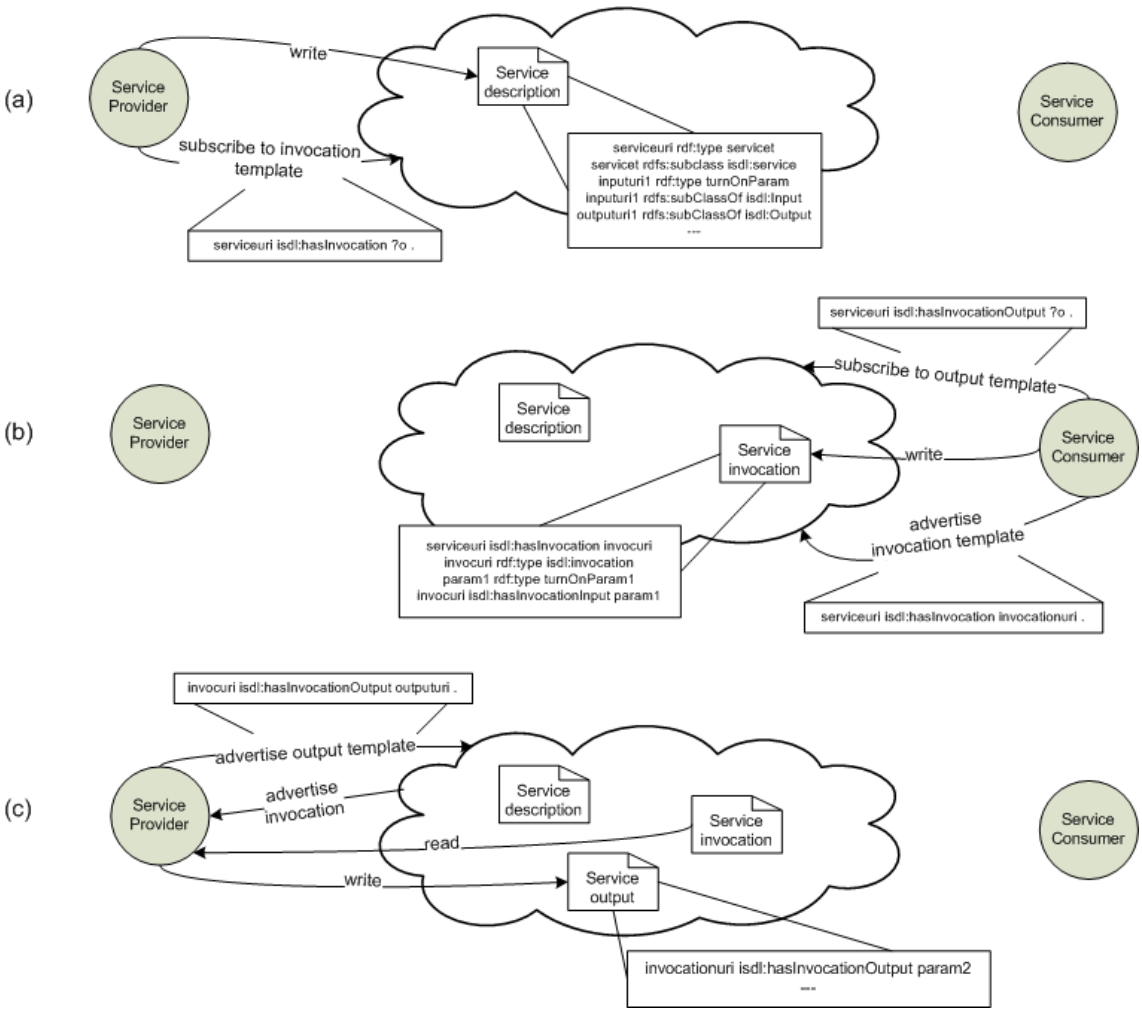


Figura 3.7.: Services over tsc++: a) registration, b) invocation from the consumer point of view and c) invocation from the service provider point of view.

3.1.3 Arquitectura propuesta

En base a lo indicado se pretenden usar las siguientes librerías en los siguientes tipos de hardware:

- Móviles CLDC: JavaME, microjena (manejo de tripletas y filtrado básico), jxme-proxyless y repositorio normal (RecordStore) o ficheros para persistir dichas tripletas.
- Sunspot, los sunspots se comunicarán con su base y en ellá se usará un gateway para Web of Things mediante el cual se accederá a los valores de los sensores y actuadores vía REST en desde un nodo miembro del espacio TS.
- XBee, los sensores se comunicarán con un gateway en el que se ofrecerña un interfaz REST de acceso a los sensores que será consumida desde un nodo que hará de intermediario con el espacio TS.
- Gumstix: JXSE y Sesame.
  - Podría incluso hacer de nodo intermediario de dispositivos embebidos o tener el Rendezvous que los moviles necesitan JXME.
  - Cabe destacar que de todas los dispositivos inicialmente planteados, este fue el único cuya implementación no se abordó finalmente. En cualquier caso, la propia versión tscSE debería funcionar en este tipo de dispositivos.

A modo de resumen sobre lo comentado previamente, se muestra un esquema de la arquitectura propuesta y las tecnologías a aplicar en cada uno de los dispositivos seleccionados. Esta arquitectura es el punto de partida para el diseño del resto de módulos que conforman el middleware ISMED: descubrimiento, composición, razonamiento y aprendizaje. Obsérvese que dadas las limitadas capacidades computacionales de las plataformas empotradas consideradas, al menos en una primera fase, tendremos que apoyarnos en diversas pasarelas que permitan la intermediación entre el mundo TCP/IP usado por los protocolos de coordinación adoptados en ISMED y los mundos de redes puras ad-hoc Zigbee usados por plataformas como Gumstix o SunSPOT. En esas pasarelas con mayor capacidad computacional podrá razonarse, aplicar algoritmos de aprendizaje e incluso en muchas ocasiones guardar en modo persistente el estado semántico generado por los distintos dispositivos que conforman el ecosistema de ISMED.

3.1.3.1 Gateway para dispositivos Embebidos

Para posibilitar la integración de las SunSPOTs en el Triple Space, se ha adaptado el gateway diseñado para cumplir el paradigma *web of things* [353]. Este gateway permite acceder a los sensores y actuadores de las SunSPOTs a través de servicios REST.

Además de las representaciones proporcionadas por el servidor (XML, JSON y HTML), se ha añadido una representación más, que devuelve un conjunto de tripletas que contiene la información semántica del recurso representado por la URL requerida por el usuario. Esta representación permite la integración de las SunSPOT en el Triple Space, reemplazando la capa que interactua con el repositorio semántico de cualquier nodo por la capa que convierte las primitivas del TS en peticiones HTTP al gateway. Las primitivas implementadas en esta capa son *read/take*, *write* y *query*.

La primitiva *read* posee dos implementaciones diferentes. La primera de ellas devuelve el grafo identificado por una URL solicitada al gateway. Por ejemplo, para obtener el grafo que describe la temperatura de una SunSPOT, la primitiva *read* es mapeada a la siguiente petición HTTP GET:

La segunda implementación devuelve la primera coincidencia del grafo que contiene al menos una tripleta que coincide con la plantilla proporcionada por el usuario. Para realizar esta operación, la capa de acceso solicita la siguiente URL pasando la plantilla proporcionada como parámetro:

La primitiva *write* parsea el contenido de las tripletas enviadas por el usuario para extraer los valores necesarios para realizar la siguiente petición HTTP POST al gateway:

La primitiva *query* obtiene todas las tripletas que corresponden con una determinada plantilla en un determinado espacio. Para ello, se realiza la siguiente petición HTTP GET:

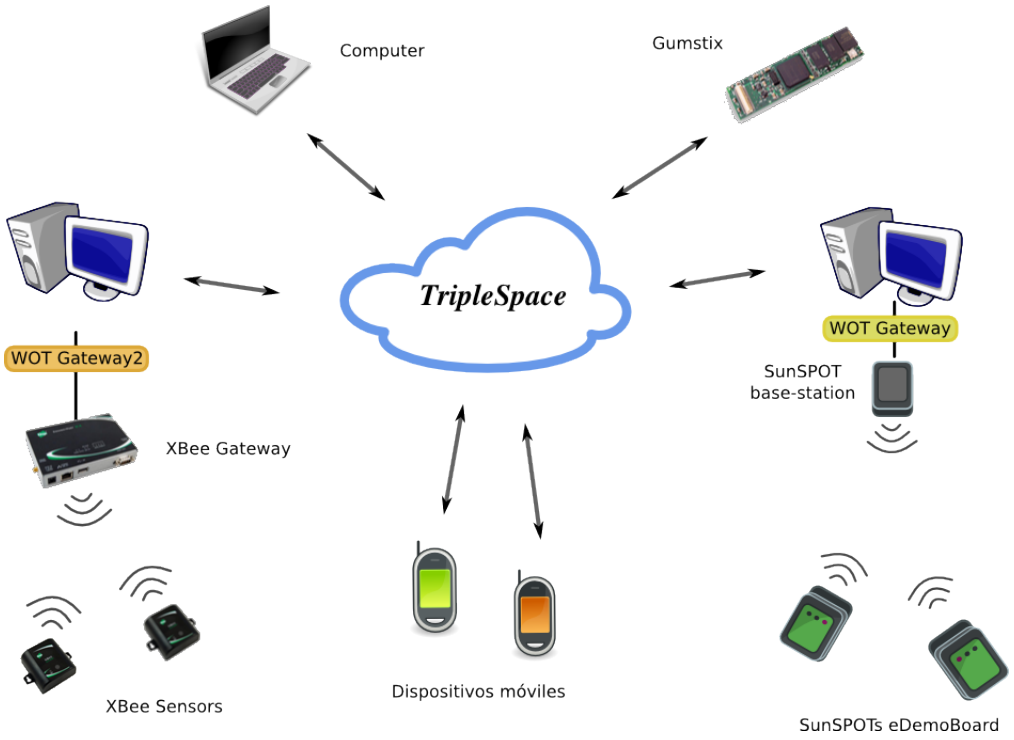


Figura 3.8.: Esquema de la arquitectura propuesto.

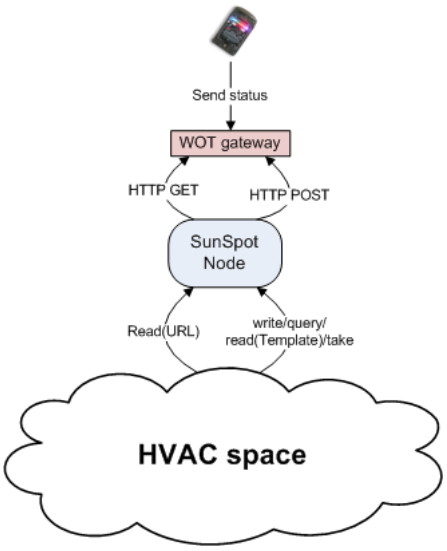


Figura 3.9.: Message exchanging between the nodes beyond the space, the WOT Gateway and a SunSPOT.

3.1.4 Diagrama de clases

En este epígrafe se tratará de aportar una visión más detallada del diseño mediante sendos diagramas de clases tanto en nodos de dispositivos móviles como en nodos que se ejecuten en computadoras normales o Gumstix.

Para estos últimos, tal y como se ha descrito anteriormente, se ha utilizado de base la versión 1.3 de la librería tsc++ [450], sobre la que se han realizado numerosos ajustes. El más importante de ellos, es que respetando su forma original, se le ha añadido la capacidad de comunicarse mediante pipes de Jxta, para poder comunicarse con los nodos en dispositivos móviles. Además, se ha añadido la primitiva queryMultiple que no existía en tsc++.

Todos estos cambios se han realizado de forma que el código original quedase lo menos afectado posible, a fin de poder integrar fácilmente los cambios que se vayan produciendo en la librería original.

3.1.4.1 tscSE

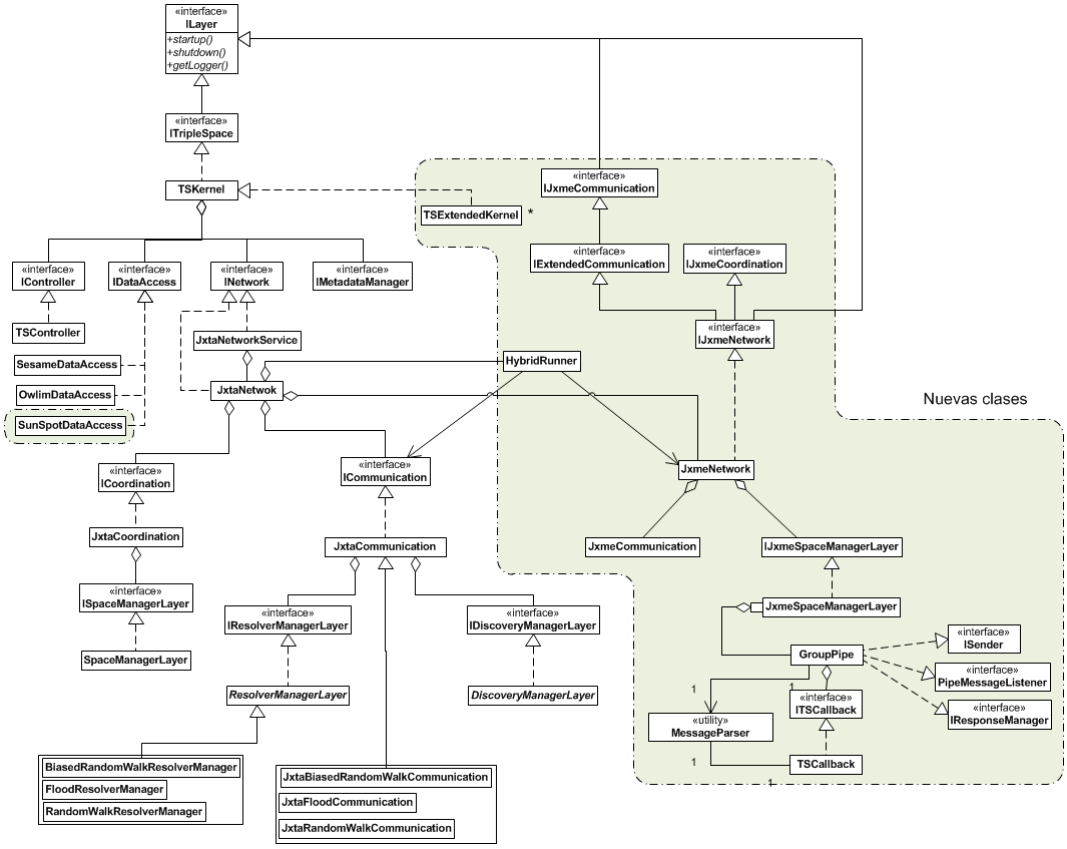


Figura 3.10.: Diagrama de clases de la tscSE.

TscSE es el nombre que le hemos dado a la parte del middleware que tiene sus origenes en tsc++. En el diagrama se muestran todas las clases añadidas para dar soporte a la comunicación con móviles en la parte derecha sombreada. Así, se puede apreciar el limitado impacto que tiene en las clases originales de tsc++ y que, como se comentaba anteriormente, es uno de los objetivos primordiales.

Para instanciar un kernel de tsc++, basta con crear una clase *TSKernel*, que esencialmente estará compuesta por una clase de acceso a datos *IDataAccess* (que tratará con los repositorios semánticos Sesame u Owlim) y otra que manejará los aspectos relativos a la red (*INetwork*).

A ese nivel aparece la primera particularidad: el API extendida de la que se hablaba antes, tendrá las primitivas del API básica, más otras primitivas, y se podrá hacer uso de ella instanciando

TSExtendedKernel, que tendrá una estructura interna similar a *TSKernel* y que ha sido obviada por claridad.

También se puede apreciar que todo *INetwork* se compone de un elementos de coordinación (*ICoordination*) y otro de comunicación (*ICommunication*). La primera aglutina la respuesta a las primitivas que realizan operaciones sobre los espacios (*create*, *join* y *leave*) y la segunda el resto de primitivas de Triple Space.

La comunicación entre nodos tsc++ se realiza mediante el intercambio de advertisements personalizados. Este tipo de comunicación no está plenamente soportada por la versión para Java ME de Jxta, por lo que se ha tenido que usar un enfoque híbrido en el que con otros nodos tsc++ se utilice ese tipo de comunicación, y con los nodos móviles se comunique mediante Pipes.

Es por ello que se ha creado la clase denominada *HibridRunner* y toda la estructura de clases de la zona derecha del diagrama. Así, en *JxtaNetwork* se llamará a *HibridRunner* que será la clase encargada de llamar en sendos Threads a las primitivas mediante el método original de tsc++ (haciendo uso de *ICoordination* e *ICommunication*) y mediante pipes (haciendo uso de *IJxmeCommunication* e *IJxmeSpaceManagerLayer*).

JxtaCommunication, que implementa *ICommunication*, tiene otras clases entre las que cabe destacar *IResolverManagerLayer* e *IDiscoveryManagerLayer*. En la primera se reciben llamadas a las primitivas *query*, *read* o *take* de otros nodos, mientras que en el segundo se tratan las relacionadas con las suscripciones (*advertisement*, *subscribe*).

El enfoque seguido en la otra rama (*JxmeCommunication*), es pedir a la capa de coordinación (*IJxmeSpaceManagerLayer*) el pipe asociado a un espacio y escribir sobre él mensajes. Así, *JxmeSpaceManagerLayer* se encarga de crear, eliminar y devolver referencias a pipes, cuyos métodos de escritura y lectura se implementan en la clase *GroupPipe*.

3.1.4.2 tscME

A continuación se detalla el esquema de clases diseñado para la versión del middleware que se desplegará en dispositivos móviles, a la que hemos llamado **tscME** y para la que nos hemos inspirado en la estructura utilizada en tsc++.

Como se puede apreciar en un primer vistazo, al igual que en tsc++, la interfaz *ITripleSpace* define las primitivas del TripleSpace definidas con anterioridad. *TSKernel* implementa dicha interfaz, y realiza las primitivas delegando en *IDataAccess* y *INetwork*, que a su vez delega en *ICoordination* e *ICommunication*.

Como particularidad, cabe destacar que *IDataAccess* en este caso tiene dos estrategias básicas de tratamiento de datos: en memoria o usando el RecordStore de Java ME CLDC. El almacenamiento de tripletas en ficheros también se contempló, pero resultaba excesivamente lenta y al desplegarla en dispositivos móviles requería firmar la aplicación (dado que de otro modo saltaban mensajes emergentes constantemente y no permitían el normal funcionamiento de la aplicación que usase la librería tscME).

JxmeCoordination usa una clase *JxmePeerBase*, que le proporciona la configuración básica necesaria para crear un nodo en Jxme. Además, tiene dos mapas en los que guarda espacios creados y unidos.

Los espacios se representan mediante *SpaceManager*, que tiene *JxmeSpace* quien escribe y lee en el pipe asociado a su espacio. La clase de mayor nivel de abstracción para el envío de primitivas a través de red que hace uso de *JxmeSpace* es *OutcomingManager*. La clase encargada de dar respuesta a las primitivas recibidas es *ResponseManager*. *ResponseManager* tiene una *IncomingList*, en las que asocia las posibles respuestas a los mensajes enviados con anterioridad.

3.1.4.3 tscLite

Para posibilitar la integración de los dispositivos SunSPOT, se ha realizado una nueva implementación de *IDataAccess*, denominada *SunSpotDataAcess*, encargada de traducir las primitivas de tscSE en llamadas HTTP, tal y como se ha explicado en el apartado 3.1.3.1.

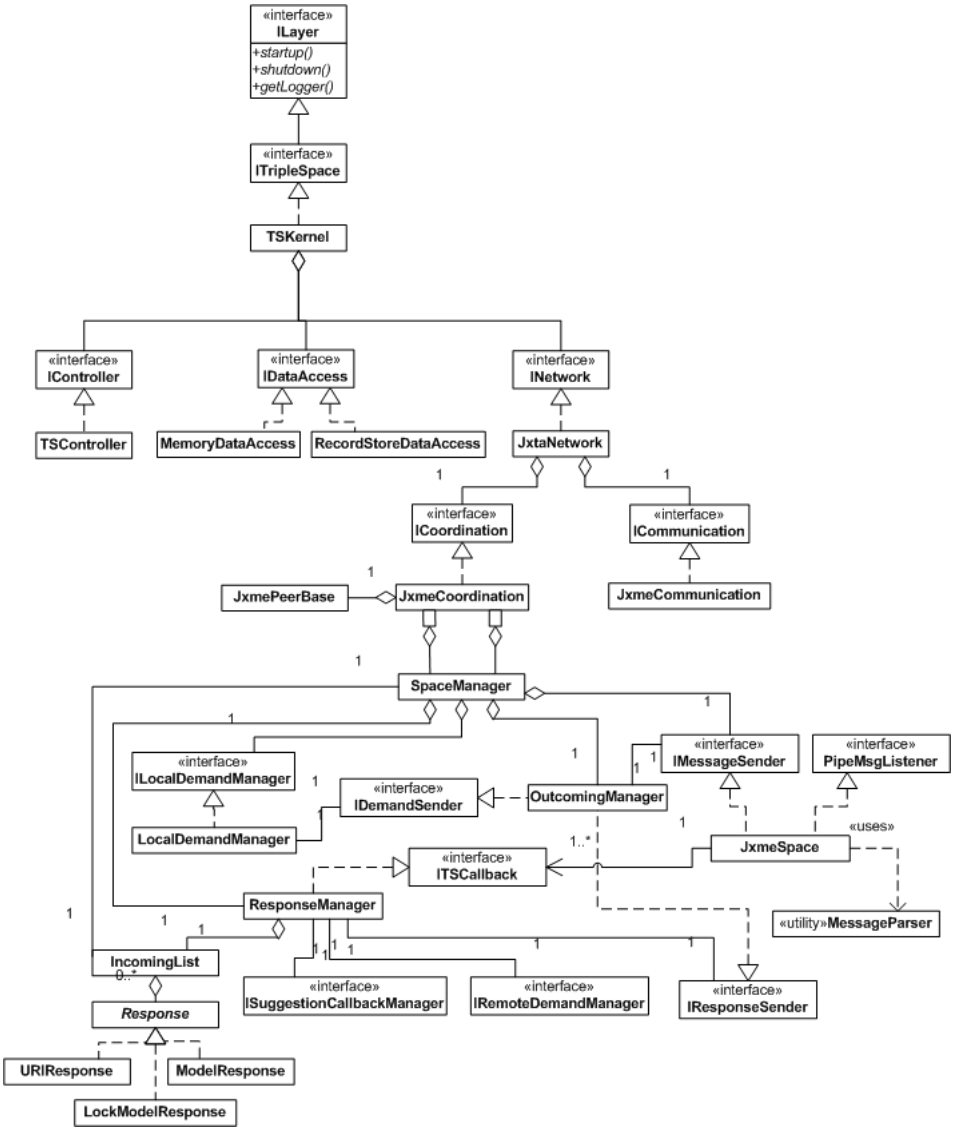


Figura 3.11.: Diagrama de clases de la librería tscME simplificado.

3.1.4.4 Escritura remota

Como ya se presentó en el apartado 3.1.2.1, a pesar de que el *negative broadcasting* se ajusta a la perfección a escenarios en los que los nodos se conectan y desconectan a la red local compartiendo su información, este enfoque presenta problemas cuando un nodo modifica la información de un actuador controlado por otro nodo diferente.

La primera aproximación para solucionar este problema fue la utilización de servicios sobre el Triple Space. La problemática de este enfoque fue la misma que la que surge entre los servicios web de tipo WS*- y de tipo REST [296]: demasiada complejidad. Como se describe en [679], el paradigma de Triple Space está básicamente orientado a recursos; al utilizar todas las primitivas recursos semánticos (tripletas), es lógico buscar una solución acorde con el paradigma, en el que las tripletas se utilicen como base. De esta manera, para simplificar la API para el desarrollador, se ha modificado la implementación de la primitiva *write* de manera que no tenga que preocuparse del tipo de escritura que debe realizar.

Para cumplir dicho objetivo, se ha implementado la primitiva *demand*. Esta primitiva permite a cada nodo mostrar su responsabilidad acerca de una determinada porción de conocimiento, por lo que se ha modificado la primitiva *write* para que envíe el conocimiento a los demás nodos siempre y cuando conozca en quién recae la responsabilidad sobre ese conocimiento.

La primitiva *demand* consta de un ciclo de vida máximo en el que deberá atender las peticiones de otros nodos, antes de eliminarlas para evitar la acumulación de éstas. Por otra parte, los nodos que se unan al espacio interrogarán a los demás para conocer las responsabilidades de cada uno.

Finalmente, la primitiva *write*, después de su modificación, ha quedado con el siguiente comportamiento:

- Si el *nodo a* intenta escribir un conjunto de tripletas y se conoce que algún otro nodo es responsable del conocimiento que se está intentando escribir (algún nodo ha llamado a la primitiva *demand* con una plantilla que representa al menos a una de las tripletas que el *nodo a* está intentando escribir), se sugerirá a los demás nodos que el *nodo a* quiere modificar cierto conocimiento con esa serie de tripletas. Esto se realiza a través de una primitiva transparante al usuario llamada *suggest*.
  - Cada nodo que haya *demandado* una plantilla que coincida con alguna de las tripletas a escribir, llamará a un método *callback* que realizará las acciones pertinentes con las tripletas recibidas.
- Si ningún nodo ha reclamado la responsabilidad sobre dicho conocimiento, será escrito en el repositorio semántico local.

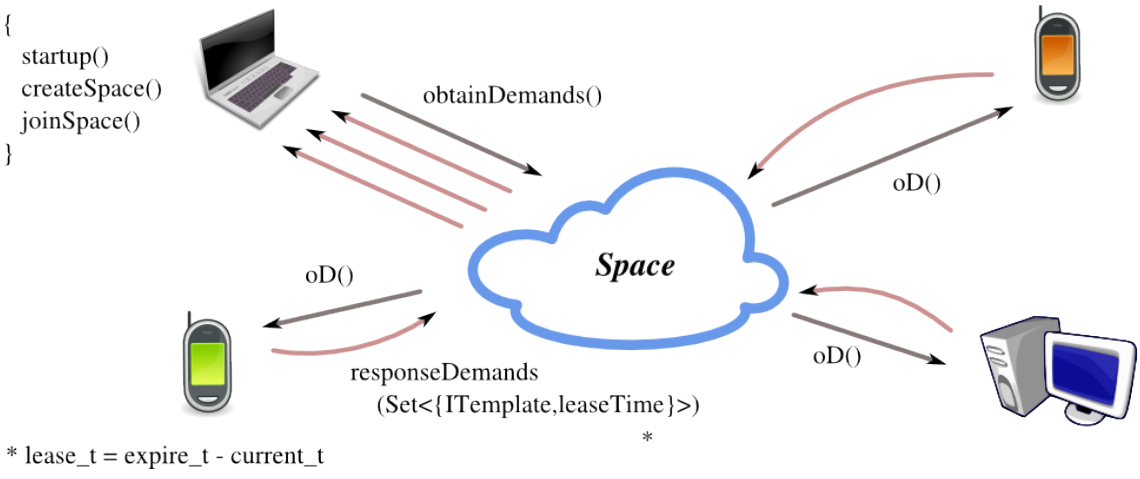


Figura 3.12.: A node which joins a space ask to the rest of the members about the demands they have received.

3.1.5 Ontología para ISMED

Esta sección describe brevemente la ontología desarrollada para ISMED que será la *lingua franca* que permita el entendimiento entre los objetivos dispares pero complementarios de los módulos de descubrimiento, composición, razonamiento y aprendizaje que conforman ISMED.

Para realizar esta ontología, se consultaron numerosas ontologías para realizarlo, entre las que cabe destacar: la de la tesis de Ramón Hervás [503] y [6].

En un primer nivel, encontraríamos el entorno (*Environment*), que estará compuesto por un conjunto de áreas (*Area*), que a su vez podrán ser edificios (*Building*), o alas (*Wing*) o plantas (*Floor*) del mismo. Un área puede a tener a su vez lugares (*Place*), que pueden ser cuartos (*Room*) o pasillos (*Corridor*).

Otro elemento importante, serían la entidades (*Entity*), que en nuestra ontología podrían ser los propios usuarios del sistema (*User*) o los dispositivos que forman la red (*Device*).

Las medidas (*Measure*), a su vez, estarán asociadas al dispositivo que las toma y podrían ser de tipo: temperatura (*TemperatureMeasure*), el encendido o apagado de un switch (*SwitchMeasure*) o de un led (*LEDMeasure*), ruido (*NoiseMeasure*), luz (*LightMeasure*) o humedad (*HumidityMeasure*).

Con la definición de ítems P2P (*P2PItem*), se ha pretendido aportar información relativa a la propia red P2P que se genera al usar el módulo de coordinación semántica. En ella hay espacios (*Spaces*) que son las subdivisiones lógicas con las que trabaja Triple Space, Rendezvous que es un elemento necesario en una arquitectura Jxta para que los mensajes se propaguen a través de la red o permitir descubrimiento de nuevos nodos en la red sin necesidad de usar multicast, y nodos (*Peer*). Los nodos pueden ser de tipo tsc++ (*tscppPeer*) en caso de que usen la versión modificada de esta librería o *tscME* en el caso de aquellos que usen la librería para móviles realizada en ISMED.

También existe la propiedad de recursos multimedia (*ResourceMultimedia*), que son clases con las que se podrá especificar algún metadato que sirva para representar una entidad y enriquecer la posible interfaz en la que se muestren dichas entidades. Estos datos podrán ser de tipo texto (*ResourceText*), imagen (*ResourceImage*), audio (*ResourceAudio*) y vídeo (*ResourceVideo*).

Finalmente, también se pueden apreciar otras clases que representan tipos más básicos:

- *Color*
- *Uncertainty.* Con la que se especificará el nivel de confianza se le otorga a una medida concreta.
- *DeviceStatus.* Con la que se expresa la disponibilidad de un dispositivo y otras propiedades relativas al mismo (disponibilidad, estado de carga de la batería, etc.).
- *PositionItem.* Es una clase que agrupa dos formas diferentes de expresar una posición: mediante ejes x, y y z (*Axis*) y mediante latitud y longitud (*Location*). La posición mediante ejes es especialmente útil para dispositivos como los acelerómetros, mientras que la localización sirve para dispositivos que tengan algún mecanismo de posicionamiento global integrado.

3.1.5.1 Jerarquía de la clase Device

Como se puede apreciar en la Ilustración 37, se ha diseñado una jerarquía en la que se especifican los tipos de dispositivos que formarán parte del proyecto ISMED y que se puede extender con facilidad.

En ella se puede apreciar una primera división en la que se distinguen los dispositivos complejos (*ComplexDevice*), que en esencia serán aquellos que están compuestos por dispositivos simples (*SimpleDevice*), que son los sensores y actuadores que nos permitirán capturar datos básicos del entorno y actuar de forma simple en él..

Entre los dispositivos complejos podemos ver otra subdivisión formada por los dispositivos autónomos (*AutonomousDevice*), que serán aquellos plenamente capaces de soportar el API extendida, y los dispositivos dependientes (*DependentDevice*), que serán aquellos que no soportan el API extendida por tener capacidad de computo más reducida.

Entre los dispositivos autónomos cabe destacar las computadoras (*Computer*) y los dispositivos embebidos (*EmbeddedDevice*), con los que se ha querido hacer referencia a aquellos dispositivos

Figura 3.13.: Esquema de la ontología de ISMED.

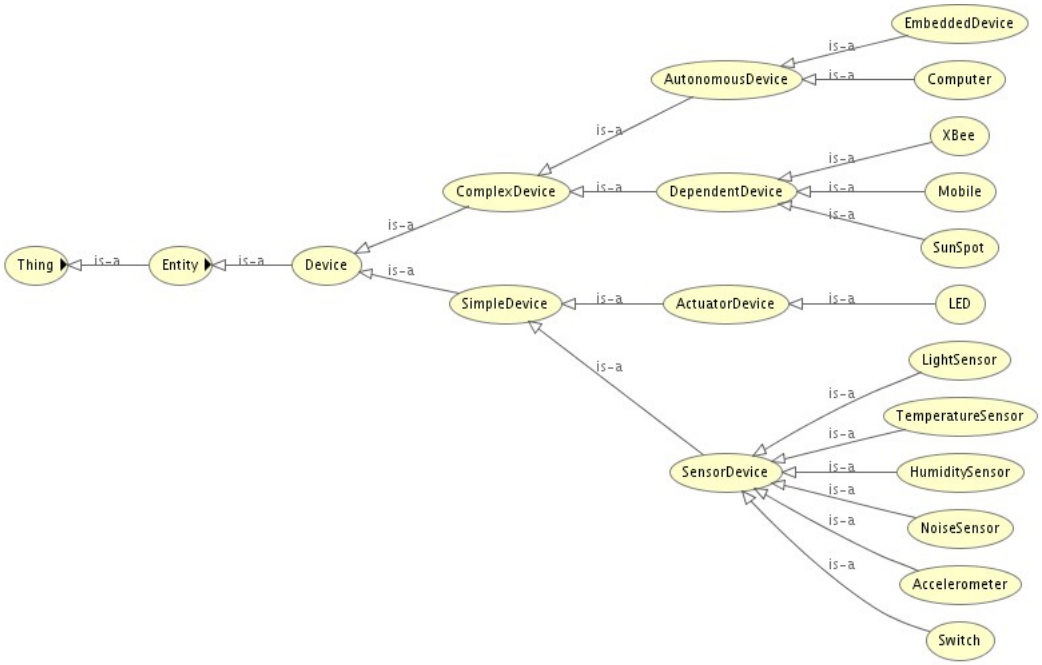


Figura 3.14.: Esquema de la ontología de ISMED.

como los Gumstix, que pese a no ser una computadora al uso, si que tienen capacidades equiparables a estas.

3.2 Aprendizaje

El aprendizaje de patrones de comportamiento a partir de la información recolectada por los sensores, por la complejidad que tiene, demanda una arquitectura dividida en varias capas que al mismo tiempo tendrán varios módulos.

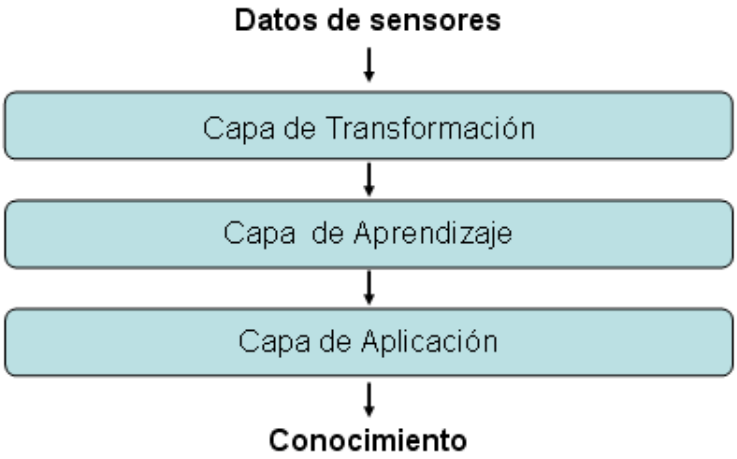


Figura 3.15.: Arquitectura global del aprendizaje

3.2.1 Capa de transformación

Una vez que los datos de los sensores hayan sido recogidos, la primera tarea a realizar es inferir acciones relevantes a partir de dichos datos. A veces, la información proveniente de los sensores será relevante en sí, por ejemplo:

desde

2009-10-20T08:15:57,InterruptorLuzOficina,on

inferimos que:

2009-10-20T08:15:57, LuzOficina,on

En este caso, la acción en sí es relevante debido a que podemos inferir la acción que el usuario ha llevado a cabo. Hay otras acciones que necesitan ser inferidos a partir un conjunto de acciones recogidos por los sensores. Por ejemplo la acción de "Entrar a la oficina" no puede ser inferida a partir de la activación de un solo sensor. Para ello, utilizamos un conjunto de acciones simples. Así,

desde

2009-10-20T08:15:54,MovimientoPasillo,on

2009-10-20T08:15:55,RFID Oficina,on

2009-10-20T08:15:54,MovimientoOficina,on

inferimos que:

2009-10-20T08:15:57, EntrarOficina,on

La manera más básica de inferencia de dichas acciones es mediante plantillas que definen qué acciones tienen que combinarse para inferir acciones relevantes. Además de la acciones, se especifican las constraints a cumplir. Estos constraints son relativos a la orden de las acciones como a la duración de las mismas. Por ejemplo, la plantilla para la acción de "Entrar Oficina" es:

Acciones:

Movimiento Pasillo

RFID Oficina,

Movimiento Oficina

Constraints:

Orden,

Movimiento Pasillo $<$RFID Oficina $<$Movimiento Oficina

Tiempo,

$T_{MovimientoOficina} - T_{MovimientoPasillo} < 3$ seg.

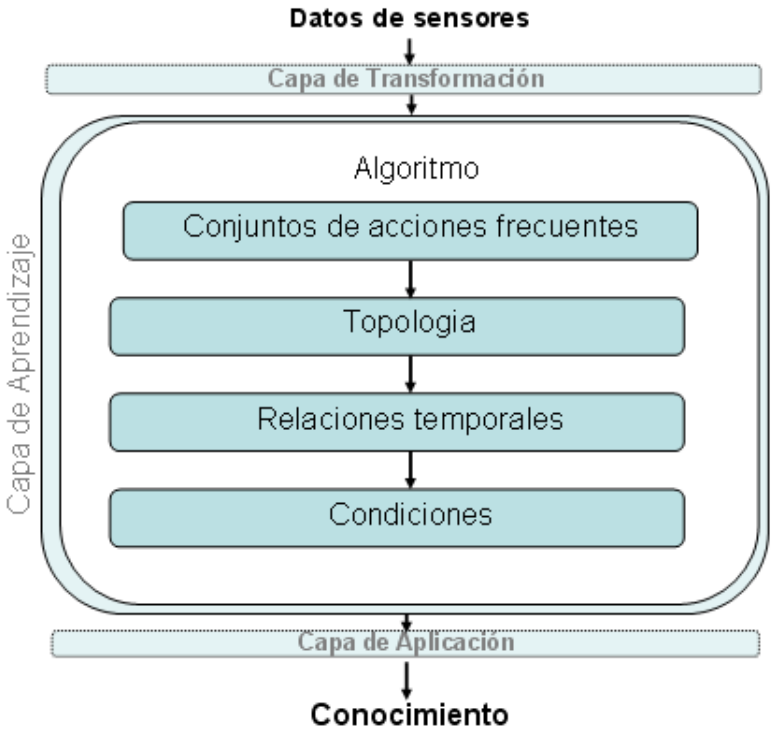
Está claro que la definición de las plantillas está determinada por el conjunto de acciones que queramos identificar así como por los sensores que hay en cada entorno.

3.2.2 Capa de aprendizaje

El objetivo de esta capa es la de extraer los patrones de comportamiento más frecuentes a partir de los datos provenientes de la capa de transformación.

Para ello, es necesario definir qué tipo de patrones se quieren descubrir. En este sentido, el objetivo propuesto en este desarrollo ha sido el de descubrir aquellos patrones que mejor (o más frecuentemente) definan el comportamiento de los usuarios. Además de decidir qué tipo de patrones se quieren descubrir se ha definido cómo se van a representar, ya que la forma en que se quieran representar los patrones influye en el proceso de aprendizaje. Así, se ha decidido representar los patrones mediante secuencias de acciones.

El núcleo de la capa de aprendizaje es el algoritmo que se ha desarrollado. Dicho algoritmo tiene varios módulos que secuencialmente ejecutados descubren los patrones frecuentes. La figura 3.2.2 representa dichos módulos:



3.2.2.1 Descubrimiento de los conjuntos de acciones frecuentes

Una vez que los datos han sido recolectados y las acciones de los usuarios hayan sido definidas, el primer paso es identificar qué conjuntos de acciones son frecuentes en el comportamiento del usuario. Es decir, el objetivo de este primer módulo es la de descubrir qué acciones ocurren frecuentemente juntos.

Para ello, se ha utilizado el algoritmo A priori [44], utilizado para el descubrimiento de reglas de asociación. Este algoritmo identifica aquellos conjuntos de acciones que ocurren más veces que el threshold definido. En este caso, dependiendo de las necesidades de cada entorno, dicho threshold puede variar para encontrar los conjuntos de acciones que se deseen.

3.2.2.2 Descubrimiento de la topología

El conjunto de acciones descubierto en el primer módulo define las acciones que el usuario lleva a cabo frecuentemente, pero no define el orden en que los lleva a cabo. Parece lógico que si esas acciones implican alguna actividad que dichas acciones se lleven a cabo en un orden concreto, siempre teniendo en cuenta la incertidumbre que produce trabajar con el comportamiento de los seres humanos. Para ello, identificamos las secuencias donde se dan dichas acciones y recogemos el orden de las mismas. Basándose en algoritmos de Workflow Mining [821] se plantea la topología base. Aunque los algoritmos de workflow mining son eficientes, no tienen en cuenta ciertos aspectos que son importantes en los entornos inteligentes.

Puede haber un sub-conjunto de acciones que no tengan un orden definido o que el usuario los haga de forma no ordenada. Un ejemplo sencillo y práctico de ello puede ser que al prepara el desayuno a veces cogemos primero la leche y luego el zumo o viceversa. Para estos casos se ha decidido crear grupos de acciones no-ordenadas. La creación de dichos grupos es definida por varios parámetros que estableces que relaciones bidireccionales serán considerados como grupos no-ordenados.

La repetición de las acciones es otro de los aspectos a tener en cuenta. Algunas acciones que son frecuentes se pueden dar más de una vez en la secuencia. Eso implica el tener que decidir si las diferentes instancias de dichas acciones serán consideradas como diferentes o iguales.

3.2.2.3 Descubrimiento de las relaciones temporales

En el módulo de descubrimiento de la topología ya se le da una dimensión temporal a la secuencia, explicitando qué acción precede a la siguiente acción. Pero esta relación es sólo cualitativa, es decir, definida solamente mediante relaciones temporales de Allen. En un entorno inteligente, por ejemplo para automatizar la siguiente acción, es interesante definir de la forma más precisa posible las relaciones temporales. Así, este módulo trata de descubrir las relaciones temporales más precisas posibles.

Las relaciones temporales se obtienen agrupando las relaciones temporales particulares. Para la agrupación de los valores particulares se utiliza la siguiente fórmula:

$$[min, max] = \bar{x} \pm (\bar{x} * tolerance) donde \bar{x} = \frac{\sum_{i=1}^{n} a_i}{n} \tag{3.1}$$

Donde: *tolerance* = desviación tolerada de $\bar{x}$ (%); a_i = la distancia temporal de un elemento; y n = número de elementos.

Una vez llevado a cabo el proceso de agrupación, si existe algún grupo que agrupe más instancias que las demandadas, la media de ese grupo será considerada como una relación temporal válida para definir cuantitativamente dicha relación.

3.2.2.4 Descubrimiento de las condiciones

Las relaciones entre dos acciones difícilmente definirán perfectamente todas las instancias de dichas acciones, por lo que hay que definir en qué casos lo definen bien y en cuáles no. Por ello, es necesario este último módulo de descubrimiento de condiciones.

Para ello se crear dos tablas, uno para aquellos casos que son cubiertos por la relación definida, y otro para aquellos que no son cubiertos. Una vez creadas esas dos tablas se utilizan técnicas de clasificación para determinar qué parámetro define mejor las diferencias entre esas dos tablas.

3.2.3 Capa de aplicación

Una vez que los patrones han sido aprendidos, se pueden utilizar para diversas aplicaciones. En este proyecto la aplicación que se ha propuesto es la de entender el comportamiento del usuario. Aún así, una de las aplicaciones más atractivas es la de automatizar la activación/desactivación de los dispositivos implicados.

3.3 Composición de servicios

Esta sección tiene como base la descripción del diseño del módulo de composición de servicios que tienen como objetivo determinar que combinaciones de funcionalidades ofrecidas por dispositivos del entorno satisfacen la tarea solicitada por el usuario. Para ello se ha decidido emplear un enfoque basado en conversaciones en el que tanto la petición de servicio así como los servicios ofrecidos por los dispositivos están descritas mediante conversaciones que describen el flujo informacional y el contextual de los servicios, siendo este enfoque empleado por primera vez en su aplicación para servicios descritos en base a efectos y condiciones. De esta manera se define que la *Conversation* asociada a un *CompositeCapability* está representada mediante tres elementos principales (ver Ilustración 40): el *Automaton* que describe el flujo de invocación de las *Capability* y/o *Method*[1], el *DataFlow* que determina el flujo de información entre los *Method* (relación entre los *ReturnValue* y los *Parameter*) y por último el *ContextFlow* que define el flujo de la relación de los *Effect* y las Condition de las *Capability*, así Conversation=<Automaton,DataFlow,ContextFlow>. Así se propone un proceso de composición que integra estos tres aspectos, los cuales son evaluados con el fin de determinar que la composiciones creadas satisfacen las restricciones definidas en la . Como se verá más adelante no será necesario evaluar los tres elementos en todos los tipos de composición planteados, siendo únicamente necesario en los casos en los que debido a la posibilidad de existencia de interdependencias entre las *Capability* de la el orden de las *Capability* de la misma y la resultante

[1] De aquí en adelante con el fin de simplificar las expresiones se va a emplear la denominación Capability para representar la expresión Capability y/o Method.

sean diferentes (donde $1 \leq res \leq RES$, siendo RES el número total de elementos Conversation calculados en el proceso de composición y que emparejan con , siendo $ResultConversationList = \{ < Conversationbase_{res}, MatchResultbase_{res} > \}$ $base_{res} = 1, \ldots, RES$) y el valor que representa la distancia entre y .

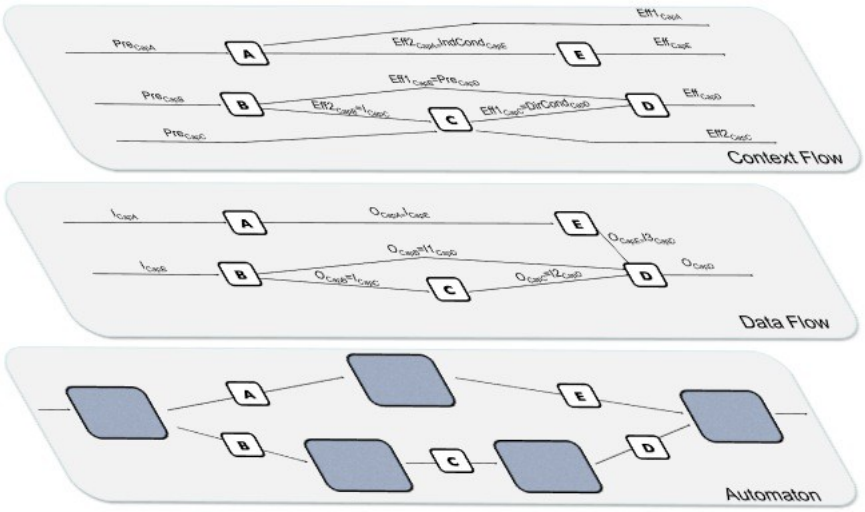


Figura 3.16.: *Conversation* representado mediante *Automaton*, *DataFlow* y *ContextFlow*.

3.3.1 Propiedades del mecanismo de emparejamiento basado en Conversation

El elemento principal a emparejar será el Automaton, en el cual el mecanismo de composición basado en Automaton tomará como base la solicitud ($Automaton_{Req}$) y lo emparejará respecto al conjunto de Automaton de las Capability de los Service que se encuentran en Registry). Para ello el sistema de composición empleará diferentes métodos de emparejamiento relativos a la granularidad del proceso de composición soportado, ofreciendo así un ámplio espectro de configuración para así ofrecer diferentes rangos de rendimiento y resultado, que serán empleados para seleccionar que tipo de composición es más adecuado para que entorno. Así, el $Automaton_{Req}$ representado en la Ilustración 3.17 podrá ser compuesto de varias maneras como será descrito más adelante.

Previa a la definición descripción de los diferentes métodos de composición soportados por el sistema a continuación se describen las premisas básicas que tienen que ofrecer las creadas tras el proceso de composición.

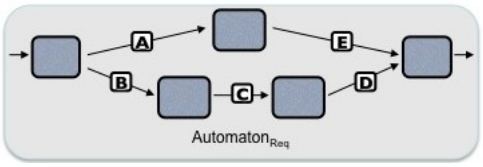


Figura 3.17.: Automatón requerido.

- Todas y cada una de las $Conversation_{res}$ generadas tras el proceso de composición tienen que emparejar correctamente con la requerida ($Conversation_{Req}$). Las $Conversation_{res}$ no solo tienen que emparejar a nivel de Automaton, DataFlow y ContextFlow sino que también tienen que emparejar en términos de descripción global, es decir, tienen que emparejar tanto la signatura como la especificación global de la $Conversation_{Req}$ y las $Conversation_{res}$ (ver Ecuación 3.2). Para tal fin se empleará el mecanismo $\boldsymbol{resultChecking(Conversation_{ada}, Conversation_{toCheck})}$, donde $Conversation_{toCheck}$ representa a la *Conversation* que hay

que validar, pasando ésta a ser una $Conversation_{res}$ en caso de que sea válida respecto a $Conversation_{ada}$, siendo $Conversation_{ada}$ una de las Conversation generadas en el proceso de adaptación de la $Conversation_{Req}$:

$$\exists Conversation \rightarrow \exists AutomatonMatch, \exists ContextFlow, \exists DataFlowy \\ resultChecking(Conversation, Conversation = true) \tag{3.2}$$

- Todas y cada una de los Automatonreg (donde $1 \leq reg \leq REG$, siendo REG el número de elementos $Conversationbase_{reg}$ que se encuentran en el *Registry*) que formen parte del Automaton_{res} resultante deben de ser ejecutados en toda su extensión. Por ejemplo, en caso de que el $Automatonbase_{Req}$ esté compuesto por tres $Automatonbase_{reg}$ éstos tienen que ser representados en toda sus extensión, es decir, tienen que comenzar en un *InitialState* ($State0base_{reg}$) y tienen que finalizar en un *FinalState* ($Fbase_{reg}$) válido. De esta manera se asegura que todos los $Automatonbase_{reg}$ que forman parte de $Automatonbase_{res}$ podrán ser invocados correctamente y todos ellos habrán llegado al *FinalState* una vez finalizada la ejecución del $Automatonbase_{res}$, evitando así que queden elementos $Automatonbase_{reg}$ en espera (ver Ecuación 3.3).

$$inv(Conversation) \rightarrow (sistate(Automatonres) = F \\ \rightarrow \forall Automaton \exists Automatonstate(Automaton) = F) \tag{3.3}$$

En toda su extensión, el enfoque de composición presentado empleará diversos tipos de esquemas de composición, los cuales ofrecen diferentes resultados para la Ilustración 3.17. Inicialmente esta clasificación de esquemas se basa en la diferenciación del tipo de composición empleado, donde $CompositionType = \{\text{compType} : \exists \{CapabilitySelection, ConversationSelection\}\}$, tomando como base la granularidad empleada para emparejar la solicitud respecto a lo ofrecido en el repositorio (ver Ilustración 3.18):

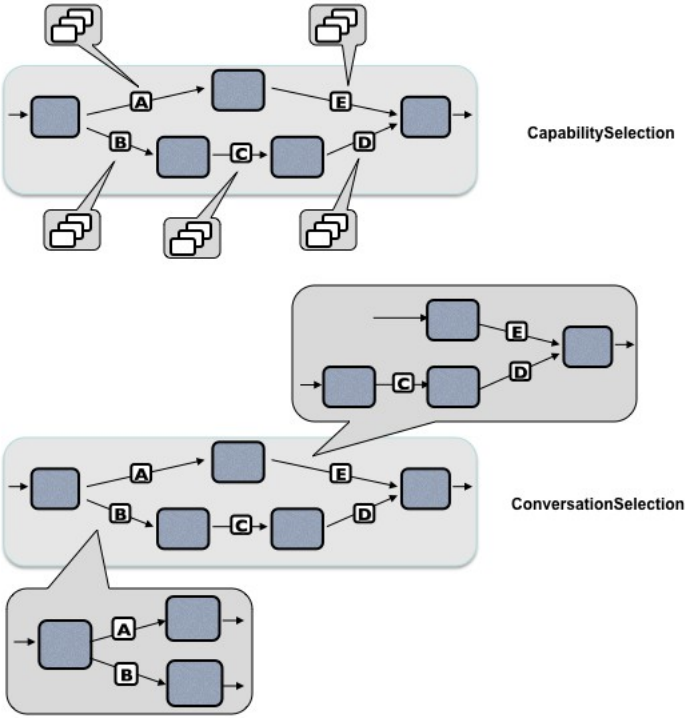


Figura 3.18.: Tipos de composición soportada.

1. **Selección de operaciones orientada a conversaciones (*CapabilitySelection*)**: Mediante esta técnica el $Automaton_{res}$ es calculado mediante la integración de los $Automaton_{reg}$ asociados a los del repositorio. Este método es uno de los más empleados para la composición de servicios en entornos de computación ubicua. Este método trata de emparejar una a una cada una de las *SimpleCapability* solicitadas en la $Conversation_{Req}$, con al menos una de las $SimpleCapability_{reg}$ que forman parte del repositorio. Así se obtiene una $Conversation_{res}$ construida exclusivamente en base a $Conversation_{reg}$ de tipo más simple, las cuales son aquellas *Conversation* que representan las *SimpleCapability*.
2. **Integración de Conversaciones orientada a Conversaciones (*ConversationSelection*)**: Mediante esta técnica el $Automaton_{res}$ es calculado mediante la combinación de los $Automaton_{reg}$ representados mediante elementos *Conversation*, obteniendo así una $Conversation_{res}$ construida en base a una o más $Conversation_{reg}$ que pueden ser tanto de tipo simple (*Conversation* que describe la invocación de un único *Capability*) o complejo (Conversation con construciones avanzadas, como: bucles, condicionales, etc.). Este tipo de composición representa un superconjunto del anterior: *CapabilitySelection* $\subseteq$ *ConversationSelection*.

3.3.1.1 Integración de Conversaciones orientada a Conversaciones

La **Selección de operaciones orientada a conversaciones** no ofrece flexibilidad para la composición ya que solo es posible realizarlo de una única manera, sin embargo, la denominada *Integración de Conversaciones orientada a Conversaciones* ofrece cuatro dimensiones de configuración, las cuales permiten representar la flexibilidad del mecanismo de composición, consiguiendo de esta manera variantes de composición diferentes en términos de rendimiento y resultados ofrecidos (ver Ilustración 3.19).

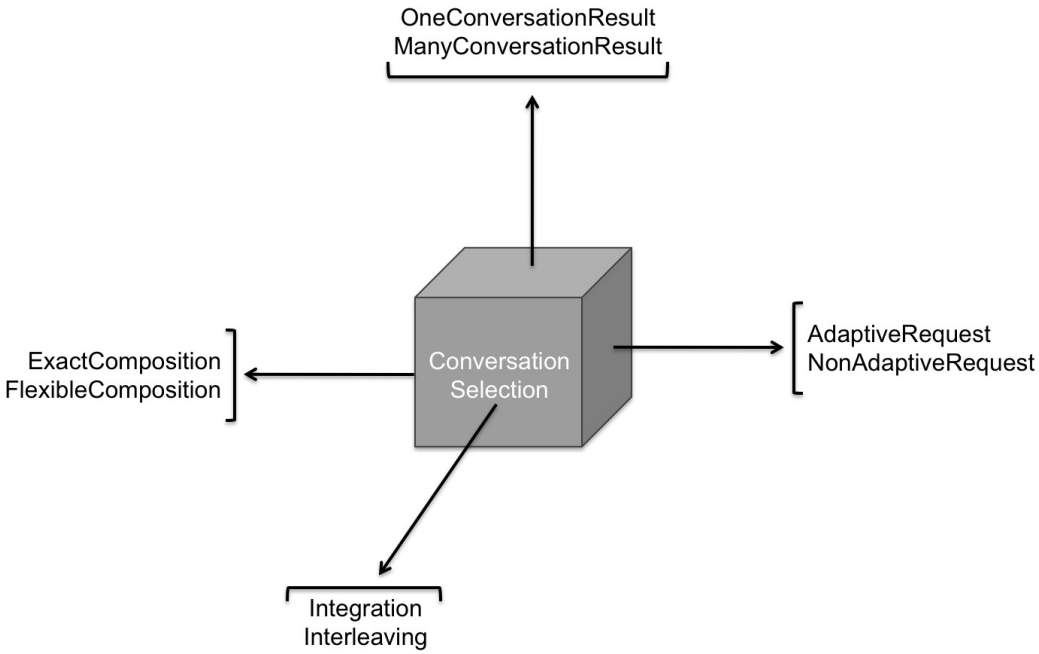


Figura 3.19.: Dimensiones para configurar las variantes de la composición basada en conversaciones.

Como se verá a continuación estas cuatro dimensiones afectan a todos los aspectos de la composición, comenzando desde la flexibilidad de la petición (*AdaptivityMode*), técnica empleada (*CompositionTechnique*) y grado de relajación (*RelaxationMode*) del mecanismo de composición, hasta llegar al tipo de resultado final deseado (*FinalCompositionType*), así *ConversationSelection*=<*AdaptivityMode*,*CompositionTechnique*,*RelaxationMode*,*FinalCompositionType*>. A continuación se ofrece una breve descripción de los diferentes parámetros que componen el elemento ConversationSelection además de ejemplos que facilitan su compresión:

a) **Adaptabilidad de la Conversation solicitada (*AdaptivityMode*):** Pueden existir casos en los que el $Automaton_{Req}$ pueda ser modificado con el fin de conseguir un emparejamiento más completo. Este tipo de caso se da cuando el orden de las *Capability* que forman parte del $Automaton_{Req}$ puedan ser modificadas sin que ello afecte al $ContextFlow_{Req}$ y al $DataFlow_{Req}$ del mismo. De esta manera al variar las posibles combinaciones del $Automaton_{Req}$ se amplía el espectro de posibles $Automaton_{res}$ que emparejan con lo solicitado. Por ejemplo, en el caso en el que se solicite un $Automaton_{Req}$ que incremente la intensidad de la luz y que después ponga la música de un estilo en concreto, en caso de que estas dos funcionalidades no tengan relación entre ellas (no exista $DataFlow_{Req}$ o $ContextFlow_{Req}$ que las relacione directa o indirectamente), el sistema trataría de buscar a su vez también una combinación de $Automaton_{Req}$ (equivalente pero en el que el orden de las *Capability* de la misma es diferente, siendo el resultado final el mismo) en la que primero se ponga la música de un estilo en concreto y después se incremente la intensidad de la luz. De esta manera, existen dos variantes del sistema, donde *AdaptivityMode* = $\{adaptMode \mid adaptMode \in \{\text{NonAdaptiveRequest,AdaptiveRequest}\}\}$:

   - ***NonAdaptiveRequest:*** Esta opción se emplea cuando no es posible o no se desea adaptar la $Conversation_{Req}$.
   - ***AdaptiveRequest:*** En cambio en la denominada AdaptiveRequest la $Conversation_{Req}$ es analizada con el fin de determinar las posibles combinaciones equivalentes existentes, con el fin de ofrecer una mayor flexibilidad al mecanismo de emparejamiento. En este caso el elemento a modificar es el $Automaton_{Req}$ para así crear nuevas versiones de la misma en la que el orden de las Capability es alterado, siendo tanto el DataFlow como el ContextFlow equivalente, es decir, entre las mismas Capability que en la $Conversation_{Req}$ original. Este proceso dará lugar al elemento AdaptedReqConversationList donde $AdaptedReqConversationList = \{Conversation_{ada}\}_{ada=1,...,ADA}$ y ADA representa el número de variantes de $Conversation_{Req}$ que se han generado durante el proceso.

   Como se puede ver en la Ilustración 3.20 en la parte superior se muestra el *AdaptedReqConversationList* el cual está formado exclusivamente por $Automaton_{Req}$, en caso de que AdaptivityMode = NonAdaptiveRequest y en la parte inferior de la se puede apreciar como el elemento AdaptedReqConversationList está formado por las diferentes combinaciones de $Automaton_{Req}$ en el caso de que AdaptivityMode=AdaptiveRequest, donde se muestran las nuevas variantes $Automaton_{ada}$ calculadas, siempre y cuando se respeten el $DataFlow_{Req}$ y el $ContextFlow_{Req}$.

b) **Técnica de composición (*CompositionTechnique*):** La técnica de composición define cual será el proceso empleado para construir el equivalente $Automaton_{res}$ al $Automaton_{Req}$[2] (ver Ilustración 3.21), siendo éste de varios tipos, tal y como se describe a continuación, *CompositionTechnique*=$\{compTech \mid \text{compTech} \in \{\text{Integration,Interleaving}\}\}$:

   - ***Integration:*** En esta técnica el $Automaton_{res}$ es construido mediante la integración de varios $Automaton_{reg}$ para así conseguir elementos $Automaton_{res}$ válidos y completos. Para ello se concatenan varios $Automaton_{reg}$ (concatenando el estado final de un $Automaton_{reg}$ con el estado inicial de otro $Automaton_{reg}$) uno detrás de otro hasta conseguir un comportamiento equivalente al solicitado.
   - ***Interleaving:*** Esta técnica emplea un enfoque más flexible que el de *Integration* en el que la composición del $Automaton_{res}$ es llevado a cabo entrelazando varios $Automaton_{reg}$. Es decir, se emplea un mecanismo más flexible que el anterior en el que es posible entrelazar partes de $Automaton_{reg}$ para conseguir un $Automaton_{res}$ equivalente. En este caso el sistema que invoque la $Conversation_{res}$ tendrá que ser capaz de gestionar las múltiples conexiones que tendrá que mantener abiertas debido al entrelazado. Cabe destacar que $Integration \subseteq Interleaving$.

     En la Ilustración 3.21 se puede observar como en el tipo de composición *Integration* el servicio es compuesto integrando, es decir, integrando consecutivamente los tres $Automaton_{reg}$ para así conseguir un $Automaton_{res}$ equivalente al solicitado. Sin embargo, en el caso de

[2] Durante la presentación de las diferentes propiedades se considerará que *AdaptivityMode=NonAdaptiveRequest*, con el fin de facilitar la comprensión de la misma.

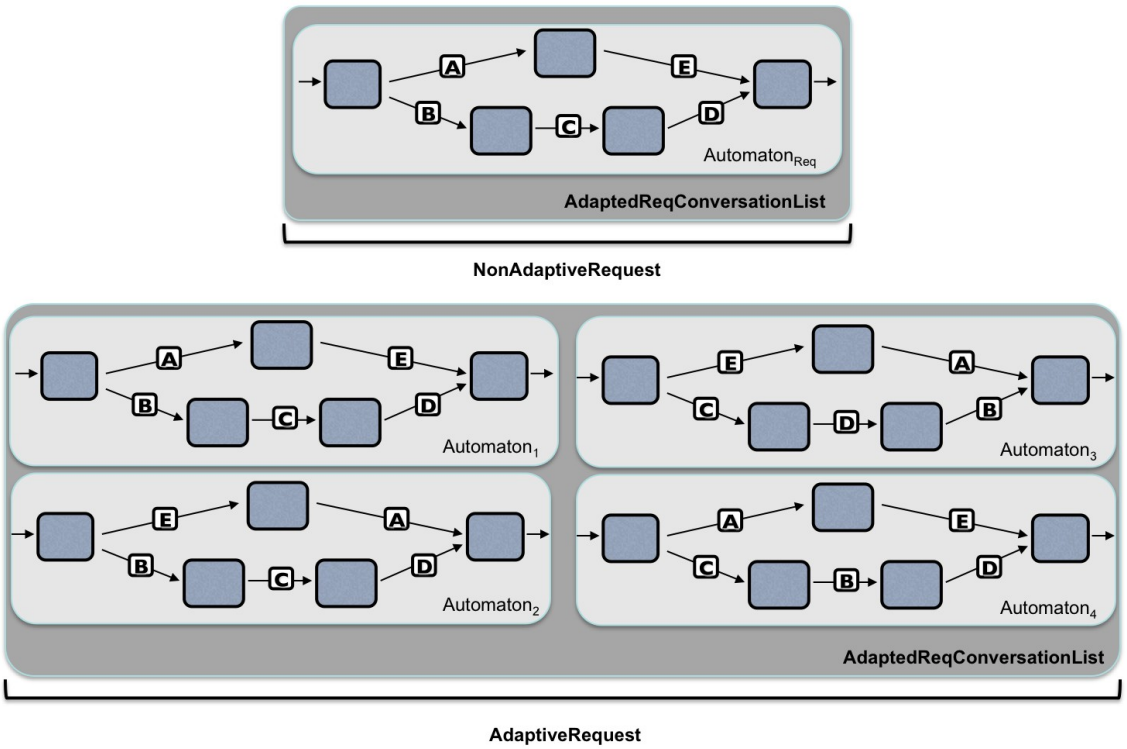


Figura 3.20.: Variantes de ConversationReq en base a la adaptabilidad de AdaptivityMode.

Interleaving pese a que a primera vista la concatenación de las $Automaton_{reg}$ no sigue el mismo orden de ejecución (concretamente en el caso de la rama de ejecución *B-C-D* solicitada la $Conversation_{res}$ dispone de la rama *B-D-C*) de las *Capability* de $Automaton_{Req}$ la composición es considerada válida. Esto es debido a que el mecanismo soporta el entrelazado entre conversaciones. Gracias a ello es posible comenzar con un *Automaton* y antes de que finalice éste, parar su ejecución para comenzar con otro *Automaton* para así volver al anterior una vez que haya finalizado el segundo y después seguir con la ejecución del primero hasta que finalice éste. Así en el caso de B-C-D, en el momento de ejecución de la composición, el mecanismo que invoca a las *Capability* tendrá que comenzar con el $Automaton_4$ y tras invocar la *Capability B* parará la ejecución de dicho Automaton y comenzará con la ejecución del $Automaton_6$ para así invocar la *Capability C* y posteriormente volver a continuar con la invocación de la *Capability D* y así llegar al estado final de cada uno de los $Automaton_{reg}$. Todo esto será posible solo si el motor de composición es capaz de soportar el entrelazado mediante la gestión de sesiones.

c) **Grado de relajación soportada en la composición (*RelaxationMode*):** Existen dos variantes del mecanismo en función del grado de relajación soportado para el resultado final (ver Ilustración 3.22), donde $RelaxationMode = \{relaxMode \mid relaxMode \in \{ExactComposition, FlexibleComposition\}\}$:

- ***ExactComposition:*** Esta técnica presenta un enfoque en el que el $Automaton_{res}$ tiene que ser estructuralmente equivalente al $Automaton_{Req}$. Es decir, tiene que tener el mismo número de elementos *Capability*.
- ***FlexibleComposition:*** Esta técnica presenta un enfoque más relajado que el anterior ya que permite que el $Automaton_{res}$ tenga en su conjunto más elementos *Capability* que las solicitadas en $Automaton_{Req}$, siempre y cuando éstas Capability extra no afecten al $DataFlow_{res}$, $ContextFlow_{res}$ ni a la solicitud global de la *CompositeCapability*, es decir, no produzca efectos indeseados. Cabe destacar que $ExactComposition \subseteq FlexibleComposition$.

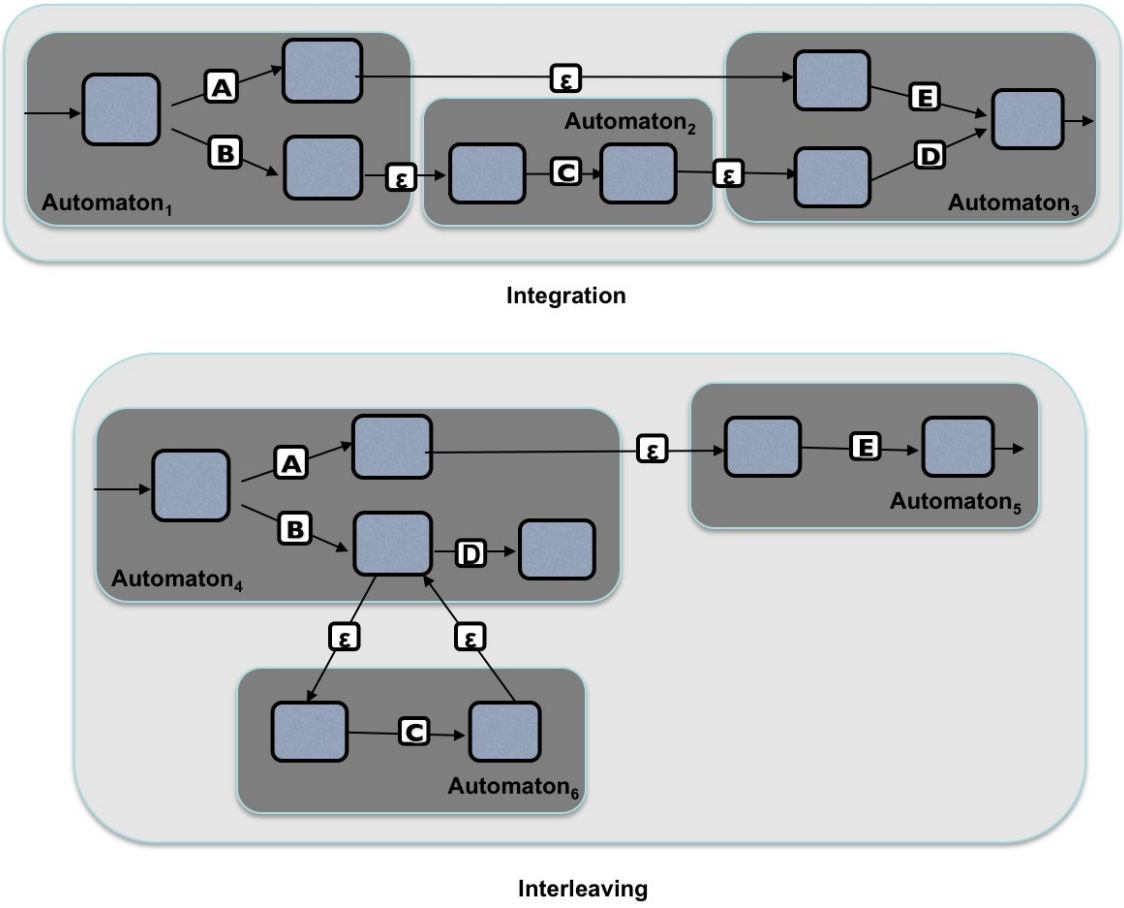


Figura 3.21.: Técnicas de composición soportadas.

Como se puede apreciar en la Ilustración 3.22, en el caso de *ExactComposition* la composición es exacta ya que dispone del mismo número de elementos Capability que la solicitud de la Ilustración 3.17. Sin embargo, en el caso de *RelaxedComposition* la composición resultante ofrece más elementos *Capability* que los solicitados pero como en global la composición resultante tiene el mismo comportamiento que el solicitado (no produce más *Effect* que los deseados ni necesita más elementos *Parameter* que los de), éste es considerado válido, previa comprobación mediante el proceso $\boldsymbol{resultChecking(Conversation_{ada}, Conversation_{toCheck})}$.

d) **Tipo de *Conversation* resultante (*FinalCompositionType*):** Dependiendo de como se construye la *Conversation* resultante existen dos variantes (ver Ilustración 3.23), así *FinalCompositionType*={*finalCompType* | finalCompType *in*{*OneConversationResult*, *ManyConversationResult*}}:

- **compuesto exclusivamente por un único (*OneConversationResult*):** En este caso la técnica exclusivamente trata de emparejar el $Automaton_{Req}$ con un único $Automaton_{reg}$ del repositorio. Es decir, el $Automaton_{res}$ estará compuesto exclusivamente por un único $Automaton_{reg}$. Esta técnica tiene un rendimiento alto, pero al contrario solo determina un subconjunto de los posibles resultados.

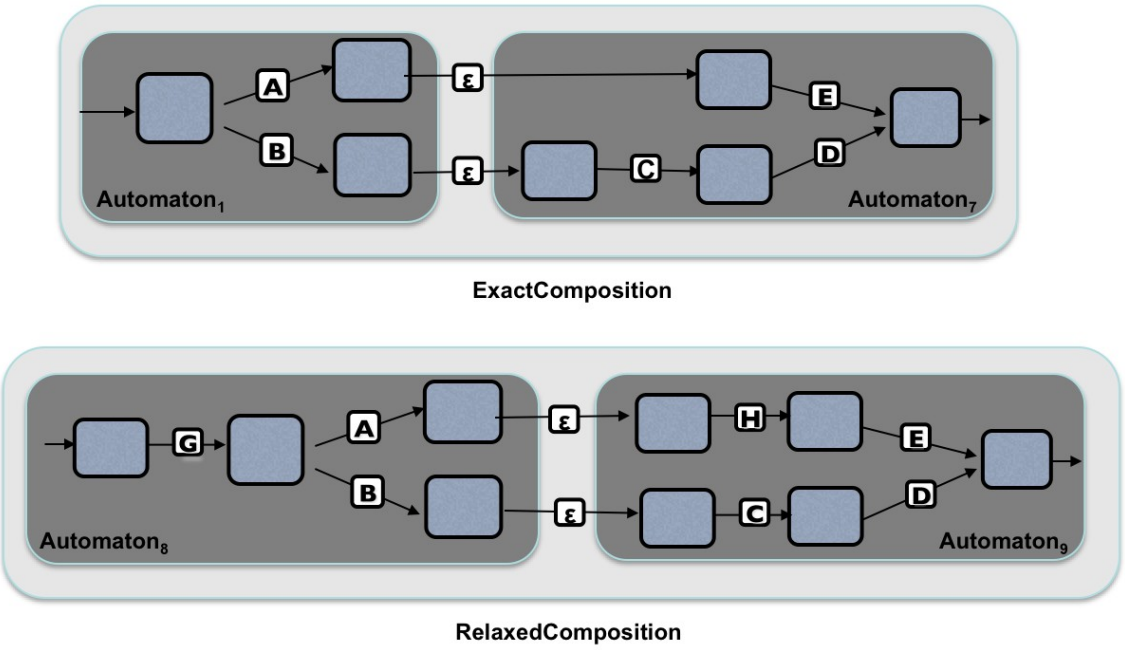


Figura 3.22.: Flexibilidad en los resultados en función del grado de relajación.

- **$Automaton_{res}$ compuesto por al menos un $Automaton_{reg}$ (*ManyConversationResult*):** En este caso la técnica trata de emparejar el $Automaton_{Req}$ tanto con uno como con varios $Automaton_{reg}$. Es decir, el $Automaton_{res}$ estará compuesto mediante la integración de varios $Automaton_{reg}$. Esta técnica tiene un rendimiento inferior al anterior, pero por el contrario ofrece todos los resultados posibles además de integrar a la anterior técnica, de esta manera *OneConversationResult* $\subseteq$ *ManyConversationResult*.

Como se puede apreciar en la Ilustración 3.23 el tipo *OneConversationResult* no es en esencia una técnica de composición ya que lo único que se en carga es de comprobar que existe una $Conversation_{reg}$ que es equivalente al solicitado, es decir, que comenzando en un estado inicial llegue a un estado final empleando elementos *Capability* equivalentes. El segundo en cambio, *ManyConversationResult* construye el resultado final combinando dos $Automaton_{reg}$.

Tomando como base las cuatro dimensiones previamente definidas, es posible representar las diferentes variables que definen todos los tipos de *ConversationSelection* que se van a analizar en

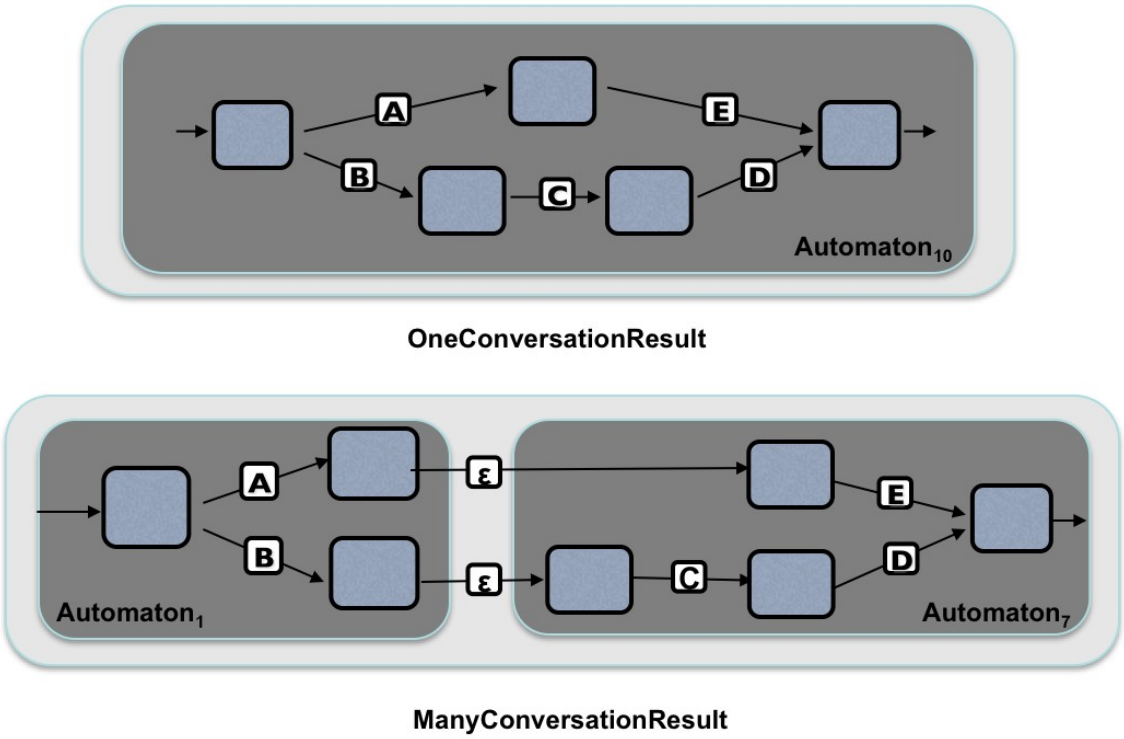


Figura 3.23.: Tipo de composición en función del tipo de resultado deseado.

el presente trabajo. Sin embargo, también es posible representar el mecanismo *CapabilitySelection*, definiendo para ello una ConversationSelection en la que *ConversationSelection* = <*NonAdaptiveRequest*,*Integration*,*ExactComposition*,*ManyConversationResult*>y empleando un subconjunto de Registry. Este subconjunto, denominado[3] , estará constituido exclusivamente por aquellas $Conversation_{reg}$ que representan una Conversation de tipo simple, es decir, estará únicamente compuesto por aquellas SimpleCapability visibles publicadas en el Registry. Por defecto y en caso de que no se defina lo contrario en la petición de descubrimiento el valor asignado y que establece la flexibilidad de la composición será la composición de tipo *CapabilitySelection*.

3.4 Descubrimiento de recursos

3.4.1 Consideraciones

Como se ha comentado en epígrafes anteriores el módulo de descubrimiento basa su funcionamiento en el API que ofrece el módulo de modelado y coordinación semántica. El módulo de descubrimiento da soporte a la necesidad de conocer qué dispositivos y servicios asociados están disponibles en ecosistema de ISMED.

En la etapa actual nos centraremos en el descubrimiento de los dispositivos existentes en el entorno, dejando para posteriores revisiones el apartado de descubrimiento de los servicios exportados por dichos dispositivos.

3.4.2 Primitivas a implementar

Tras la evaluación del escenario y del problema, se ha visto la necesidad de dar respuesta a las siguientes necesidades: descubrimiento de dispositivos y servicios, descubrimiento automático y sistema de suscripción. A continuación se detallarán cada una de las áreas involucradas en módulo de descubrimiento y las primitivas que ofrecerá.

[3]

3.4.2.1 Descubrimiento de dispositivos y servicios

Cada uno de los nodos tendrá la capacidad de conocer al resto de dispositivos conectados en su mismo espacio y por extensión el descubrimiento de sus servicios asociados. La búsqueda o descubrimiento se subdivide en dos posibles escenarios:

3.4.2.1.1. Descubrimiento Para esta primera hipótesis se ha de considerar la necesidad de un dispositivo, en un instante dado, de determinar que otros dispositivos están presentes en el sistema. El objetivo por tanto será conocer al resto de integrantes de la red. Para ello haciendo uso de la primitiva *query* (ofrecida por el módulo de modelado), se lanzará una consulta al espacio de tripletas por cada tipo de dispositivo conocido. El resultado de dicha query será transformado o mapeado a objetos que serán almacenados en una caché, con el objetivo de minimizar el lanzamiento de consultas al espacio. La primitiva a implementar es:

- *discoverDevices.*

En un futuro se incorporarán a este módulo nuevas primitivas que aporten la funcionalidad requerida para el descubrimiento de servicios.

3.4.2.1.2. Descubrimiento automático En un intento de automatizar el descubrimiento de dispositivos se ha modelado un protocolo de comunicación. Mediante la adopción de este protocolo un dispositivo será capaz de determinar la entrada o salida de otras unidades en la red de ISMED. El protocolo establecido es el siguiente:

1) Cuando un dispositivo se integra en la red realizará una suscripción (por cada tipo de dispositivo) para recibir las posibles notificaciones de los nuevos dispositivos. En este caso la primitiva a invocar será: subscribe (módulo de modelado). El template especificado para la suscripción será del tipo:

   ? <rdf:type><ismed:DeviceType>

2) Si un dispositivo presente en la red recibe una notificación procedente de otro nodo, comprobará si se trata de una notificación de entrada o salida.

Para satisfacer el protocolo adoptado, se implementarán las siguientes primitivas que serán ofrecidas por el módulo de descubrimiento:

- **start:** Suscripción para recibir notificaciones de nuevos dispositivos
- **stop:** Elimina la suscripción del dispositivo.

Si bien estas primitivas serán utilizadas en el sistema final, quizás deban ser modificadas, adaptadas o ampliadas para contemplar el descubrimiento de servicios mediante el sistema de notificaciones.

Cabe destacar en el caso del descubrimiento automático, que el rendimiento o efectividad del mismo está ligada a la eficacia del sistema de notificación en que se basa TripCom y a la existencia o no de nodos rendezvous. Por tanto el rendimiento de este mecanismo deberá ser evaluado para determinar su viabilidad.

3.4.2.1.2.1. Sistema de suscripción Siguiendo la idea propuesta en apartado de Descubrimiento automático, el módulo de descubrimiento requerirá la utilización del sistema de suscripción / notificación provisto por el API del módulo de modelado. Este sistema será ampliado para contemplar la suscripción a eventos como la llegada de nuevos servicios a la red de ISMED.

Por otra parte gracias este sistema, un dispositivo será capaz de suscribirse para recibir notificaciones a cerca de determinados eventos como por ejemplo las diferentes mediciones que puedan realizar los sensores de un nodo SunSpot.

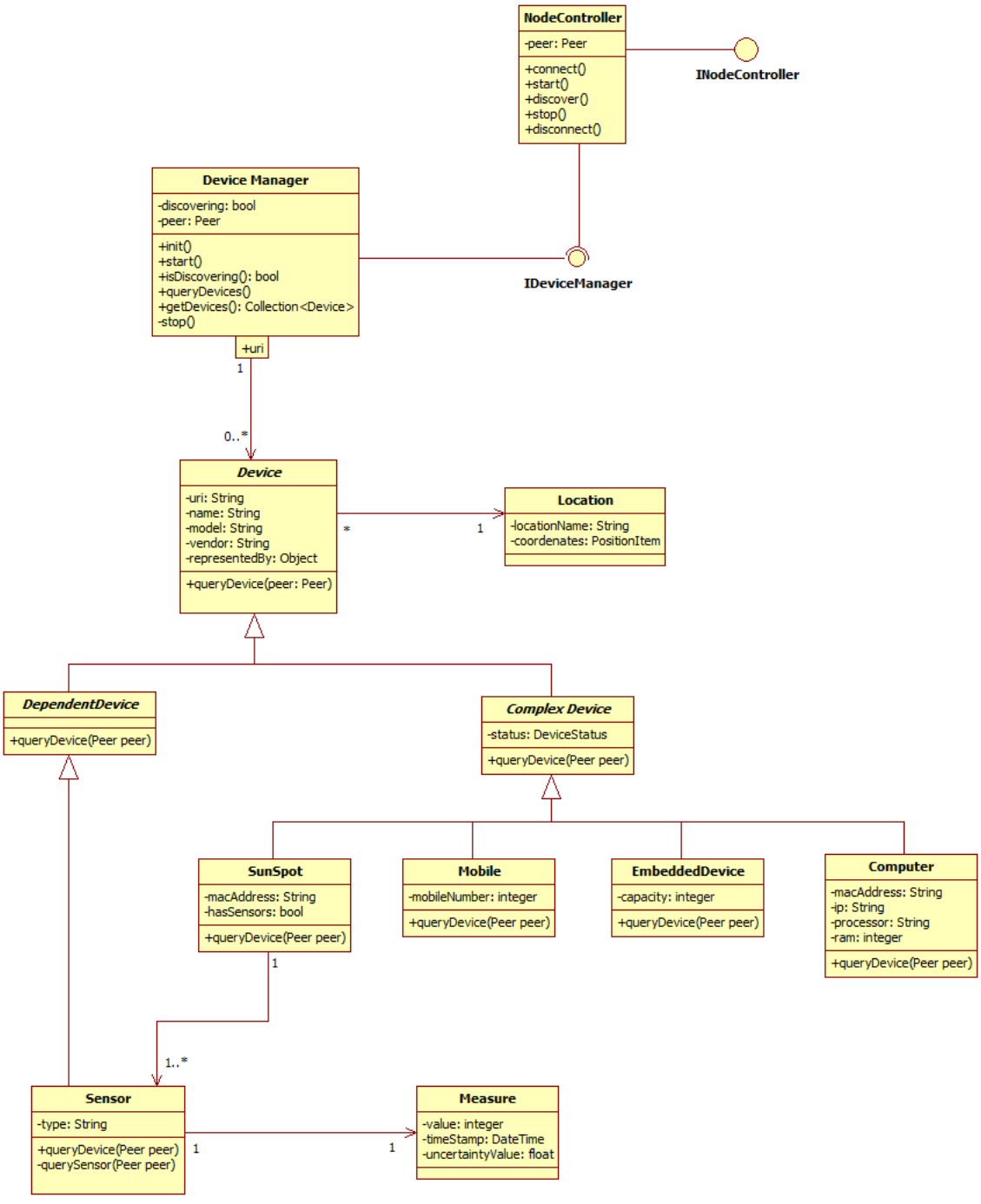


Figura 3.24.: Diagrama de clases del módulo de descubrimiento.

3.4.3 Diagrama de clases

3.5 Razonamiento

El problema del razonamiento distribuido ha demostrado ser un problema de una entidad tal que requeriría un esfuerzo y unos recursos que exceden claramente a los que el proyecto ISMED puede aportar.

Existen múltiples razones que hacen atractiva la idea de adoptar un enfoque de razonamiento distribuido en la inteligencia artificial [527]: Los humanos son capaces de hacerlo.

- Existen máquinas capaces de ejecutar tareas en paralelo.
- Se han conseguido mejoras en la eficiencia y en la velocidad de ejecución de algunas aplicaciones.
- El desarrollo modular de los sistemas .Facilitar el desarrollo, mantenimiento y gestión del conocimiento, así como las estrategias de razonamiento.
- Múltiples técnicas de razonamiento. Se pueden utilizar distintas técnicas de razonamiento, ya que algunas de ellas pueden proporcionar un mejor rendimiento a la hora de llevar a cabo determinadas tareas sobre un determinado conocimiento.
- Múltiples puntos de vista. El conocimiento que se necesita durante el razonamiento puede ser representados en diferentes dominios.
- Se consigue una mayor fiabilidad, ya que en caso de que un módulo sufra una avería o algún problema el sistema podría seguir funcionando.

Un sistema de razonamiento distribuido está compuesto por módulos separados, llamados agentes generalmente, y comunicados entre ellos. Cada agente puede actuar como una unidad independiente capaz de solucionar problemas. La mayoría de los sistemas existentes se pueden clasificar según su distribución:

- Sistemas de control centralizado con conocimiento compartido
- Sistemas de control y conocimiento totalmente distribuidos.

Estos sistemas deben de proveer una buena coordinación entre los agentes, así como una buena estructura de comunicación y técnicas de razonamiento distribuidos.

Volviendo a la semántica, el enfoque que se suele utilizar para soportar interoperabilidad entre ontologías es la definición de relaciones semánticas entre entidades que pertenecen a diferentes ontologías. Dicho enfoque se conoce con el nombre de "mapping" de ontologías. Sin embargo éstas no son suficientes para garantizar la interoperabilidad entre ontologías ya que también se debe proveer la capacidad de razonar con sistemas compuestos por múltiples ontologías interrelacionadas por "mappings" semánticos. El enfoque de razonamiento más extendido a la hora de razonar con múltiples ontologías, suele consistir en tratar las múltiples ontologías como una única global que incluya todas las ontologías y sus "mappings". Sin embargo este problema tiene las siguientes desventajas:

- No es escalable
- Perdida de la privacidad y autonomía del conocimiento ontológico
- Perdida de especificidad del lenguaje y del razonamiento

Ha habido diferentes propuestas de lógicas cuyo objetivo era definir una semántica formal para la integración de múltiples ontologías. No obstante los formalismos más influyentes para la Web Semántica están basados en lógicas descriptivas (DL). Los cuatros formalismos más importantes para la integración de múltiples ontologías son [733]:

- Distributed Description logics (DDL) es un framework para combinar ontologías DL mediante mappings semánticos .

- E-connections es un framework para combinar ontologías, cuya principal característica es que combina ontologías DL que cada una modelan una porción del dominio y no se solapan entre ellas.
- Package-based Description Logics (P-DL). Es un framework para ontologías distribuidas que permiten importar entidades de ontologías entre módulos.
- Integrated Distributed Description Logics (IDDL). Es un framework para alinear ontologías DL mediante mappings semánticos bidireccionales.

Una vez llegados a este punto debemos de hacer una distinción entre ontologías distribuidas y razonamiento distribuido. Cuando se habla de ontologías distribuidas se refiere a una colección de ontologías, físicamente o lógicamente, dispersas en múltiples nodos. Por otra parte el razonamiento distribuido se refiere a los casos en los que el razonamiento no está centralizado [796].

Siendo más concretos se puede hacer una distinción entre procesos distribuidos y paralelizados. Los procesos paralelos se suelen referir a las aplicaciones centralizadas dónde el objetivo es incrementar el rendimiento del sistema. Bajo este punto de vista la paralización está relacionada con la partición de un problema para obtener una mejoría en el rendimiento. Este es el caso de cuando existe una única ontología para la cual existe un control central que indica cómo dividirla. Por otra parte, la paralización también se puede usar con el problema de la paralización algorítmica del proceso de inferencia. La distribución suele referirse al caso en el que las aplicaciones están distribuidas en múltiples localizaciones físicas. En estos casos, el control central no existe, así como la paralización no se puede conseguir de la misma manera. En vez de eso, se enfoca en procedimientos de razonamiento descentralizado entre los diferentes nodos de la aplicación. Así que mientras en los sistemas paralelos ponen más atención en la relación coste/rendimiento, los sistemas distribuidos se enfocan en temas de distribución, componentes heterogéneos sobre los cuales no se dispone de un control global.

La combinación de las ontologías distribuidas y el razonamiento distribuido da lugar a dos dimensiones que permiten categorizar el problema del razonamiento distribuido con ontologías distribuidas. De la combinación de estas dimensiones podemos obtener cuatro casos bases como se muestra en la figura 3.25 (Georgios Trimponias):

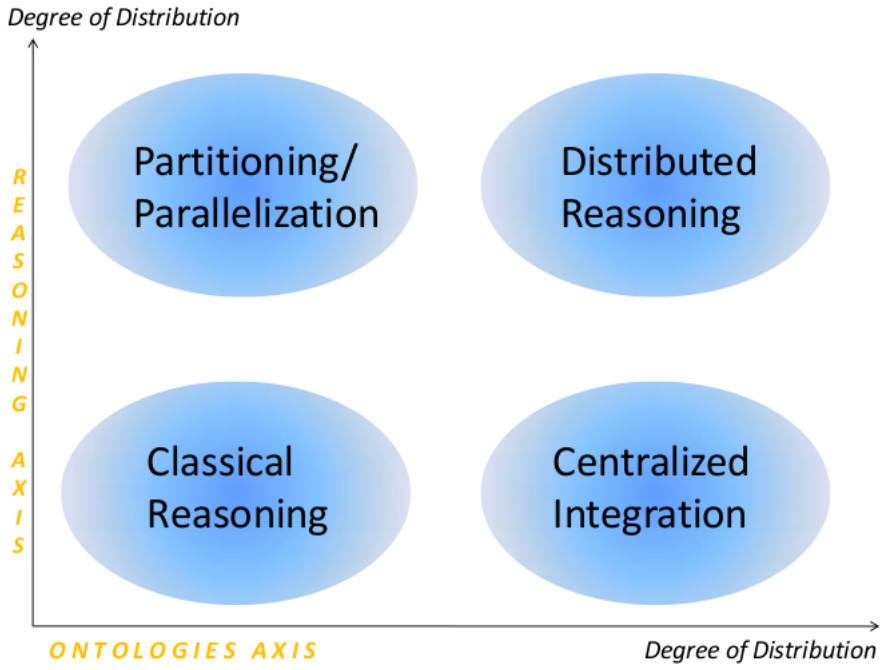


Figura 3.25.: Ontologías distribuidas frente razonamiento distribuido.

- El primer caso (Classical Reasoning) se refiere al caso clásico en el que tenemos una única ontología global sobre la que aplicamos razonamiento centralizado.

- En el segundo caso (Partitioning / Parallelization) tenemos una única ontología en la que el proceso de razonamiento es distribuido. Este caso es el más parecido al razonamiento paralelo, mencionado anteriormente, y posee relación con el problema de dividir una ontología con el objetivo de aumentar el rendimiento.

- El tercer caso (Centralized Integration) se da cuando existe una red de ontologías distribuidas sobre en las que únicamente se les aplica un único proceso de razonamiento o razonador.

- El cuarto y último caso (Distributed Reasoning) es cuando existe una red de ontologías distribuidas en las que se puede razonar aplicando procedimientos de razonamiento distribuidos.

Con el objetivo de superar los problemas anteriormente descritos, el proyecto DRAGO (Distributed Reasoning Architecture for a Galaxy of Ontology) [734] propone un algoritmo tableau distribuido, que es capaz de comprobar la satisfactibilidad de los conceptos en un conjunto de ontologías interrelacionadas mediante la combinación de procedimientos tableux locales que comprueban la satisfactibilidad dentro de una ontología.

DRAGO posee una arquitectura peer-to-peer en la que cada par registra un conjunto de ontologías y "mappings", y en las que el razonamiento esta implementado usando razonamiento local en las ontologías registradas y coordinándose con otros pares cuando las ontologías locales estén semánticamente conectadas con ontologías registradas en otros pares.

Es por ello, que el razonamiento que se realizará, al igual que en TripCom, será a nivel de cada nodo. A continuación se especifica el tipo de razonamiento que se puede lograr en cada nodo que use el framework desarrollado para ISMED.

- Sesame [147]. Pese a que el tipo de repositorio que se configure no tiene porque permitir inferencia (como es ocurre en el caso de tsc++), Sesame ofrece un motor de inferencia sobre RDFS.

- Owlim [460]. Tiene un motor de inferencias que permite razonar sobre RDFS (existe una optimización a costa de limitar su expresividad) y sobre dialecto de OWL cercado a OWLite (que ignora cardinalidad para valores mayores de 1).

- Dispositivos móviles.

  Se han encontrado serias dificultades para encontrar un motor de inferencia que pueda ser ejecutado en un dispositivo móvil. A lo largo de este curso, se publicó un artículo en el que se mencionaba un esperanzador motor de inferencia para dispositivos móviles JavaME con el perfil CDC llamado µOR [55]. µOR hace entailments para un subconjunto de OWL Lite.

  Desafortunadamente a día de hoy ese motor no es ni público, ni está preparado para la versión CLDC de JavaME (aunque según indicaban sus creadores se encontraban trabajando en ello).

  Mientras se busca una alternativa viable con la cual solucionar dicho problema, en tscME no existirá inferencia alguna. Las clases RecordStoreDataAccess y MemoryDataAccess almacenarán tripletas agrupándolas en espacios y grafos análogamente a Sesame u Owlim, y mediante MicroJena se podrá consultar cuando una tripleta cumple un patrón concreto, pero nunca inferir nada que no haya sido explícitamente escrito previamente.

3.6 Interrelación capa de coordinación semántica y módulos de descubrimiento, composición, razonamiento y aprendizaje

Los módulos de descubrimiento, composición, razonamiento y aprendizaje van a mediar con el módulo de modelado y coordinación semántica a través del API siguiendo el paradigma Triple Space Computing descrito en detalle en la memoria del 2008.

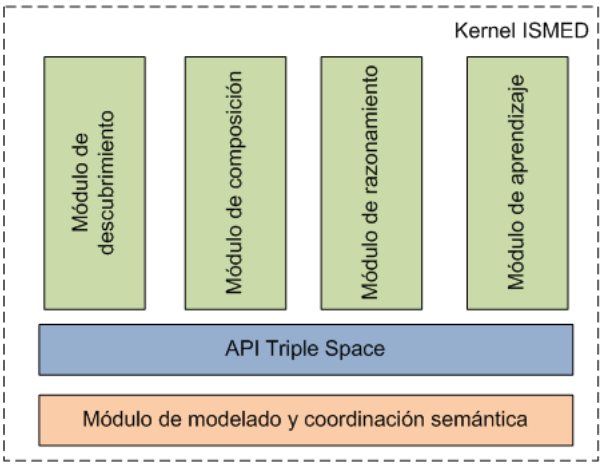


Figura 3.26.: Diagrama de clases del módulo de descubrimiento.

Tal API permitirá al módulo de descubrimiento formular consultas por medio de las primitivas que provee sobre servicios en el entorno. Asimismo, el módulo de descubrimiento podrá también registrarse para recibir notificaciones sobre la aparición de nuevos servicios.

El módulo de composición de servicios se apoyará en el módulo de descubrimiento para encontrar candidatos de servicios que puedan emparejarse. Además, se apoyará en módulo de razonamiento para poder inferir nuevas asociaciones entre servicios.

El módulo de razonamiento permitirá a los dispositivos equipados con middleware ISMED tener capacidad de reactividad, para de modo proactivo adaptarse a las nuevas circunstancias del entorno que les rodea.

El módulo de aprendizaje tendrá por misión identificar nuevas reglas de comportamiento del entorno y la optimización de las reglas actualmente existentes, que nutran el módulo de razonamiento.

La cooperación y entendimiento entre los diferentes módulos de ISMED requiere **la existencia de una ontología común entre ellos**, es decir, un vocabulario que permita describir los servicios exportados por los diferentes dispositivos del ecosistema de ISMED. Además, estos servicios habrán de estar anotados para que los módulos de razonamiento y aprendizaje sean capaces de interrogarlos para obtener conocimiento sobre el que razonar y aprender.

En conclusión, la API de Triple Spaces que proporciona el módulo de modelado y coordinación semántica constituye la infraestructura de consulta, coordinación y comunicación que puede usarse en la implementación de los demás módulos.

A este respecto, cabe destacar que debido a la limitada capacidad de cómputo de algunos de los dispositivos que se usarán, sólo la capa del módulo de modelado y coordinación semántica será implementada por completo en todos ellos (requiriendo en muchos casos además distintas versiones de dicho middleware).

De esta forma, si bien el aprendizaje, descubrimiento o la composición de servicios se realizarán en un nodo concreto de la red, los datos en los que se apoyarán se encontrarán distribuidos en todo momento.

4. IMPLEMENTACIÓN

Este capítulo describe la totalidad de la implementación realizada en el proyecto ISMED a partir del diseño realizado.

4.1 Modelado y coordinación semántica

El módulo de modelado y coordinación semántica ofrece una infraestructura para permitir la comunicación entre los distintos módulos que hay en cada nodo de la red. Es decir, está pensado para funcionar en conjunción con el resto de nodos.

Es por ello que pese a que se han construido multitud de aplicaciones apoyándose en el mismo, ninguna muestra mejor la esencia de las primitivas implementadas que aquella aplicación construida para probar la correcta integración entre los distintas versiones del middleware del módulo de coordinación.

Dichas aplicaciones se han hecho tanto para los nodos tscSE, como para los de tscME.

4.1.1 tscSE

Esta aplicación permite probar cada una de las primitivas mediante 4 pestañas:

- Connection. En la que se pueden especificar parámetros de configuración de tsc++ tales como dónde se guardarán los archivos de log, el nombre del peer, etc.
- Spaces. Desde aquí se podrá llevar a cabo la gestión de los espacios, y se podrán crear, unirnos a ellos o abandonarlos.
- Write. Desde aquí se puede llamar a la primitiva *write* seleccionando un ficheroen el que haya escritas ciertas tripletas (pudiendo ser el formato rdf u turtle) y volcándolas en un espacio determinado.
- Query. En esta pestaña se llamarán las consultas sobre el espacio basadas en tripletas *query*, *take* o *read*. El resultado podrá ser consultado después en View>Results, o compararlo directamente con las tripletas que se espera que devuelva especificadas en un archivo.
- QueryMultiple. Esta pestaña permite definir consultas SPARQL de una forma bastante básica y lanzar las consultas sobre el espacio.

4.1.2 tscME

Salvando las distancias surgidas por las limitaciones a la hora de definir una interfaz de usuario rica en dispositivos móviles, la aplicación desarrollada es análoga en comportamiento a la aplicación descrita en el anterior epígrafe, pero apoyándose en la librería tscME.

El diagrama mostrado en la figura 4.2 explica el funcionamiento de la aplicación. Primero se especifican los datos del *Rendezvous* al que un peer móvil necesita conectarse, después se permite gestionar espacios, y sobre espacios ya creados (*create*, *join* o *leave*) se pueden llevar a cabo primitivas básicas como *query*, *read*, *take*, *demand* o *write*.

4.1.3 Ejemplo de funcionamiento

Para profundizar en el funcionamiento de dicha aplicación se mostrarán las capturas de las distintas pantallas que forman la capa de la vista de la misma relacionandolas con las pantallas definidas en la figura 4.2.

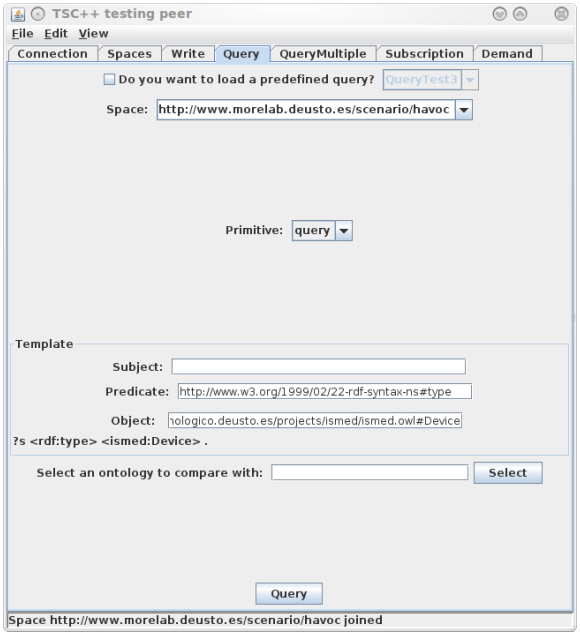


Figura 4.1.: Take sobre el espacio tsc://espacio5, de la que se esperan los resultados definidos en el archivo filename2.owl.

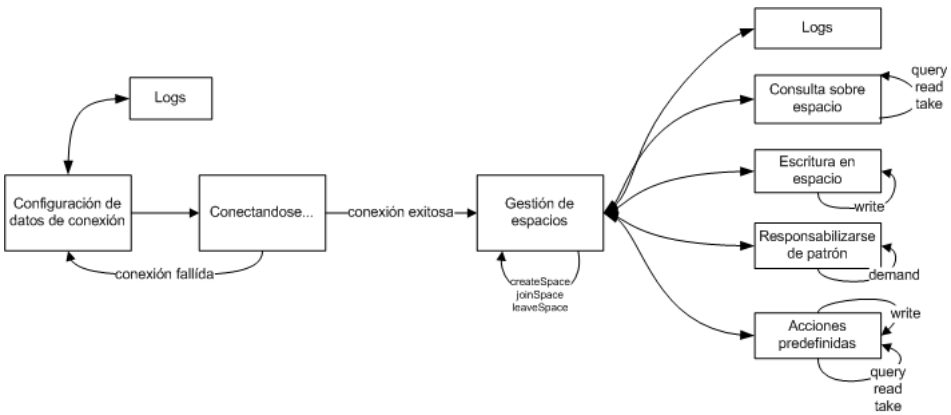


Figura 4.2.: Esquema de la aplicación de prueba de integración de tscME.

Figura 4.3.: Pantalla de conexión.

En la figura 4.3 se muestran los parámetros básicos de configuración de la red de Jxme que se deberán proveer. El más importante es la dirección del Rendezvous que propagará los mensajes del móvil al resto de nodos. Una vez hecho esto, el usuario querrá unirse a algún grupo, creando un espacio y uniendose a él (figura 4.4).

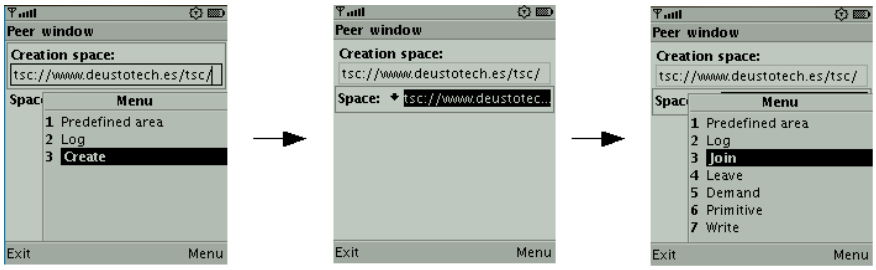


Figura 4.4.: Gestión de espacios.

Las aplicaciones descritas permiten probar otras aplicaciones y gestionar los espacios creados. Además, permitirían construir escenarios tan complejos como quisiésemos, logrando una mejor comprensión del funcionamiento de Triple Space. Un escenario podría ser el mostrado en la figura 4.5.

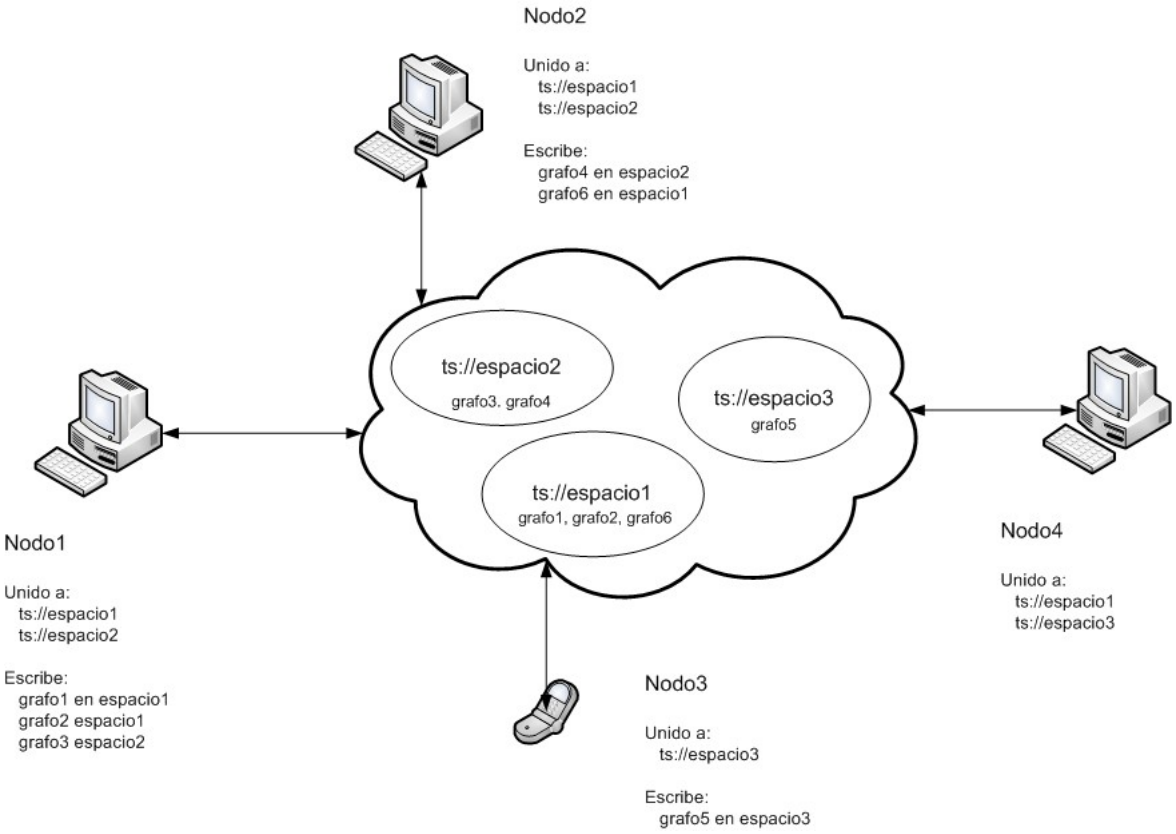


Figura 4.5.: Situación de partida para el ejemplo explicado en este epígrafe.

En él se puede apreciar mejor como si el nodo4 realizase una consulta al espacio3 preguntando por información contenida en el grafo3, nadie le respondería nada porque esa información la tendría el nodo1 asociada al espacio2.

De la misma forma, si el nodo1 realizase una consulta query sin especificar sujeto y predicado, pero indicando el objeto “http://perro"(“?s ?p <http://perro>.”) sobre el espacio2, recibiría una

respuesta del nodo2 con todas aquellas tripletas del grafo4 que tuviesen "http://perro" en el objeto y las uniría con aquellas de su repositorio local que cumpliesen con dicho patrón (las correspondientes al grafo3).

Si el nodo4 realizase una take sin especificar sujeto, predicado u objeto sobre el espacio3, recibiría el grafo5 del nodo3 la primera vez, pero nada la segunda, porque el nodo3 habría eliminado ese grafo de su repositorio la primera vez.

Finalmente, si el nodo2 hiciese una read sobre el espacio1 preguntando por una información contenida en el grafo1, el grafo2 y el grafo4, recibiría el grafo1 o el grafo2 desde el nodo1, dado que el grafo4 estaría en el espacio2 y un mismo nodo sólo puede devolver cómo máximo un grafo ante una take o una query.

4.2 Aprendizaje

El módulo de aprendizaje se ha desarrollado, para facilitar la integración con los otros módulos, en Java. Además se ha desarrollado un interfaz gráfico que facilita la interacción del usuario con el sistema. A continuación, utilizando el interfaz gráfico, se detallarán los pormenores de la implementación, así como las diferentes opciones que posibilita dicha implementación. La pantalla inicial permite seleccionar el conjunto de datos a utilizar en el proceso de aprendizaje.

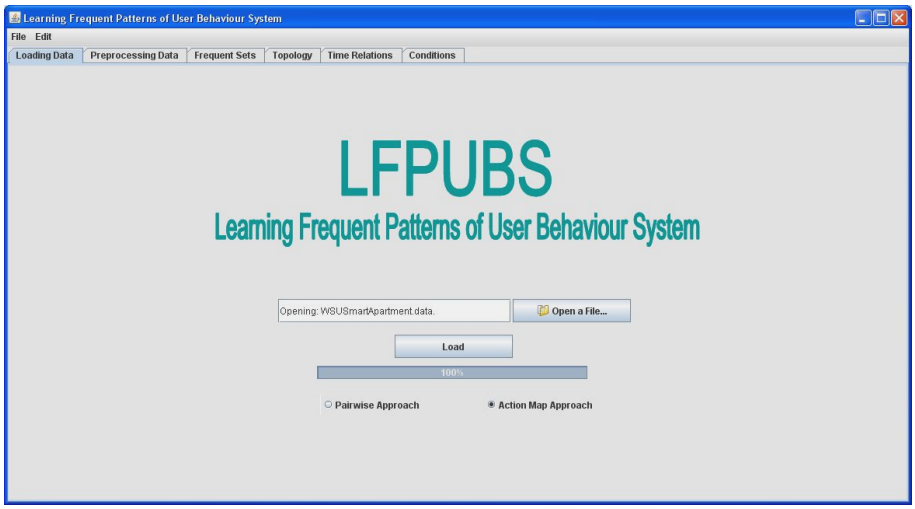


Figura 4.6.: Pantalla inicial del interfaz gráfico.

Como se ha detallado en el apartado de diseño, el primer paso es la transformación del conjunto de datos. Para ello, se permiten utilizar tanto información temporal como información acerca de las acciones. El resultado de este paso es el conjunto de datos transformado.

Una vez que los datos hayan sido transformados, el módulo de aprendizaje permite llevar a cabo todos los pasos definidos en el diseño.

4.2.1 Descubrimiento de secuencias frecuentes

El primero de ellos es el de descubrir los conjuntos de acciones que son frecuentes. Para ello, se pueden definir varios tresholds que definirón el treshold para considerar un conjunto como frecuente o no. Para ello se han definido tres parámetros que permiten descubrir el conjunto de acciones que son frecuentes:

- Confidencia mínima del conjunto de acciones.
- Confidencia mínima para el conjunto de acciones extras.
- Conficencia mínima para el conjunto de secuencias extras.

Estos tres parámetros se pueden ver en la siguiente figura.

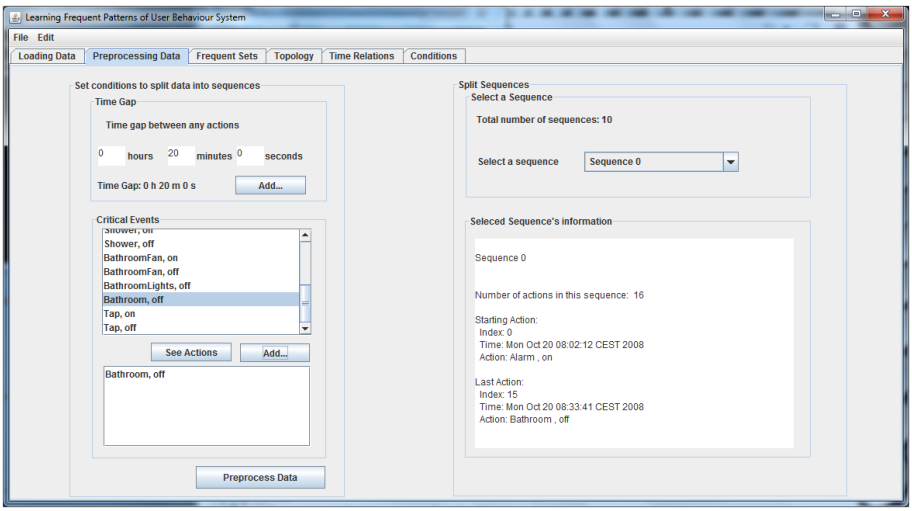


Figura 4.7.: Transformar los datos en secuencias.

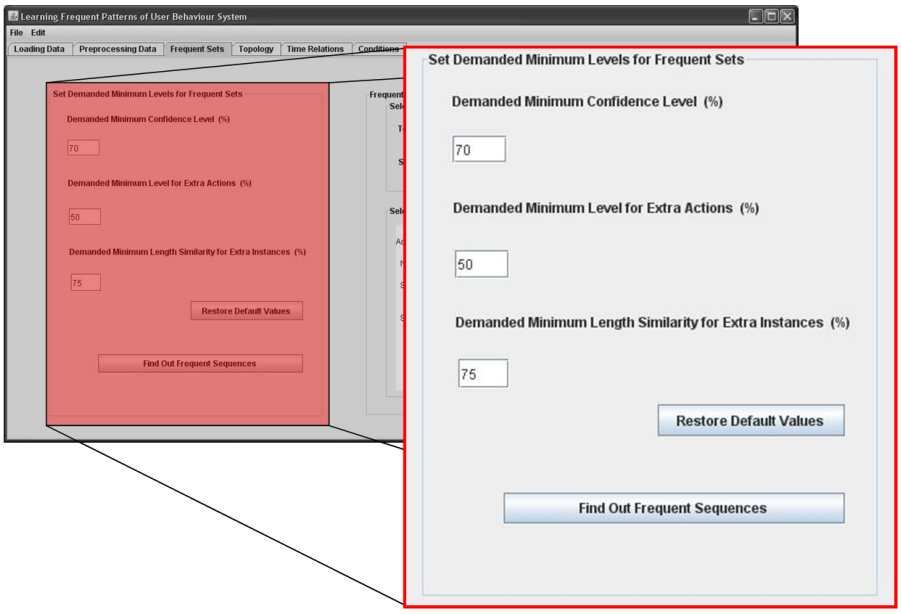


Figura 4.8.: Definición de los parámetros que permiten descubrir conjuntos de acciones frecuentes.

4.2.1.1 Identificando conjuntos básicos de acciones frecuentes

El objetivo de este primer paso es tan simple como eficiente. Teniendo en cuenta la confidencia mínima definida por el usuario, el sistema descubre aquellos conjuntos de acciones que ocurren más veces que dicho nivel. A partir de ese momento, esos conjuntos son considerados como *Conjuntos Básicos Frecuentes*. Para ello, se ha aplicado el algoritmo Apriori que descubre conjuntos de acciones en grandes cantidades de datos.

Aunque los conjuntos de acciones descubiertos son frecuentes, la complejidad de los entornos inteligentes hace que sean necesarias otras tareas para descubrir los conjuntos de acciones frecuentes.

4.2.1.2 Identificando acciones extras

Una de esas particularidades es que varias acciones que no hayan sido identificadas como parte de un conjunto frecuente, sean frecuentes si se consideran solamente las secuencias donde el conjunto de acciones frecuente sucede. Así, considerando sólo dichas secuencias y teniendo en cuenta la confidencia mínima definida por el usuario para las acciones extras, se lleva a cabo otro proceso de descubrimiento de acciones frecuentes. Dichas acciones, conjuntamente con las acciones descubiertas previamente, definen el conjunto de acciones frecuentes.

4.2.1.3 Identificando las secuencias

Finalmente, para facilitar la tarea de los siguientes pasos, es necesario identificar aquellas secuencias donde el conjunto de acciones tiene lugar. Al igual que con las acciones extras, debido a la complejidad de los entornos, puede que existan secuencias donde no se incluyan todas las acciones pero si la mayoría. En este sentido, teniendo en cuenta la confidencia definida por el usuario, se identifican aquellas secuencias donde ocurren más acciones que las definidas por el usuario.

4.2.2 Descubrimiento de la topología

Una vez que se hayan descubierto los conjuntos frecuentes, el siguiente paso es descubrir el orden (la topología) en el cual el usuario normalmente desarrolla dichas acciones.

En este sentido, la definición del comportamiento mediante Mapas de acciones facilita la comprensión de dicho comportamiento. Debido al poco trabajo realizado en este ámbito, se han tenido que trasladar ideas de otros ámbitos, tales como Workflow Mining [849, **?**, 854]. Al igual que en el paso previo, el interfaz del usuario permite definir los parámetros que definen cómo se va a descubrir dicho orden.

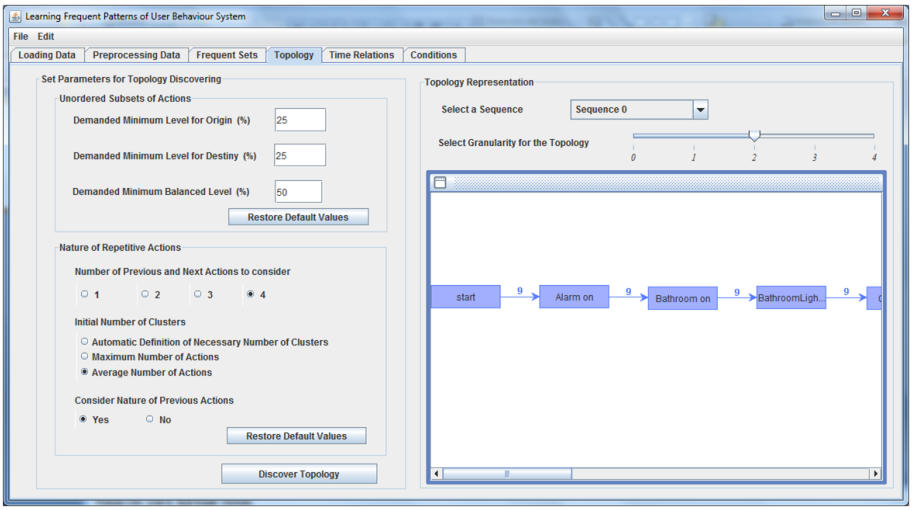


Figura 4.9.: Definición de los parámetros que permiten descubrir la topología.

La complejidad en este paso viene dada por el comportamiento del usuario. En este sentido, se han considerado la probable ocurrencia de acciones repetitivas y subconjunto de acciones no-

ordenados. Las acciones repetitivas son aquellas acciones cuya naturaleza es la misma pero que tienen diferente objetivo. Por otro lado, un subconjunto de acciones no-ordenados es dos o más acciones donde es imposible determinar un orden concreto.

Finalmente cabe mencionar que el sistema identifica la frecuencia mínima de la topología para evitar que hay incongruencias en la representación del comportamiento. Así, utilizando un algoritmo de búsqueda identifica la mínima granularidad posible para que haya un camino desde la acción inicial hasta la acción final.

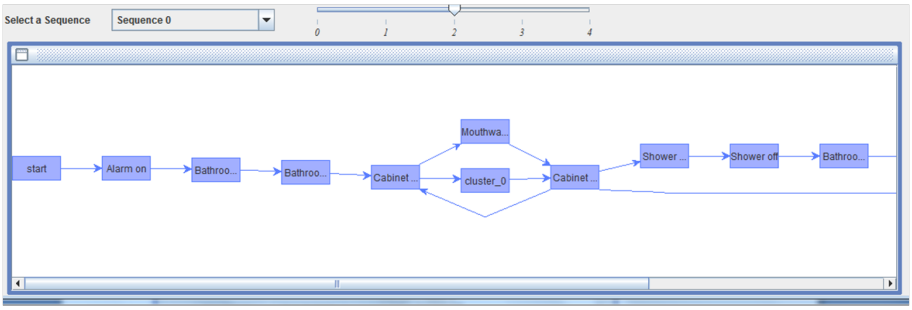


Figura 4.10.: Definición de los topología con la mínima granularidad.

4.2.3 Descubrimiento de Relaciones temporales

La topología en sí define una relación temporal entre las acciones del usuario. Esta relación viene definida por el termino 'after', ya que la relación entre las dos acciones es solamente definida mediante una relación cualitativa. El objetivo de este paso es descubrir, si es posible, valores cuantitativos que definen mejor esas relaciones. Aunque las relaciones cualitativas también representan una relación temporal, esta relación es más preciso si esa relación es cuantitativa. Las relaciones cuantitativas, además de permitir comprender el comportamiento del usuario, permiten automatizar las acciones.

Para ello, el sistema provee dos algoritmos (algoritmo básico y algoritmo EM) cuya única diferencia es la forma en que se agrupan las diferentes instancias. El usuario del sistema puede elegir entre estos dos algoritmo tal como muestra la siguiente imagen.

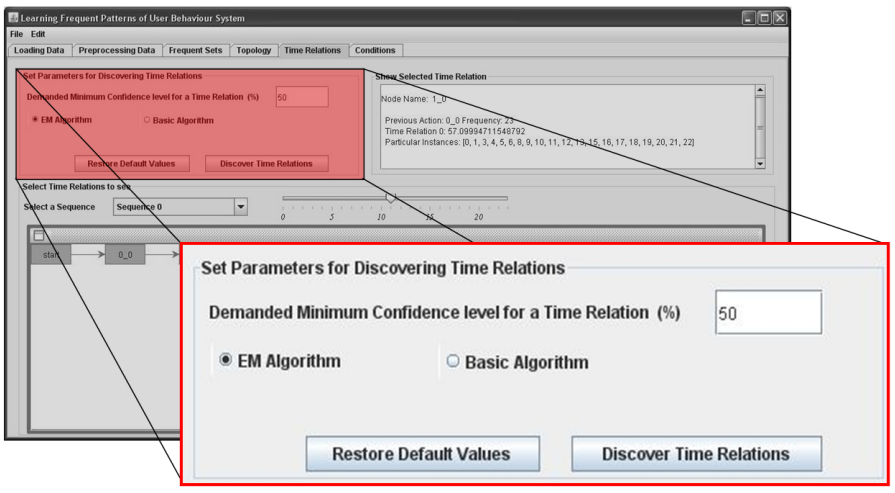


Figura 4.11.: Selección del algoritmo para el descubrimiento de relaciones temporales.

Ambos algoritmos están basado en la idea de agrupación de las diferentes instancias. Una vez que los grupos hayan sido definidos una relación viene definida por un valor cuantitativo si su frecuencia es mayor que la demandada.

4.2.4 Descubrimiento de condiciones

Finalmente, es necesario contextualizar el comportamiento. Para ello se han identificado dos tipos de condiciones, por un lado las condiciones específicas y por otra parte las condiciones generales.

Las condiciones específicas son necesarias cuando una acción es seguida por más de una acción. En esos casos, es necesario definir bajo qué condiciones se sigue un camino u otro. Para descubrir dichas condiciones se generan dos tablas (cubierto y no-cubierto) donde se almacena la información temporal (hora de la semana, hora del día, mes,...) así como la información contextual (temperatura, humedad, ...) de las ocurrencias de cada instancias. Una vez que las tablas hayan sido generadas, se utilizan técnicas de clasificación [861] para separa estas tablas y generar las condiciones específicas.

Shower Off – BathroomFan On

	Sequence 1	Sequence 5	Sequence 6	Sequence 8
time of day	08:29:37	08:29:28	08:30:39	08:23:29
day of week	Monday	Friday	Monday	Wednesday
Temp. Bathroom	19	22	21	17
Hum. Bathroom	72	75	71	78

Shower Off – BathroomLights On

	Sequence 3	Sequence 10
time of day	08:28:18	08:29:07
day of week	Wednesday	Friday
Temp. Bathroom	22	18
Hum. Bathroom	65	69

Figura 4.12.: Tablas a separar para la generación de condiciones específicas.

Por otro lado están las condiciones generales que permiten contextualizar el comportamiento en general. Para ello, se utiliza la información temporal (hora del día y día de la semana). Así, el objetivo principal es generar condiciones generales que cubran todas las instancias.

Una vez que las condiciones generales son descubiertas, el comportamiento del usuario está totalmente definido. En la siguiente figura se puede ver cómo se representan las condiciones mediante el interfaz gráfico.

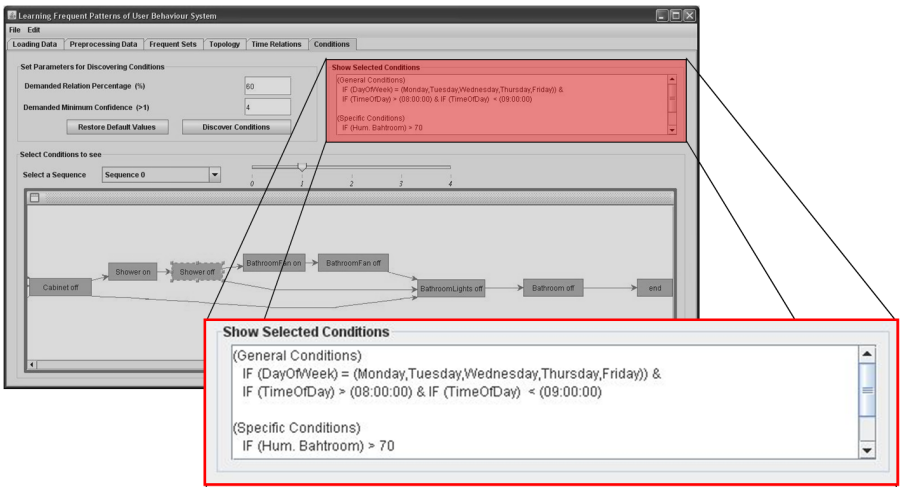


Figura 4.13.: Representación de las condiciones generales.

4.2.5 Integración del módulo de aprendizaje con el módulo de coordinación semántica

Como se ha explicitado en el apartado de diseño, el descubrimiento de las condiciones se lleva a cabo generando dos tablas donde se explicitan las instancias cubiertas y no-cubiertas. A continua-

ción, mediante técnicas de clasificación, se extraen las condiciones que diferencian esas dos tablas. Para dicha separación se almacenan en dichas tablas la información proveniente de los sensores de contexto.

Al utilizar toda la información del contexto (temperatura, humedad, luminosidad) se dan casos donde la condición que mejor define la relación no tiene significado. Por ejemplo, si la relación es entre las acciones .entrar oficinaz .encender la luz", es lógica que dichas acciones están relacionadas por la luminosidad de la oficina, es decir, que el usuario encienda la luz si la luminosidad de la oficina no es suficiente.

En una primera aproximación se utilizaba toda la información del contexto para definir las condiciones, obteniendo a veces condiciones no muy lógicas que relacionaban las acciones con variables del entorno de diferente naturalidad. Por ejemplo, siguiendo con el ejemplo anterior, se podía llegar a definir una condición que decía que el usuario encienda la luz de la oficina cuando la humedad de la misma sea menor que x.

Así, se decidió que integrando el módulo de aprendizaje con el módulo de coordinación semántica se podría utilizar la información semántica del entorno para llevar a cabo un descubrimiento de condiciones mucho más eficiente. Para ello, se decidió utilizar dos parámetros que definen la naturalidad de las acciones: la ubicación de la acción y el tipo de información del contexto (temperatura, humedad, luminosidad).

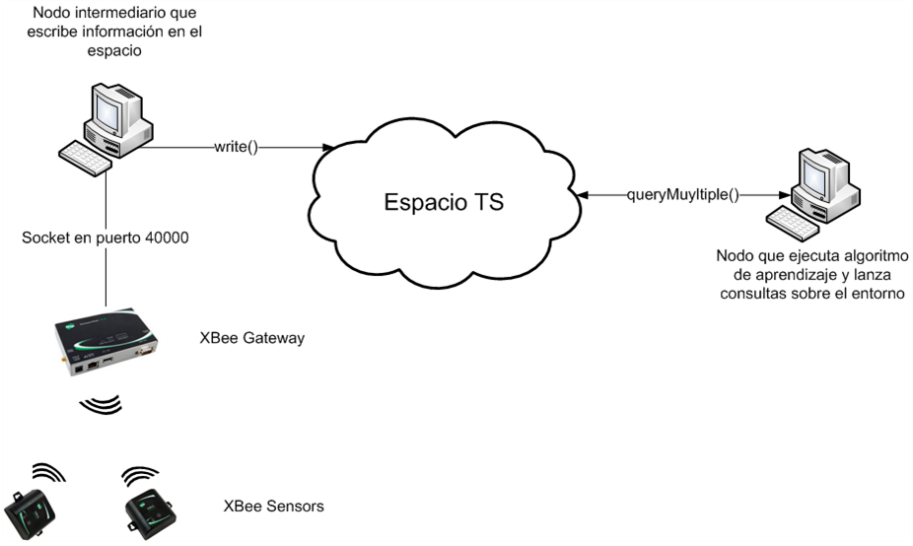


Figura 4.14.: Esquema de la aplicación de integración desarrollada.

Como se puede apreciar en el diagrama, para dar solución a dicha problemática, se definieron dos nodos que trabajan sobre un mismo espacio: un nodo que escribiese y otro que consultase la información escrita por él. El nodo escritor, introduce la información de dónde se encuentran qué dispositivos, y los sensores que componen estos dispositivos (por ejemplo, de humedad o de luminosidad).

La particularidad de este nodo es que al estar implementado en Java, no se puede cargar en el Gateway XBee, que sólo permite cargar scripts desarrollados en Python. Por ello, el Gateway se comunicará mediante un socket con un servidor hecho en Java que será quien a su vez escriba las tripletas que recibe por ese socket en el espacio TS.

El nodo consultor es utilizado por el algoritmo de aprendizaje, que es el encargado de consultar que sensores son de una naturaleza concreta y están ubicados en un lugar concreto, utilizando la primitiva queryMultiple con la consulta mostrada a continuación.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ismed: <http://www.tecnologico.deusto.es/projects/ismed/ismed.owl#>
CONSTRUCT {
  ?mota ismed:name ?name .
```

```
}
WHERE {
  ?mota rdf:type ismed:XBee .
  ?mota ismed:name ?name .
  ?mota ismed:madeUpOf ?sensor .
  ?sensor rdf:type <TYPE> .
  ?mota ismed:locatedIn <PLACE> .
}
```

Como se puede apreciar, en esta primera iteración, sólo se escribe información relativa a los dispositivos que existen en el entorno y no las capturas que estos mismos van realizando a lo largo del tiempo. El siguiente paso comprendería escribir dichas medidas y mantenerlas actualizadas, y sería muy sencillo de implementar siguiendo la filosofía de la aplicación descrita en este epígrafe.

4.3 Composición de servicios

Pese a que se ha realizado un gran trabajo teórico en el apartado de composición, que ha dado incluso origen a una tesis, finalmente no se ha podido de integrar dicho trabajo en el resto del middleware.

Cualquier mecanismo de descubrimiento de servicios está compuesto por los siguientes elementos: *arquitectura de descubrimiento* que se utiliza como base para el descubrimiento, *descripción de servicios* y *selección de servicios* (mecanismo de razonamiento para el emparejamiento de servicios) [432]. Tal y como apuntan en [793], Triple Space es un paradigma que resuelve por si sólo el descubrimiento de servicios de los procesos de composición de servicios. Para ello, basta con buscar servicios igual que se buscan otros recursos dentro del espacio (ver sección 4.4).

Por lo tanto, para integrar la composición de servicios, los pasos a seguir serían adoptar un estándar de descripción de servicios (ver sección 2.4.4) e implementar un mecanismo de emparejamiento o selección de servicios en cada nodo (ver 2.3).

Pese a ello, se ha optado por aunar esfuerzos en lograr respetar el paradigma basado en recursos en el que se centra Triple Space, reimplementando para ello la escritura de conocimiento en el mismo, más allá de explorar la forma en la que servicios web semánticos encajarían dentro el TS.

Sobre dicho paradigma basado en recursos, se pueden crear facilmente mash-ups sobre los datos semánticos ejecutando consultas SPARQL complejas sobre Triple Space (ver escenario de la sección 5.1). Estos mash-ups serían el equivalente a la composición de servicios web tradicionales, donde se mezcla funcionalidad para crear una funcionalidad más compleja, en enfoques orientados a recursos, donde se combina y refina información para extraer información de mayor utilidad.

4.4 Descubrimiento de recursos

A continuación se detallan aspectos relativos a la implementación del módulo de descubrimiento, que suponen una continuación natural de lo explicado en el epígrafe 3.4.

4.4.1 Consideraciones sobre la implementación

Cabe destacar que, si bien el kernel de ISMED se ejecutará en los diferentes dispositivos mencionados en el epígrafe 3.1.3, para el caso que nos ocupa y con el objetivo de clarificar la dimensión y el contenido del módulo de descubrimiento, el despliegue de dicho módulo se ha realizado sobre un nodo cuyo cometido es monitorizar el sistema ISMED.

4.4.2 Estructura del módulo

Con el fin de satisfacer las necesidades del módulo de descubrimiento y tal como ilustra el diagrama de clases presentado en el apartado de diseño (ver el epígrafe 3.4.3), se ha dividido la estructura del módulo de descubrimiento en dos bloques funcionales diferenciados:

- Controlador de Nodo (Node Controller): La misión principal del controlador es gestionar y coordinar la conexión y desconexión de los dispositivos al espacio de tripletas. El controlador proveerá de dos servicios: *connect* y *disconnect* para este fin. Por otra parte hará uso de una serie de primitivas ofrecidas por el API que nos provee el módulo de coordinación semántica:
  - *Conexión* (**connect**): Primitiva ofrecida por el módulo de descubrimiento para establecer el vínculo entre un dispositivo y el Triple Space. El proceso de conexión requiere efectuar los siguientes pasos: Creación del espacio asociado, unión a dicho espacio y arranque del kernel del Triple Space. Durante el proceso de conexión se invocarán las siguientes funciones del API ofrecido por el módulo de modelado:
    - *create(space)*
    - *join(space)*
    - *startup()*
  - *Desconexión* (**disconnect**): Primitiva ofrecida por este módulo para interrumpir el enlace entre un dispositivo del ecosistema y el espacio asociado. Para a la desconexión se hace uso de la primitiva:
    - *shutdown()*
- Gestor de dispositivos (Device Manager): Este bloque funcional da respuesta, ofreciendo una serie de primitivas, al resto de necesidades: Descubrimiento de dispositivos y sistema de suscripción. Destacar que con el objetivo de minimizar el lanzamiento de consultas sobre el espacio, el gestor de dispositivos mantendrá una caché de dispositivos activos y sus sensores asociados, en caso de poseerlos.
  - Sistema de suscripción: Tal y como se ha comentado en el apartado de diseño referente a este módulo, se ha implementado un protocolo de comunicación entre dispositivos, con el objetivo de conseguir que una entidad reconozca la entrada / salida de dispositivos en la red. Se ha hecho uso de las primitivas PUSH que ofrece el módulo de modelado, para satisfacer dicho protocolo. Primitivas ofrecidas:
    - **start:** Se invoca en el momento de ingresar en la red, para notificar al resto de nodos su llegada (*advertise*) y realizar la suscripción (*subscribe*) al espacio con el objetivo de recibir las notificaciones producidas por la llegada de nuevos nodos al sistema[1].
    - **stop:** Primitiva invocada cuando un dispositivo abandona la red. El objetivo es cancelar la suscripción realizada al ingresar (*unsuscribe*) y notificar la salida al resto de nodos (*advertise*).
  - Descubrimiento de dispositivos: Gracias a la primitiva **discover_devices** un nodo será capaz de conocer al resto de dispositivos que conforman la red de ISMED. Como el resto de funciones, se apoya en el módulo de modelado y coordinación semántica. Concretamente se hará uso de la primitiva query provista por el API de dicha capa.
    - Query: Se lanzará, al espacio de tripletas, una consulta por cada tipo de dispositivo contemplado en la ontología de ISMED. Esta consulta se realizará en forma de "template", compuesto por un sujeto, predicado y objeto. En este caso el sujeto es el componente a localizar, siendo el predicado la constante RDF *type* y el objeto el tipo de dispositivo definido en la ontología.
      - Ejemplo de "template" para obtener todos los dispositivos SunSpot:
        ? <rdf:type><ismed:SunSpot>

      De esta forma se determinará el número e identidad de los dispositivos presentes en la red. Por cada entidad encontrada se lanzarán determinadas query's para obtener toda la información (detallada en el diagrama de clases) relevante de cada dispositivo.

[1] Al recibir una notificación por parte de un nuevo dispositivo se procederá a su descubrimiento e inclusión en la caché. Este mecanismo es idéntico al descrito en el apartado de descubrimiento de dispositivos.

4.4.3 Esquema de arquitectura y funcionamiento

4.4.3.1 Conexión

Tal y como ilustra la figura 4.15, el módulo de descubrimiento ofrece la primitiva **connect** para establecer la unión con el espacio de tripletas.

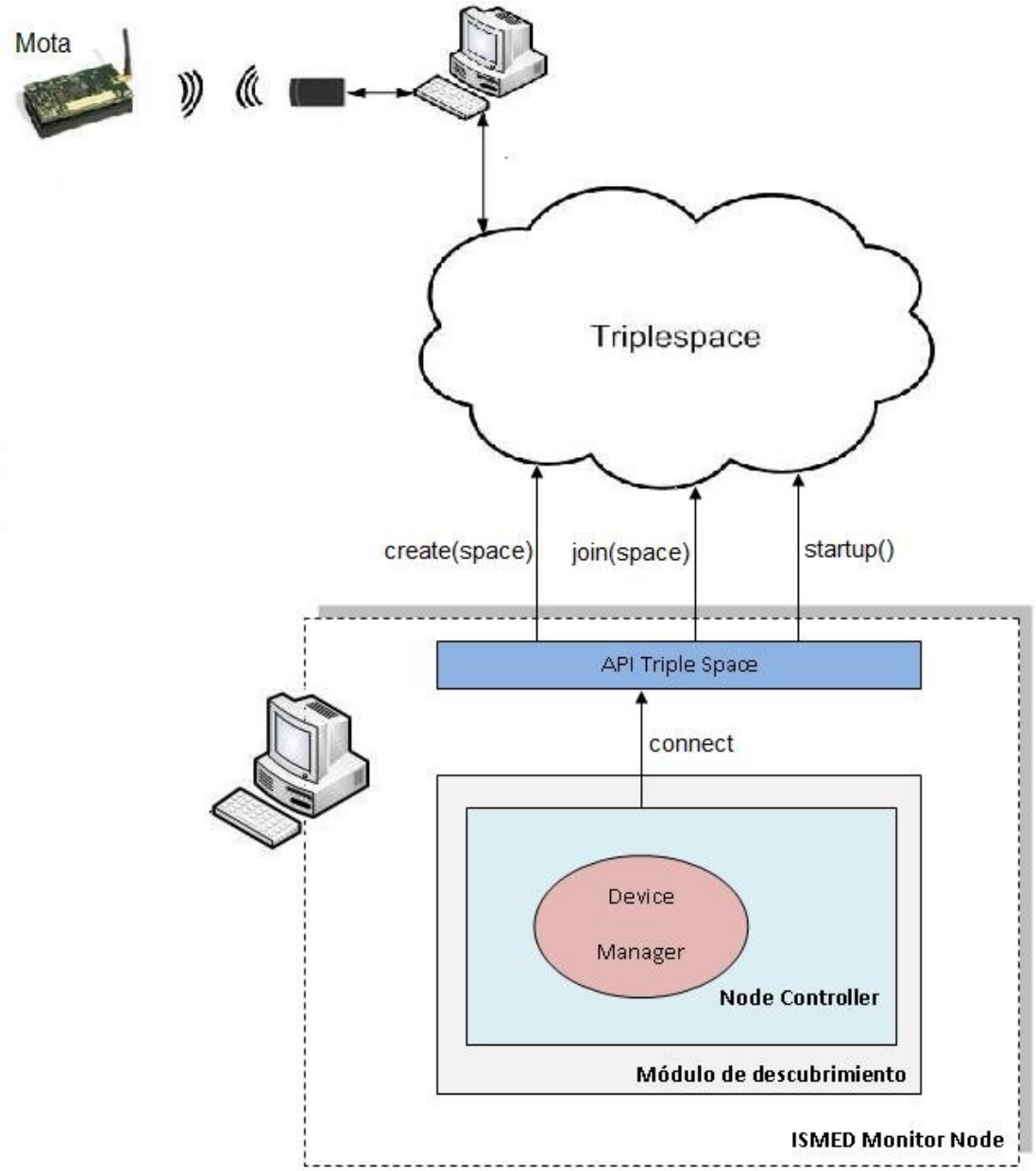


Figura 4.15.: Entrada de un nuevo dispositivo - conexión.

4.4.3.2 Suscripción

Con la llegada de un nuevo dispositivo la se realiza una suscripción para recibir notificaciones de otros nodos y notificar al resto de miembros de la red la presencia del mismo.

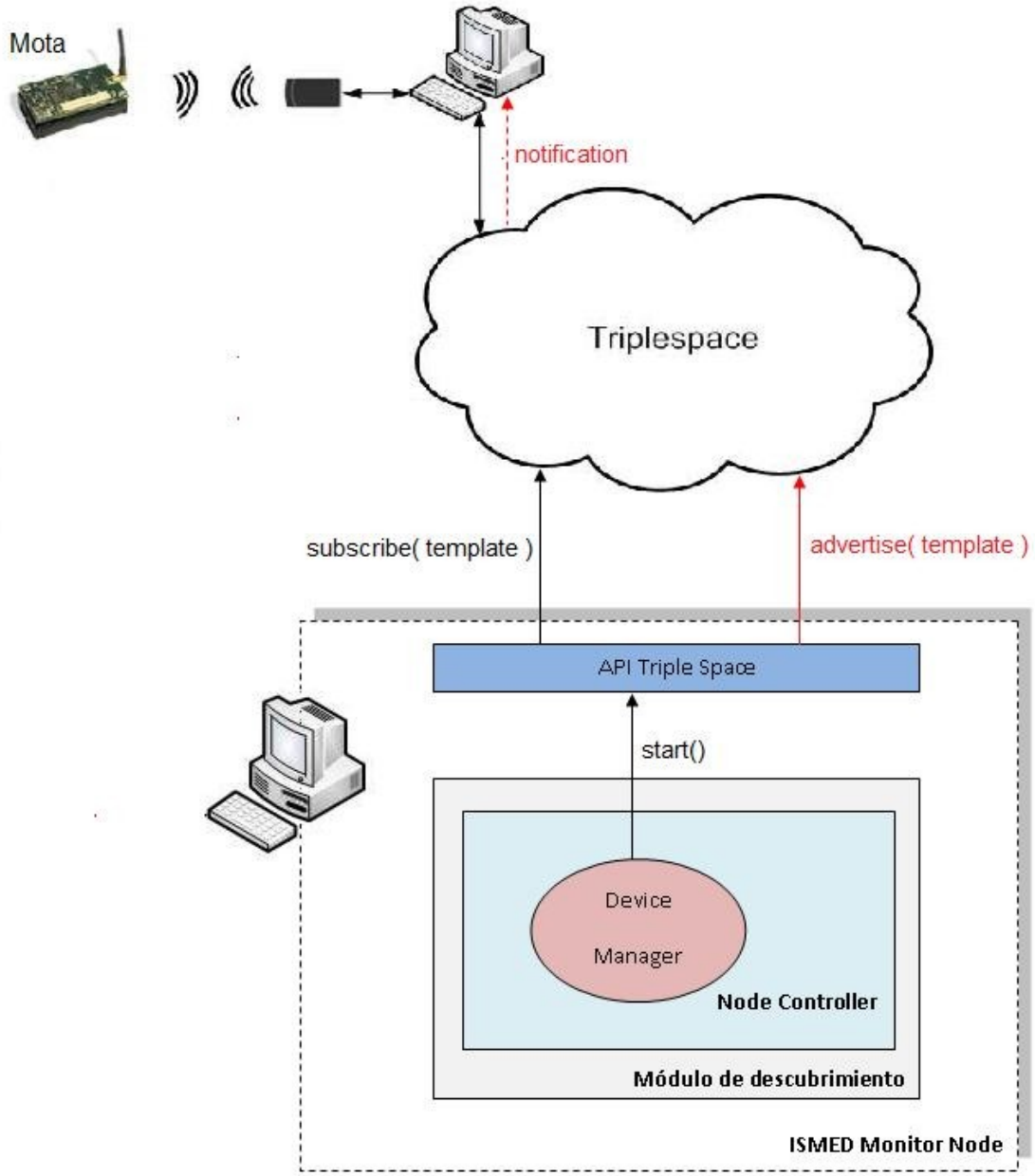


Figura 4.16.: Suscripción y notificación de llegada.

4.4.3.3 Descubrimiento

El descubrimiento de dispositivos puede producirse bien por necesidad del nodo en cuestión o por la recepción de una notificación. En el primer caso se analizará la red para detectar la presencia de nuevos dispositivos, mediante querys, sobre el espacio de tripletas. En el segundo caso solo se realizará el descubrimiento del dispositivo que lanzó el aviso, pero de manera análoga al primer caso.

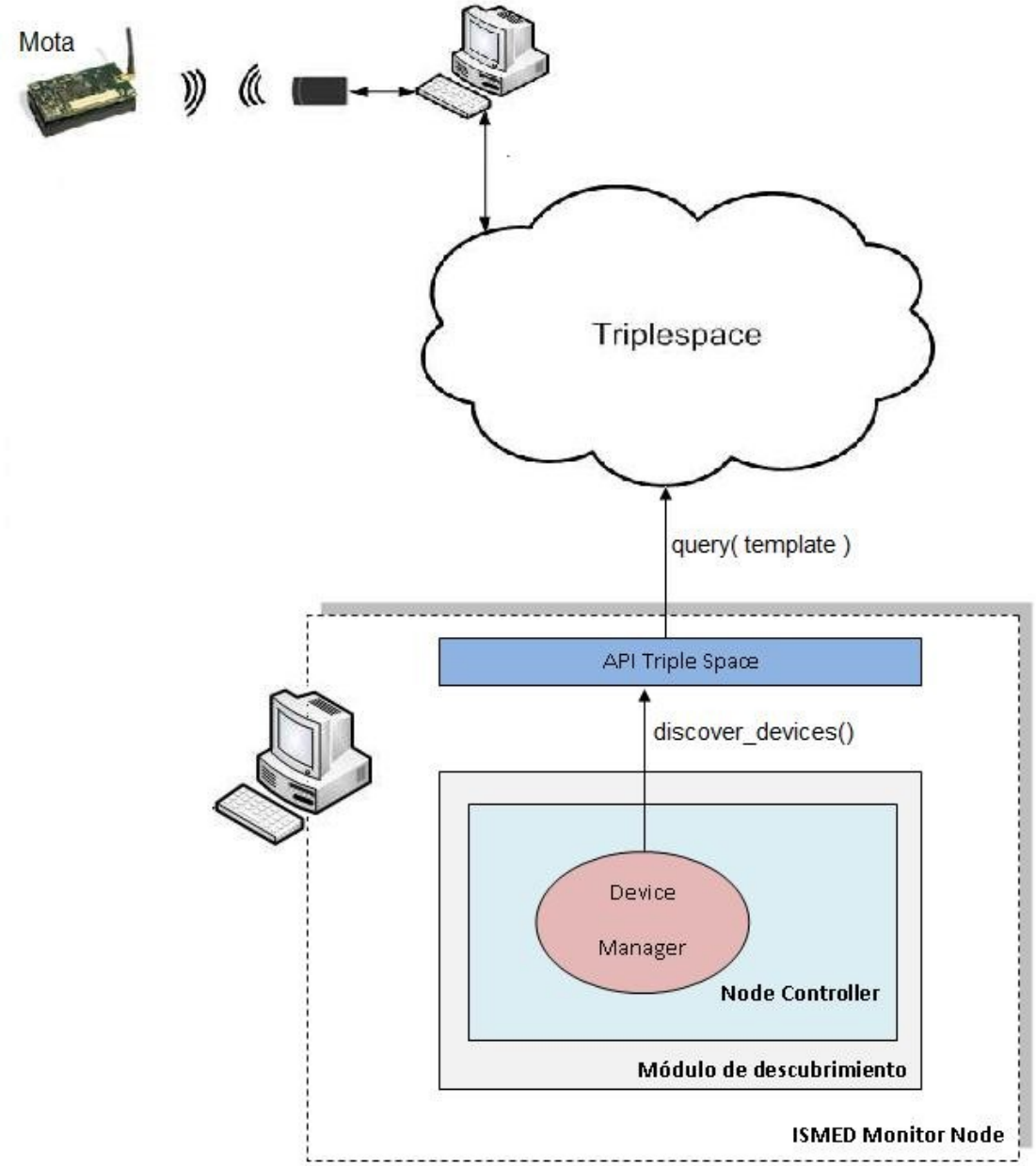


Figura 4.17.: Descubrimiento de dispositivos presentes en la red.

4.4.3.4 Des-suscripción

Este escenario se producirá cuando un dispositivo abandone la red de ISMED. En primer lugar el nodo cancelará la suscripción realizada en el momento de su llegada y en segundo lugar, con

el objetivo de notificar al resto de miembros su salida, lanzará un aviso mediante la primitiva *advertise*.

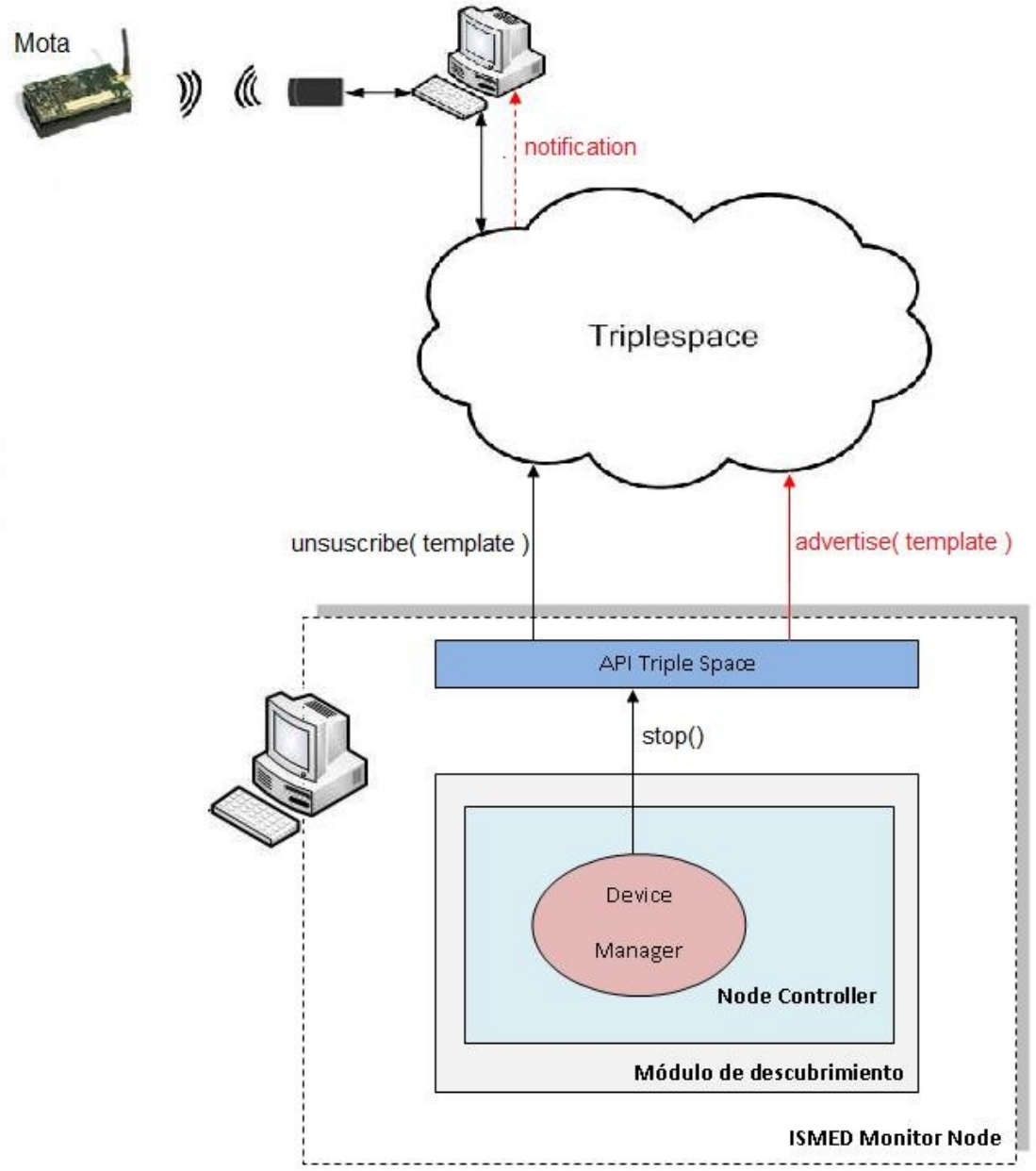


Figura 4.18.: Salida de un dispositivo de la red.

4.4.4 Desconexión

4.4.5 Capturas de pantalla de la aplicación

A continuación se presentan, en forma de capturas de pantalla, las representaciones gráficas de cada escenario propuesto en el epígrafe 4.4.3. Cabe destacar que el despliegue se ha realizado sobre un nodo que pretende realizar las funciones de monitorización de la red ISMED y por tanto la interfaz ha sido diseñada exclusivamente para el caso en cuestión.

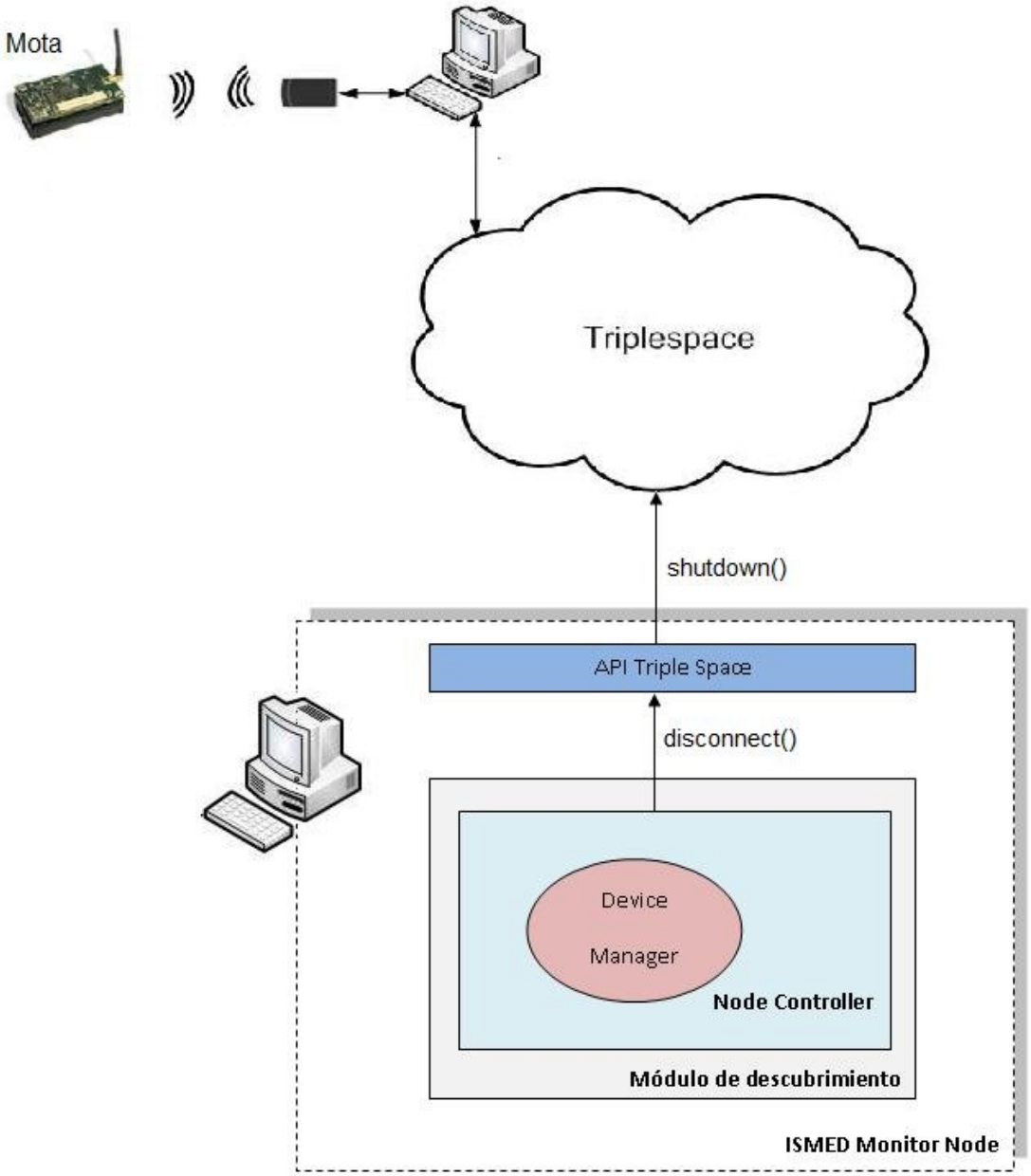


Figura 4.19.: Cierre de la conexión.

4.4.5.1 Nodo sin conexión

Véase la figura 4.20.

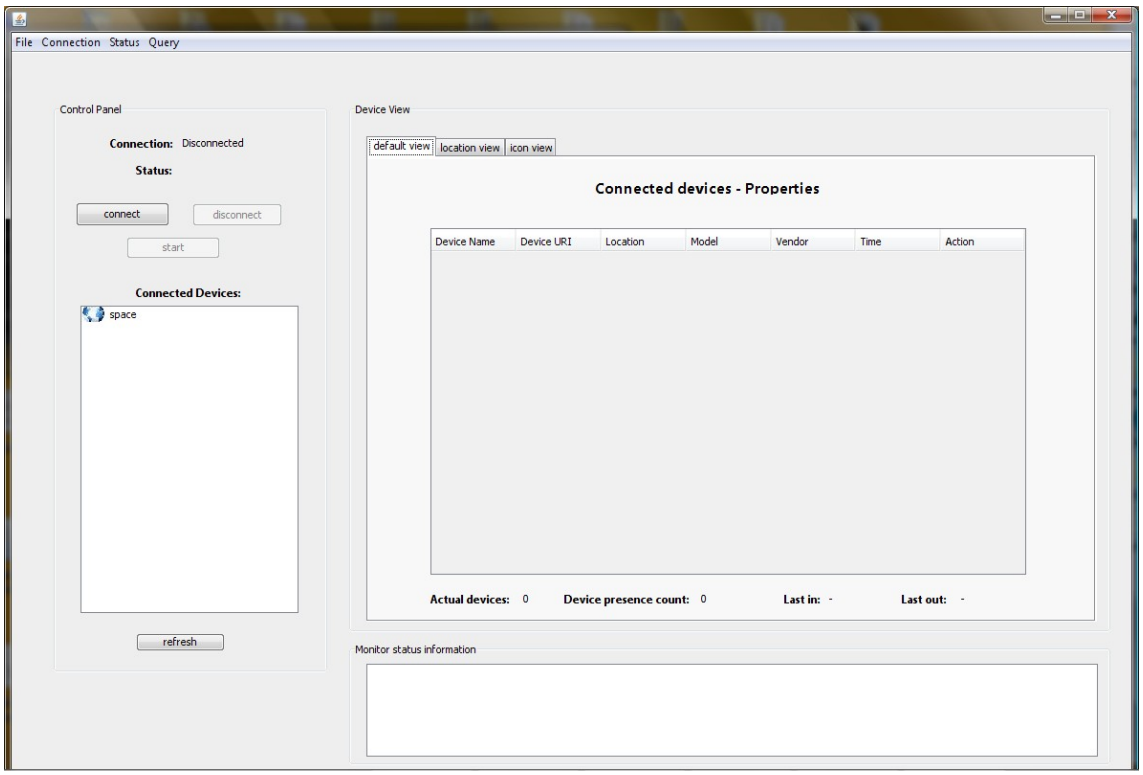


Figura 4.20.: Nodo monitor sin conexión.

4.4.5.2 Nodo conectado sin dispositivos activos

Véase la figura 4.21.

4.4.5.3 Conexión Nokia-N95

Véase la figura 4.22.

4.4.5.4 Conexión SunSPOT

Véase la figura 4.23.

4.4.5.5 Desconexión SunSpot

Véase la figura 4.24.

4.5 Razonamiento

El **razonamiento distribuido** es un objetivo muy difícilmente alcanzable, por lo que toda inferencia que se pueda realizar en el módulo de coordinación, deberá ser a nivel local.

El caso del proyecto ISMED es un caso especial debido a su complejidad y el tipo de consultas que soporta. En ISMED no existen múltiples otologías si no una única y el conocimiento está dividido entre los diferentes nodos que conforman el sistema. Es decir existe un único vocabulario común (la ontología) y existen diferentes instancias (individuos) de ese vocabulario distribuidas a través de los nodos. Atendiendo al tipo de consultas que se pueden realizar se puede determinar lo siguiente:

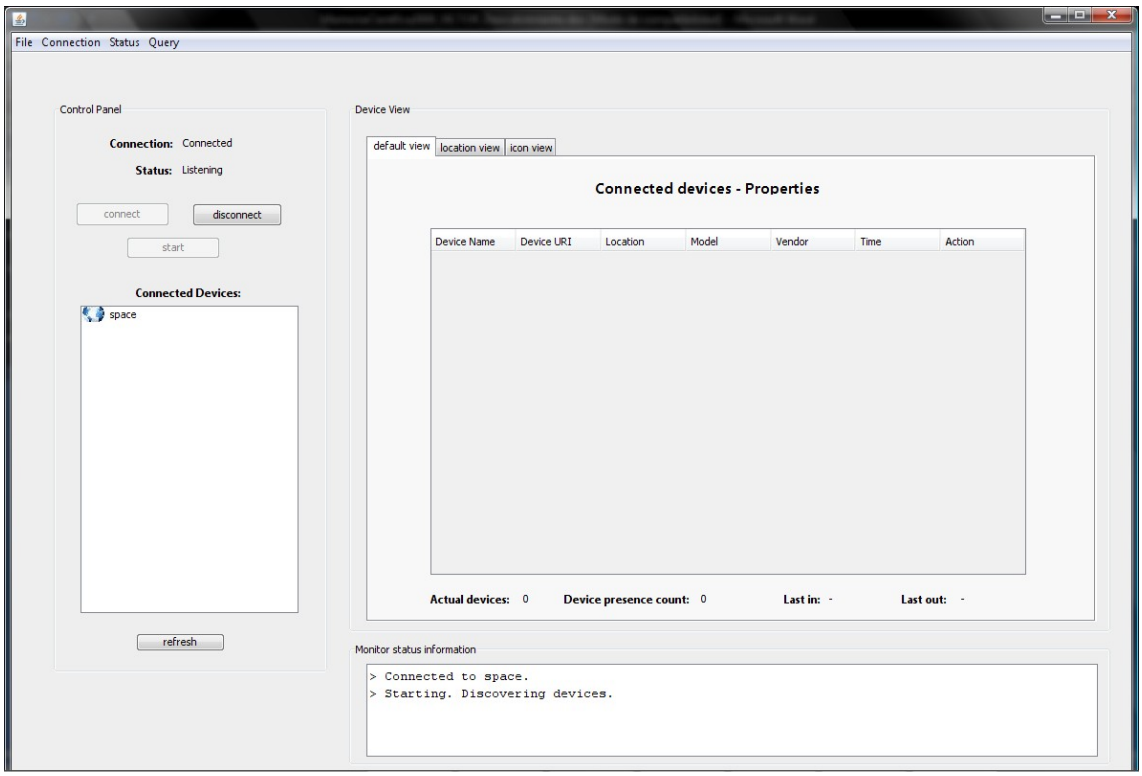


Figura 4.21.: Nodo conectado sin dispositivos activos.

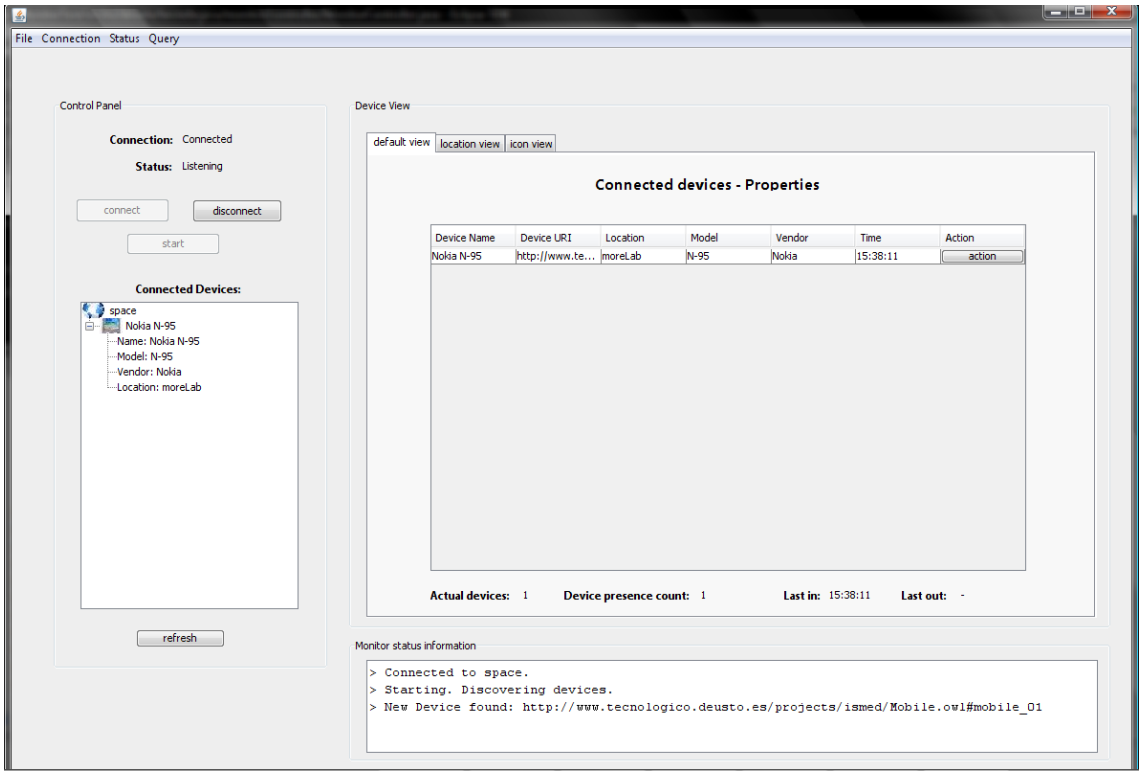


Figura 4.22.: Conexión Nokia-N95.

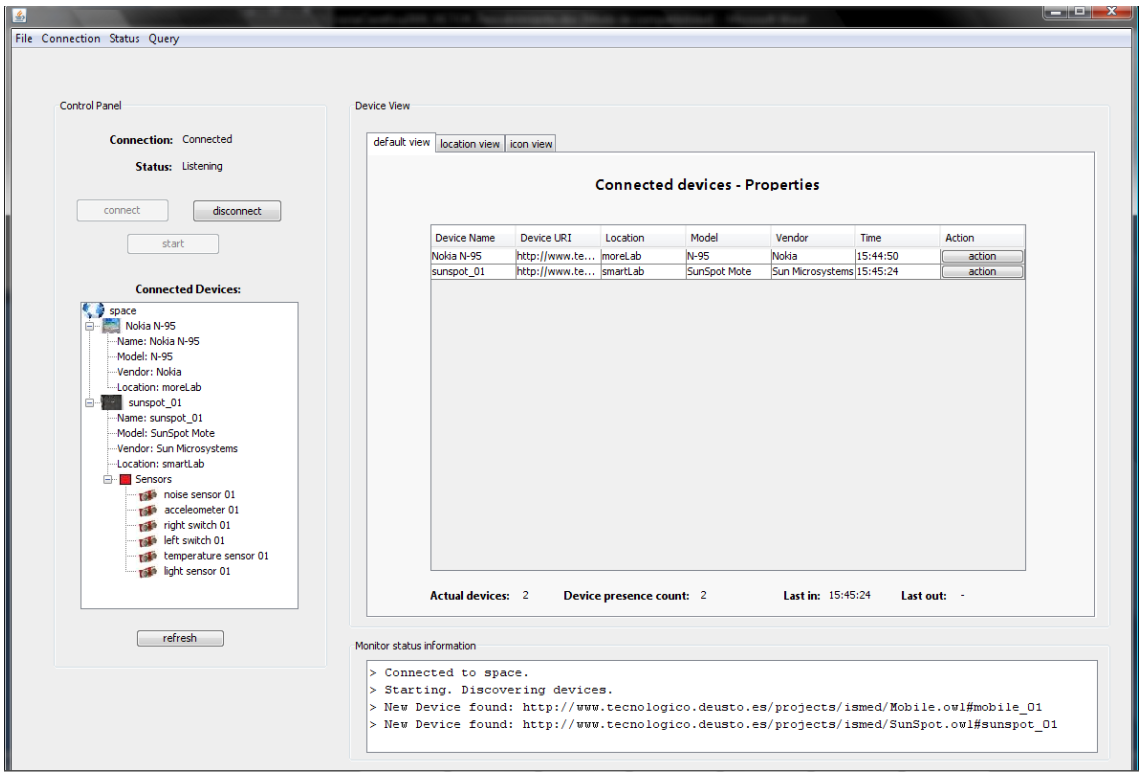


Figura 4.23.: Dispositivo móvil y SunSpot conectados.

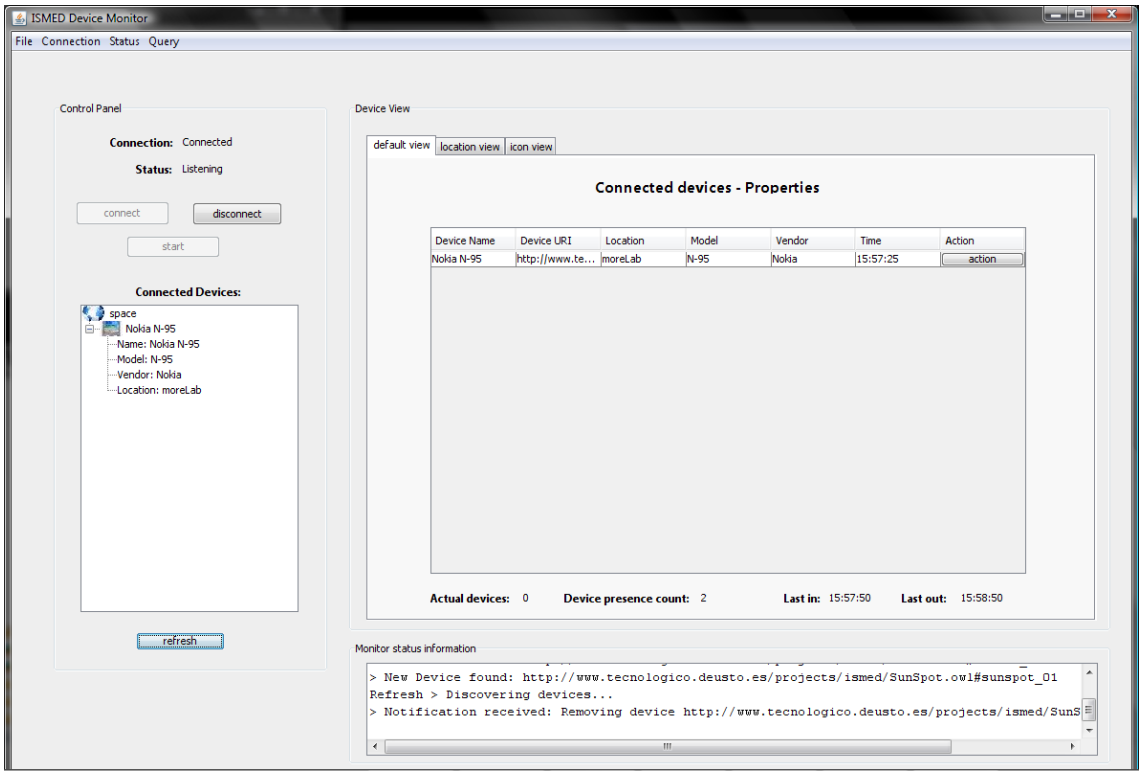


Figura 4.24.: Desconexión SunSpot

- Cuando se envía una consulta simple a un nodo y este nos responde basándonos únicamente en el conocimiento que este posee. En este caso es claro que se está hablando de "Classical Reasoning", ya que se posee una única ontología
- Las consultas múltiples que se descomponen en consultas más pequeñas que son enviadas a los distintos nodos de la red, para posteriormente filtrar las respuestas de estos y dar una respuesta final.

En este caso el grado de distribución no está claro. A pesar de que es cierto que únicamente se utiliza una única ontología, pero no está claro el grado de distribución alcanzado. Teniendo en cuenta la relación de Triple space con la computación paralela y distribuida, parece sencillo determinar que se está ante un caso de "Partitioning / Parallelization", pero esto no es cierto ya que no existe ningún planificador central que determine como realizar el razonamiento, si no que se pregunta a todos los nodos de forma indiscriminada, además de que no se tiene en cuenta ninguna cuestión de rendimiento. A su vez esta conclusión hace pensar que nos encontramos en un caso de "Distributed Reasoning", lo cual sigue sin ser verdad. Por una parte no existen las múltiples ontologías y por otra el nodo que ha recibido la consulta múltiple está realizando un razonamiento en local sobre los resultados de los nodos a los que se ha enviado las distintas subconsultas generadas a partir de la consulta múltiple. Por tanto podríamos decir que existe una distribución en cuanto a la forma de obtener el conocimiento en una query múltiple y que sobre éste se aplicará un razonamiento clásico.

En conclusión no es posible decir que en ISMED se da un razonamiento distribuido, sino más bien un razonamiento clásico, con la puntualización de que en las consultas múltiples existe una recogida del conocimiento distribuido sobre el cual se aplicará un razonamiento local.

Una estrategia posible sería consultar a distintos nodos y luego **razonar sobre los resultados obtenidos**. Esto es lo que hace la primitiva *queryMultiple* en esencia tras recibir las respuestas de cada una de los templates básicos que se extraen de la consulta SPARQL.

En la aplicación realizada para probar consultas SPARQL, se ve mejor explicado este enfoque.

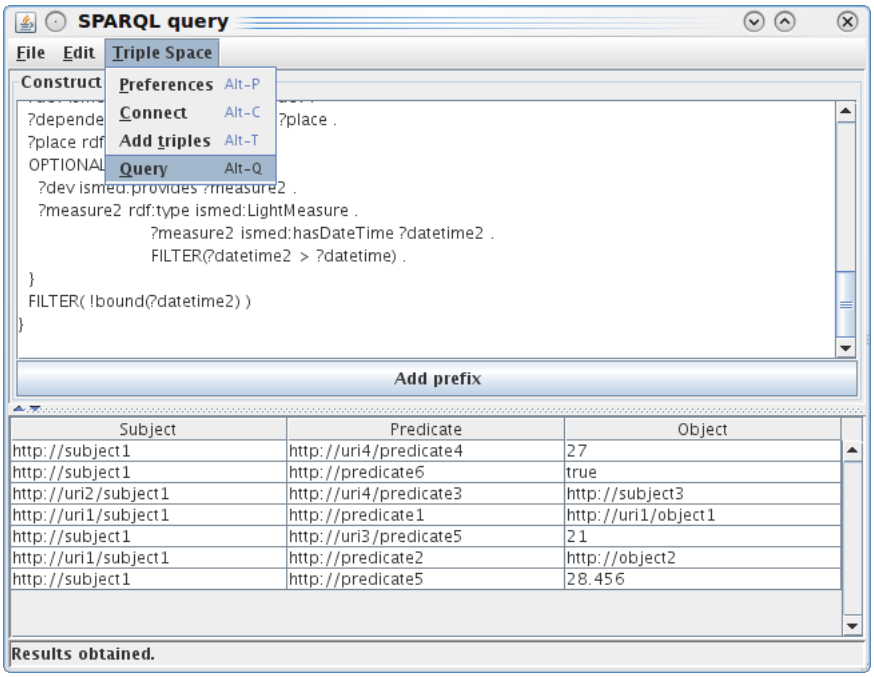


Figura 4.25.: Ventana de consulta desde la que se pueden escribir tripletas y realizar consultas de tipo queryMultiple.

Un caso práctico sería uno en el que se iniciasen tres nodos en el mismo espacio. Dos de ellos escribirían información sobre dos dispositivos distintos ubicados en dos lugares distintos en el mismo espacio (ver Ilustración 4.26).

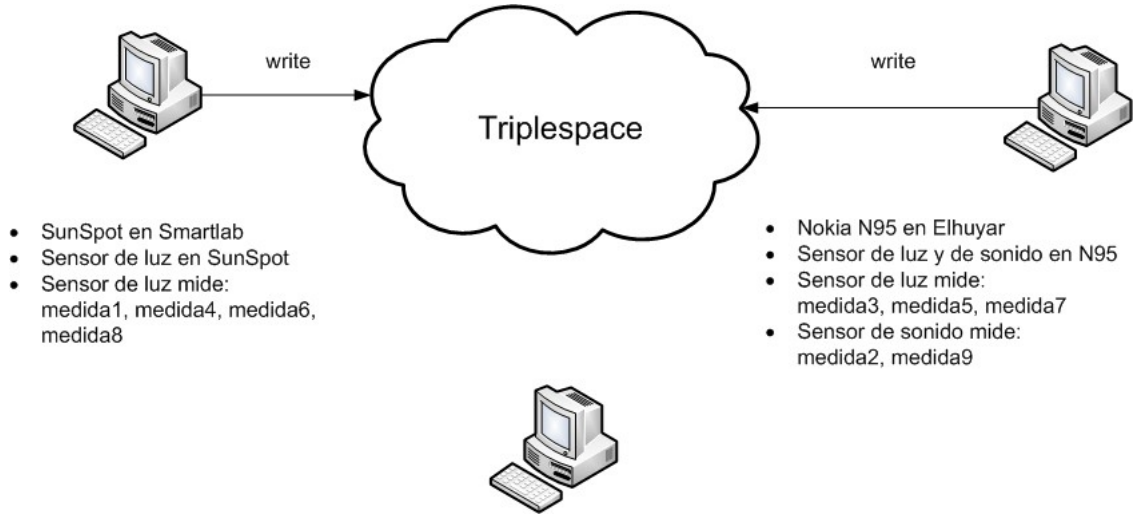


Figura 4.26.: Dos nodos escribiendo información en el mismo espacio.

Cuando el tercer nodo realizase una consulta como la que se muestra en la Tabla 4.5, sobre el espacio, esta consulta se descompondría en templates básicos (ver Tabla 4.5), que serían enviados a la vez por la red. Las respuestas serían devueltas de forma individual, y el tercer nodo esperaría un tiempo determinado recogiendo y agrupando cada una de las respuestas recibidas (ver Ilustración 4.27).

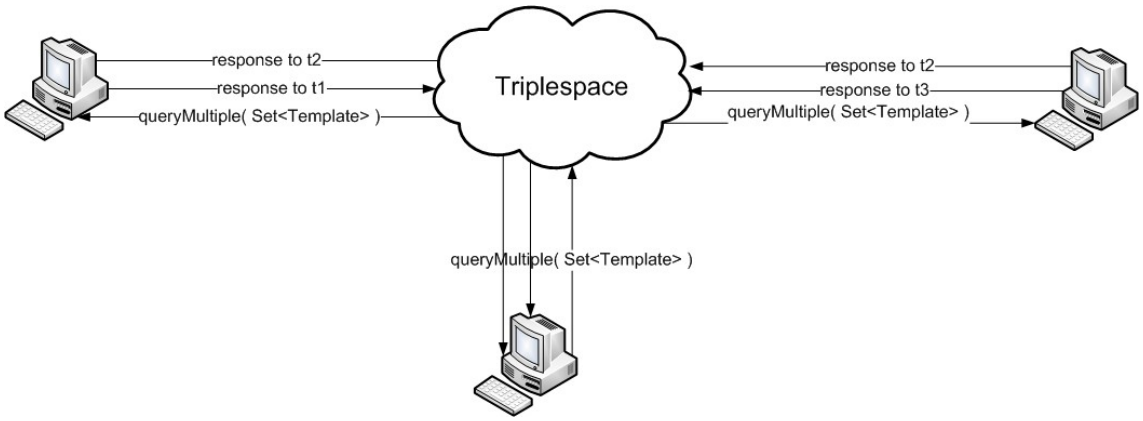


Figura 4.27.: El tercer nodo hace una consulta de tipo *queryMultiple* sobre el espacio y los dos nodos restantes le contestan.

Sobre ese conjunto de respuestas recibidas, se aplicaría de nuevo la consulta original a modo de filtro, devolviendo las respuestas concretas a la consulta original. En este último "filtro" se podría realizar inferencia sin mayor complejidad.

Este enfoque, podría usarse incluso a nivel externo del módulo de coordinación, siguiendo el esquema de componentes explicado en el epígrafe [Error: No se encuentra la fuente de referencia.]

Otra mejora posible es hacer **inferencia en cada uno de los nodos al responder a una primitiva**. Tsc++ no hace inferencia a nivel de nodo, pero los repositorios que utiliza sí que lo permiten. Aunque no de forma definitiva, se han realizado versiones de SesameDataAccess y OwlimDataAccess que razonaban a nivel local.

Desgraciadamente, Sesame (y consecuentemente Owlim) no son capaces de inferir tripletas en base a un espacio determinado (a nivel de repositorio, el espacio y el grafo se traducen en recursos,

```
CONSTRUCT {
?measure rdf:type ismed:LightMeasure .
?measure ismed:hasValue ?value .
?measure ismed:hasDateTime ?datetime .
}
WHERE {
?measure rdf:type ismed:LightMeasure .
?measure ismed:hasValue ?value .
?measure ismed:hasDateTime ?datetime .
?dev rdf:type ismed:SimpleDevice .
?dev ismed:provides ?measure .
?dev ismed:isPartOf ?dependentdev .
?dependentdev ismed:locatedIn ?place .
?place rdf:type ismed:Room .
OPTIONAL {
?dev ismed:provides ?measure2 .
?measure2 rdf:type ismed:LightMeasure .
?measure2 ismed:hasDateTime ?datetime2 .
FILTER(?datetime2 >?datetime) .
}
FILTER( !bound(?datetime2) )
}
```

Tabla 4.1.: Consulta SPARQL que muestra la última medida de luminosidad tomada por cualquier dispositivo ubicado en una habitación cualquiera.

```
?s rdf:type ismed:LightMeasure .
?s ismed:hasValue ?o .
?s ismed:hasDateTime ?o .
?s rdf:type ismed:SimpleDevice .
?s ismed:provides ?o .
?s ismed:isPartOf ?o .
?s ismed:locatedIn ?o .
?s rdf:type ismed:Room .
```

Tabla 4.2.: Templates básicos obtenidos al descomponer la consulta de la Tabla 4.5.

```
File dataDir = new File( storageFolder + /spaces");
// spacesRepository = new SailRepository(new NativeStore(dataDir));
spacesRepository = new SailRepository(
new ForwardChainingRDFSInferencer(new NativeStore(dataDir))
);
spacesRepository.initialize();
```

Tabla 4.3.: Extracto de código modificado para permitir que Sesame infiera al realizar consultas.

```
File dataDir = new File( storageFolder + /spaces");
spacesRepositories.put (
spaceuri,
new SailRepository( new ForwardChainingRDFSInferencer(new NativeStore(dataDir)))
);
spacesRepositories.get(spaceuri).initialize();
```

Tabla 4.4.: Extracto de código en el que se inserta un repositorio en el mapa creado a tal efecto.

que son agrupaciones jerárquicas lógicas de tripletas[2]) y usan como base para dicha inferencia el total de las tripletas albergadas en dicho repositorio. Para solucionar eso, en las versiones preliminares realizadas, se ha creado un nuevo repositorio por cada espacio.

[2] Así, la primitiva de org.openrdf.repository.RepositoryConnection.add() escritura en Sesame es:
void add(Iterable<Statement>arg0, Resource... arg1) throws RepositoryException
Y en SesameDataAccess se traduce a:
con.add(triples, contextSpaceURI, contextGraphURI);

5. VALIDACIÓN

En las siguientes secciones se describirá el método usado para verificar el correcto funcionamiento de los módulos desarrollados, tanto individualmente, como en conjunto a través de un escenario en el que todos entran en funcionamiento.

5.1 Definición de escenario de uso y evaluación

Existen muchos escenarios propios de ambitos como la domótica o instrumentación urbana donde el middleware propuesto en ISMED podría usarse. Uno de esos casos podría ser el control de la temperatura de una habitación. En dicho escenario, que usa las funcionalidades de todos los módulos descritos, existirán al menos 5 nodos, que se muestran en la figura 5.1.

En el escenario se pueden apreciar dos tipos de proveedores de contexto: las SunSPOTs y el proveedor de datos metereológicos. Los primeras son dispositivos físicos que comparten los datos que capturados mediante sus sensores a través de un nodo que ejerce de pasarela. El proveedor de datos metereológicos, por otro lado, es un proveedor de contexto virtual que obtiene la información necesaria de Internet, la semantiza y la escribe en un espacio determinado. Para ello, el nodo consulta sistemáticamente el espacio en busca de nuevos entornos en el espacio y comprueba si Yahoo! tiene un identificador de lugar asociado al nombre de dicho lugar. En caso positivo, consulta el servicio Yahoo! Weather [1] y obtiene la temperatura actual para dicha localización (t_e). El espacio tiene además un actuador: un aire acondicionado que decrementa la temperatura. La temperatura actual podrá ser monitorizada por el dispositivo móvil.

El nodo regulador usa toda la información obtenida por los sensores cuando se detecta la presencia de alguien en un lugar concreto para regular la temperatura de dicho lugar. Para inferir dicha presencia, se tiene en cuenta cuando el dispositivo móvil que pertenece a alguien se encuentra en el espacio. Para llevar a cabo la regulación de la temperatura, el nodo regulador comprueba si existe algún sistema de aire acondicionado en el lugar que desee regular y de ser así lo activa.

Para llevar a cabo dicha regulación, se ha tenido en cuenta la normativa que dicta que la temperatura dentro de los edificios (t_i) debe ser menor a 26 °C cuando el aire acondicionado está encendido y otra que regula que la temperatura de una oficina debe de estar entre 17 °C y 27 °C

Algoritmo de control básico de la temperatura interior

```
mientras ti<26
  si te<26: el usuario debería abrir las ventanas
  en caso contrario: se enciende el aire acondicionado
```

Así, el nodo gestor de SunSPOT reclama (*demand*) un *template* relacionado con los LEDs de las SunSPOT que controla y el nodo responsable del aire acondicionado hace la URI que lo identifica en la ontología descrita en la figura 5.1. Cuando el nodo regulador decide reducir la temperatura de la habitación, tratará de escribir dicha información, pero dado que tanto los nodos responsables de la SunSPOT y del aire acondicionado han dicho ser responsables de dicho conocimiento la primitiva de escritura se convertirá en una primitiva *suggest*. A partir de ahí, ambos nodos decidirán qué hacer con dicha información. En el ejemplo, el responsable de las SunSPOT encenderá o apagará los LEDs de todas ellas usando el gateway, de forma que su estatus será automáticamente actualizado y el aire acondicionado activará o desactivará el mismo y actualizará su estatus.

Además de decidir a qué temperatura poner la habitación, el sistema, basándose en el módulo de aprendizaje, se adaptará a las costumbres que tiene el usuario sobre cuando encender o apagar el aire acondicionado. Así, este sistema descubrirá con que otras acciones está relacionado la interacción del usuario con el aire acondicionado y proveerá una valiosa información al sistema para anticipar

[1] http://developer.yahoo.com/weather/

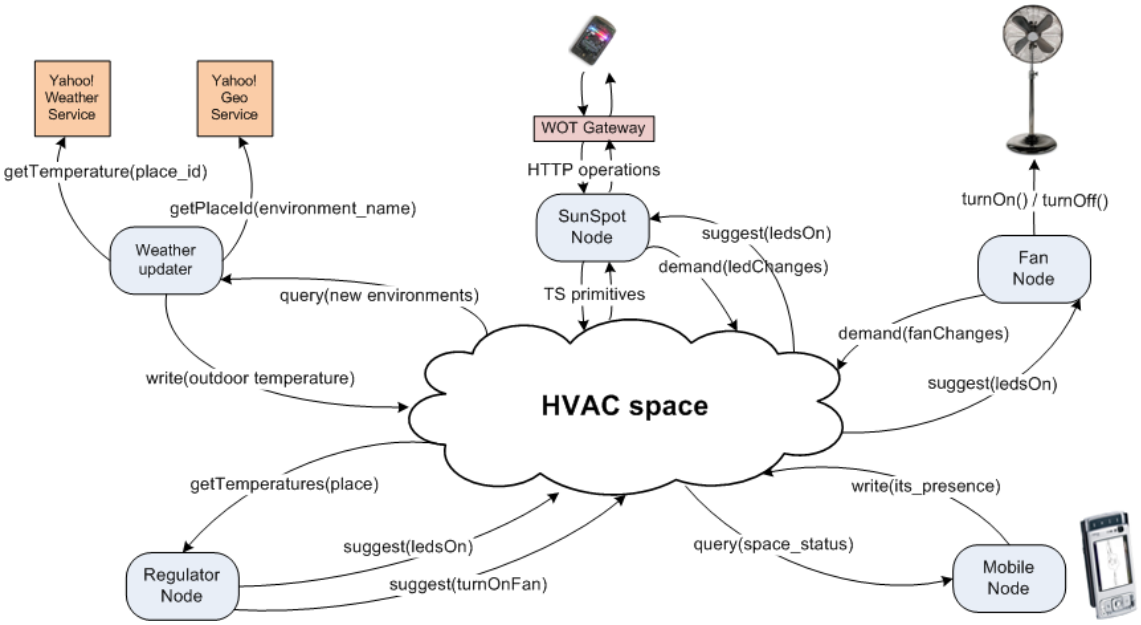


Figura 5.1.: Implementation of the described scenario using resource based approach.

a las necesidades del usuario en el futuro. Además, la contextualización del comportamiento también permitirá identificar en base a qué variables del entorno (temperatura, humedad,...) actúa el usuario de una forma u otra.

5.2 Modelado y coordinación semántica

Para verificar el correcto funcionamiento de las funcionalidades implementadas en el módulo de modelado y coordinación se han realizado test unitarios sobre la mayoría de las clases desarrolladas. No obstante, para verificar el correcto comportamiento del módulo, se hace necesario desplegar múltiples nodos con distinta información y comprobar que cada primitiva tiene el comportamiento que se espera de ella.

Para dichos tests, centrados en verificar la parte de comunicación del módulo de coordinación, se ha usado la aplicación monitora descrita en 4.1 con los datos que se definirán en la sección 5.2.1 y los casos de uso definidos en 5.2.2.

5.2.1 Información necesaria para la validación

En la sección 5.2.2 se hará referencia a las tripletas que se muestran a continuación a fin de aportar claridad a la definición de los casos de uso en dicha sección.

Por claridad las tripletas se expresarán en formato *turtle*[2]. En el siguiente extracto se expresarán todos los prefijos utilizados en el resto de fragmentos, donde se han obviado para que no sean repetitivos. En los siguientes simplemente se facilitará el nombre con el que se hará referencia a ellos en los casos de uso.

```
@prefix : <http://www.tecnologico.deusto.es/projects/ismed/sample.owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@base <http://www.tecnologico.deusto.es/projects/ismed/sample.owl> .
```

[2] http://www.w3.org/TeamSubmission/turtle/

GrafoQA

```
:subject1 :predicate1 :object1 ;
          :predicate2 :object2 ;
          :predicate3 :object3 ,
                      :subject3 .

:subject2 :predicate2 :object2 .

:subject3 :predicate3 :object3 ;
          :predicate1 :subject1 .
```

GrafoQB

```
:subject1 :predicate5 "21"^^xsd:int ,
                      "27"^^xsd:int ;

          :predicate4 "28.456"^^xsd:double ;

          :predicate6 "true"^^xsd:boolean .
```

GrafoRA

```
:subject2 :predicate1 :object1 ;
          :predicate2 :object2 .

:subject3 :predicate3 :object3 ;
          :predicate2 :object2 ;
          :predicate1 :object1 .
```

GrafoRB

```
:subject4 :predicate1 :object1 ;
          :predicate2 :object2 ;
          :predicate3 :object1 ;
          :predicate4 :object4 ;
          :predicate3 :object5 .

:subject5 :predicate3 :object5 .
```

GrafoRC

```
:subject5 :predicate5 :object5 ;
          :predicate3 :object1 ;
          :predicate3 :object2 ;
          :predicate4 :object4 .

:subject3 :predicate5 :object5 .
```

GrafoTA

```
:subject1 :predicate1 :object1 ;
          :predicate2 :object2 ;
          :predicate3 :object3 .
```

```
:subject2 :predicate2 :object2 ;
          :predicate3 :object3 .
```

GrafoTB

```
:subject3 :predicate2 :object2 .

:subject4 :predicate3 :object1 ;
          :predicate4 :object4 .

:subject5 :predicate3 :object5 .
```

GrafoTC

```
:subject6 :predicate3 :object2 ;
          :predicate5 :object4 ;
          :predicate6 :object6 .
```

GrafoEQ1

```
:subject1 :predicate1 :object1 ;
          :predicate2 :object2 ;
          :predicate3 :object3 ,
                      :subject3 ;
          :predicate5 "21"^^xsd:int ,
                      "27"^^xsd:int ;
          :predicate4 "28.456"^^xsd:double ;
          :predicate6 "true"^^xsd:boolean .
```

GrafoEQ2

```
:subject1 :predicate2 :object2 .
```

GrafoEQ3

```
:subject1 :predicate3 :object3 .
```

GrafoEQ4

```
:subject1 :predicate6 "true"^^xsd:boolean .
```

GrafoQMA

```
###  Definición de clases
:sensorhumedad1 rdf:type ismed:HumiditySensor .
:sensorluz1 rdf:type ismed:LightSensor .
:sensorluz2 rdf:type ismed:LightSensor .
:sensortemperatura1 rdf:type ismed:TemperatureSensor .
:sensortemperatura2 rdf:type ismed:TemperatureSensor .
:smartlab rdf:type ismed:Room .
:aula1 rdf:type ismed:Room .
:aula2 rdf:type ismed:Room .
```

GrafoQMB

```
### Definición del resto de individuos
:sensor2 rdf:type ismed:XBee ;
         ismed:macAddress "00:00:00:00:01"^^xsd:string ;
         ismed:name "Sensor2"^^xsd:string ;
         ismed:locatedIn :aula1 ;
         ismed:madeUpOf :sensorluz1 ,
                        :sensortemperatura1 .
```

GrafoQMC

```
:sensor3 rdf:type ismed:XBee ;
         ismed:macAddress "00:00:00:00:02"^^xsd:string ;
         ismed:name "Sensor3"^^xsd:string ;
         ismed:locatedIn :aula2 ;
         ismed:madeUpOf :sensorhumedad1 ,
                        :sensorluz2 ,
                        :sensortemperatura2 .
```

GrafoEQM

```
integ:sensor3 ismed:name "Sensor3"^^xsd:string .
```

5.2.2 Casos de uso para la validación de primitivas

Para tratar de comprobar que todas las primitivas de Triple Space funcionan en un escenario con varios nodos, se han realizado unas pruebas que se describen a continuación verificando que en toda ellas se ha producido el comportamiento esperado.

5.2.2.1 No pertenencia a un espacio

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio2* (create+join).
3. El *nodo a* escribe *GrafoTA* en *espacio1* (write).
4. El *nodo b* realiza la consulta *?s ?p ?o .* sobre *espacio2* (query).

Resultados esperados:

- El *nodo b* no obtiene respuesta alguna al realizar la consulta sobre un espacio en el que nadie ha escrito nada previamente.

5.2.2.2 Unión a un espacio

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio1* (create+join).
3. El *nodo a* escribe *GrafoTA* en *espacio1* (write).
4. El *nodo b* realiza la consulta *?s ?p ?o .* sobre *espacio1* (query).

Resultados esperados:

- El *nodo b* obtiene las tripletas de *GrafoTA* como respuesta a su consulta.

5.2.2.3 Salida de un espacio

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio1* (create+join).
3. El *nodo a* escribe *GrafoTA* en *espacio1* (write).
4. El *nodo a* abandona el *espacio1* (leave).
5. El *nodo b* realiza la consulta *?s ?p ?o .* sobre *espacio1* (query).

Resultados esperados:

- El *nodo b* no obtiene respuesta alguna al realizar la consulta sobre un espacio al que sólo el pertenece en ese momento.

5.2.2.4 query

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio1* (create+join).
3. El *nodo a* escribe *GrafoQA* en *espacio1* (write).
4. El *nodo a* escribe *GrafoQB* en *espacio1* (write).
5. El *nodo b* realiza la consulta *:subject1 ?p ?o .* sobre *espacio1* (query).
6. El *nodo b* realiza la consulta *:subject1 :predicate2 ?o .* sobre *espacio1* (query).
7. El *nodo b* realiza la consulta *:subject1 ?p :object3 .* sobre *espacio1* (query).
8. El *nodo b* realiza la consulta *?s :predicate6 ?o .* sobre *espacio1* (query).

Resultados esperados:

- En la primera query, el *nodo b* obtiene las tripletas de *GrafoEQ1* como respuesta a su consulta.
- En la segunda query, el *nodo b* obtiene las tripletas de *GrafoEQ2* como respuesta a su consulta.
- En la tercera query, el *nodo b* obtiene las tripletas de *GrafoEQ3* como respuesta a su consulta.
- En la cuarta query, el *nodo b* obtiene las tripletas de *GrafoEQ4* como respuesta a su consulta.

5.2.2.5 read

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio1* (create+join).
3. El *nodo a* escribe *GrafoRA* en *espacio1* (write).
4. El *nodo a* escribe *GrafoRB* en *espacio1* (write).
5. El *nodo a* escribe *GrafoRC* en *espacio1* (write).
6. El *nodo b* realiza la consulta *:subject4 :predicate4 :object4 .* sobre *espacio1* (read).

Resultados esperados:

- En *nodo b* obtiene las tripletas de *GrafoRB* como respuesta a su consulta.

5.2.2.6 take

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio1* (create+join).
3. El *nodo a* escribe *GrafoTA* en *espacio1* (write).
4. El *nodo a* escribe *GrafoTB* en *espacio1* (write).
5. El *nodo a* escribe *GrafoTC* en *espacio1* (write).
6. El *nodo b* realiza la consulta *:subject3 ?p ?o .* sobre *espacio1* (take).

Resultados esperados:

- En *nodo b* obtiene las tripletas de *GrafoTB* como respuesta a su consulta.

5.2.2.7 queryMultiple

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio1* (create+join).
3. El *nodo c* se conecta al *espacio1* (create+join).
4. El *nodo d* se conecta al *espacio1* (create+join).
5. El *nodo a* escribe *GrafoQMA* en *espacio1* (write).
6. El *nodo b* escribe *GrafoQMB* en *espacio1* (write).
7. El *nodo c* escribe *GrafoQMC* en *espacio1* (write).
8. El *nodo d* realiza la consulta SPARQL descrita a continuación sobre *espacio1* (queryMultiple).

Resultados esperados:

- En *nodo d* obtiene los resultados parciales desde *nodo a*, *nodo b* y *c GrafoTB* y los filtra obteniendo la tripleta descrita en *GrafoEQM*.

Consulta SPARQL a realizar

```
 CONSTRUCT {
?mota ismed:name ?name .
 } WHERE {
?mota rdf:type ismed:XBee .
?mota ismed:name ?name .
?mota ismed:madeUpOf ?sensor .
?sensor rdf:type ismed:HumiditySensor .
?mota ismed:locatedIn integ:aula2 .
 }
```

5.2.2.8 Invocación de servicios

Pasos a realizar:

1. Implementar servicio de suma que busca dos parámetros numéricos de entrada y escribe uno de salida. Para ello habrá que seguir las interfaces descritas en la sección 3.1.2.1. A este servicio le llamaremos *servsuma*.
2. El *nodo a* se conecta al *espacio1* (create+join).
3. El *nodo b* se conecta al *espacio1* (create+join).
4. El *nodo c* se conecta al *espacio1* (create+join).
5. El *nodo a* escribe la definición de *servsuma* en el espacio (define).
6. El *nodo b* construye una invoca el servicio *servsuma* facilitando los dos datos de entrada (número 2 y 3) en *espacio1* y creando un aviso de dicha acción (invoke).

Resultados esperados:

- El *nodo a* recibe la invocación del servicio descrito. Obtiene los datos de la invocación del espacio (take), ejecuta el servicio y escribe el resultado (5) en el *espacio1* avisando de dicha acción (write+advertise).
- El *nodo b* recibe el aviso de que el resultado de su invocación ha sido llevado a cabo, por lo que lo leerá del *espacio1* (take).

5.2.2.9 Cambio del entorno a través de demand

Pasos a realizar:

1. El *nodo a* se conecta al *espacio1* (create+join).
2. El *nodo b* se conecta al *espacio1* (create+join).
3. El *nodo c* se conecta al *espacio1* (create+join).
4. El *nodo a* reclama su responsabilidad sobre *:subject3 ?p ?o .* en *espacio1* (demand).
5. El *nodo b* trata de escribir *GrafoTB* en *espacio1* (write).

Resultados esperados:

- El intento de escritura del *nodo b* se transforma en una sugerencia (suggest) de cambio que *nodo a* procesa pero *nodo c* no.

5.3 Aprendizaje

El módulo de aprendizaje también ha sido validado utilizando datos creados de forma artificial así con datos recolectados en entornos reales. Además del entorno creado en MGEP, los otros dos entornos reales (MavPad y WSU Smart Apartment).

Para validar el módulo de aprendizaje en el entorno de MGEP un estudiante interactuó con el entorno durante 1 mes, recogiendo el sistema las acciones que realizaba así como las variables del entorno. La identificación de las acciones realizadas por el usuario fue posible gracias al desarrollo de un guante que mediante tecnologías como ZigBee y RFID identificaba con qué objetos estaba interaccionando el usuario. Para la recogida de los datos del entorno se utilizaron los sensores de temperatura, humedad y luminosidad adquiridas durante este proyecto. Una vez recogido los datos, se ejecutó el sistema de aprendizaje siendo identificados varios patrones del usuario.

Teniendo en cuenta la aplicación propuesta en el escenario mencionado anteriormente, el módulo de aprendizaje fue capaz de determinar en comportamiento habitual del usuario en lo que se refiere a la temperatura de la habitación donde se instalaron los sensores. Para ello, además de los sensores

en la misma habitación se instalaron sensores de temperatura y humedad en el exterior para poder contextualizar el comportamiento del usuario en base a esa información.

En realidad se llevaron a cabo dos tipos de validaciones. En la primera el colaborador actuaba de forma totalmente libre, es decir, sin ningún comportamiento definido a-priori. En esta primera validación, debido a la naturaleza del comportamiento, fue necesario bajar el nivel de confidencia (hasta el 25 %) para poder detectar comportamientos donde se incluyesen acciones sobre la temperatura del entorno.

En la segunda validación, el colaborador actuó de forma predefinida para validar si el sistema era capaz de detectar dichos comportamientos y de forma (más que nada la topología) correcta. En este sentido, y considerando un nivel de confidencia del 75 %, el sistema fue capaz de detectar los patrones de comportamiento del usuario.

El módulo de aprendizaje también fue validado en otros entornos reales. Uno de ellos fue el entorno de Mavpad [881] que cuenta con:

- 25 sensores en objetos tales como lámparas, enchufes,...
- 45 sensores de entorno (temperatura, humedad,...)
- 36 sensores de movimiento instalados a lo largo del entorno.

La característica específica del conjunto de datos recogido en este entorno es que los usuarios de este entorno no tenían ningún comportamiento preconcebido por lo que la tarea de identificación de patrones simula de forma real su ejecución en un futuro. En este entorno se recolectaron datos en tres diferentes periodos de tiempo (Trial 1, Trial 2 y Trial 3).

En lo que se refiere al entorno de WSU Smart Apartment, este entorno cuenta con:

- 14 sensores instalados en objetos.
- 25 sensores de movimiento para detectar la ubicación concreta del usuario en cada momento.

La mayor diferencia de este conjunto de datos era que los comportamientos de los usuarios de este entorno estaban predefinidos de modo que se sabía de antemano los patrones que el módulo de aprendizaje debería de descubrir. En este caso, el sistema de aprendizaje fue capaz de descubrir dichos comportamientos demostrando su capacidad para descubrir comportamientos frecuentes de los usuarios. Los diferentes usuarios realizaban las siguientes actividades:

- Realizar una llamada de teléfono para recoger instrucciones para cocinar.
- Lavar las manos.
- Cocinar de acuerdo a las instrucciones recogidas en la llamada.
- Comer lo cocinado.
- Lavar los platos.

5.3.0.10 MavPad environment

Como se ha indicado anteriormente en el entorno de MavPad se recogieron datos en tres periodos de tiempo. Así, el módulo de aprendizaje fue validada utilizando estos tres datasets de forma independiente.

Para el primer paso, el de descubrimiento de conjuntos de acciones frecuentes, se han llevado a cabo cuatro diferentes tests por cada dataset, considerando diferentes niveles de confidencia (25 %, 50 %, 75 % y 100 %). A continuación se muestran el número de conjuntos de acciones que han sido descubiertos, así como el tiempo de ejecución.

Una vez que los conjuntos de acciones fueron identificados, el siguiente paso fue identificar la topología por cada conjunto de acciones. En ese sentido, fue posible identificar una topología por cada conjunto de acciones. La identificación de relaciones temporales fue una de los pasos más críticos teniendo en cuenta que no existía ningún conocimiento a-priori sobre los datos. La tabla 5.2 resume el número de patrones (en porcentaje) donde ha sido posible identificar relaciones temporales cuantitativas.

Tabla 5.1.: Número de patrones descubiertos en cada trial y tiempo de ejecución (en milisegundos).

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Total Patterns**	**Total Patterns**	**Total Patterns**
25 %	8 (156 ms)	3 (219 ms)	1 (153 ms)
50 %	4 (78 ms)	1 (188 ms)	1 (68 ms)
75 %	1 (62 ms)	1 (93 ms)	1 (56 ms)
100 %	0 (35 ms)	0 (35 ms)	0 (35 ms)

Tabla 5.2.: Número de patrones donde ha sido posible identificar relaciones temporales cuantitativas y el tiempo de ejecución.

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Action Patterns with Time Relations**	**Action Patterns with Time Relations**	**Action Patterns with Time Relations**
25 %	48 (48/56 (86 %)) (844 ms)	43 (43/56 (77 %)) (2071 ms)	No Time Relations needed
50 %	18 (18/22 (82 %)) (437 ms)	5 (5/7 (71 %)) (1651 ms)	No Time Relations needed
75 %	No Time Relations needed	5 (5/7 (71 %)) (578 ms)	No Time Relations needed
100 %	No Time Relations needed	No Time Relations needed	No Time Relations needed

Finalmente, fue posible identificar las condiciones necesarias para contextualizar de forma precisa cada patrón. En lo que se refiere a las condiciones generales, cabe mencionar que fue posible identificar condiciones generales en el 100 % de los patrones. Sobre las condiciones específicas cabe destacar que la naturaleza del entorno así como el comportamiento de los usuarios, dificultó el descubrimiento de dichas condiciones. Aún así, fue posible descubrir condiciones específicas en la mayoría de los patrones, tal como muestra la Tabla 5.3.

5.3.0.11 WSU Smart Apartment environment

Como se ha mencionado anteriormente, el conjunto de datos recogidos en el entorno de WSU Smart Apartment contiene un comportamiento que se sabía de antemano. Así, la validación del módulo de aprendizaje con este dataset fue una prueba de fuego para el módulo, y sobre todo para algunos de los pasos (por ejemplo para el de descubrimiento de topología).

El módulo de aprendizaje fue capaz de descubrir el conjunto de acciones que los usuarios llevaban a cabo. Así, 21 acciones fueron identificadas como parte del conjunto de acciones. La validación de

Tabla 5.3.: Número de patrones (en porcentaje) donde ha sido posible descubrir condiciones específicos y el tiempo de ejecución.

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Action Patterns with Conditions**	**Action Patterns with Conditions**	**Action Patterns with Conditions**
25 %	7 (7/10 (70 %)) (500 ms)	6 (6/9 (67 %)) (1015 ms)	No Conditions needed
50 %	2 (2/3 (67 %)) (188 ms)	1 (1/2 (50 %)) (1062 ms)	No Conditions needed
75 %	No Conditions needed	1 (1/2 (50 %)) (156 ms)	No Conditions needed
100 %	No Conditions needed	No Conditions needed	No Conditions needed

este conjunto de datos fue una prueba de fuego para el paso de descubrimiento de topología, ya que se sabía de antemano en qué orden se llevaban a cabo las acciones. La Tabla 5.4.

Tabla 5.4.: Acciones involucrados en cada una de las actividades y su orden.

Activity	**Involved Actions**
Make a phone call	'PhoneBook On'–>'Phone On'–>'Phone Off'
Wash hands	'Water On'–>'Water Off'
Cook	'Cabinet On'–>'Raisins On'–>'Oatmeal On'–>'MeasuringSpoon On'–>'Bowl On'–>'Sugar On'–>'Cabinet Off'–>'Water On'–>'Water Off'–>'Pot On'–>'Burner On'–>'Burner Off'
Eat	'Cabinet On'–>'Medicine On'–>'Cabinet Off'–>'Water On'–>'Water Off'–>'Cabinet On'–>'Medicine Off'–>'Cabinet Off'
Clean	'Water On'–>'Water Off'

El módulo de aprendizaje fue capaz de identificar la topología para este conjunto de acciones. Además fue capaz de identificar acciones repetitivas, tales como "Water On.º Çabinet On", y también subconjuntos de acciones no-ordenadas. Cabe mencionar que estas características especiales se conocían a-priori, y que el módulo de aprendizaje fue capaz de descubrir esas particularidades.

Sobre las relaciones temporales cuantitativas, fue posible identificar relaciones cuantitativas en un

86 % de casos, mientras que en un 67 % de los casos fue posible identificar condiciones específicas.

5.4 Descubrimiento de recursos

El descrubrimiento en la solución de coordinación basada en un Triple Space distribuido propuesta se obtiene por el simple hecho de estar conectado a un espacio concreto. Por tanto, la propia validación llevada a cabo en el módulo de modelado y coordinación 5.2 valida los aspectos relativos al descubrimiento de conocimiento en el entorno.

Este conocimiento puede modelar aspectos relativos a datos recogidos por sensores, a dispositivos de naturaleza heterogénea (móviles, ordenadores, sensores, actuadores, etc.) y/o servicios. Para ello, sólo hace falta definirlos en la ontología que se use en conjunción con el módulo de modelado y coordinación (en nuestro caso, hemos definido una ontología ya recoge parte de esa información tal y cómo se ha descrito en 3.1.5). En la implementación descrita en 4.4 se ha explicado cómo se realizó un escenario que muestra específicamente la capacidad de cada nodo para llevar a cabo dichas tareas de búsqueda de elementos en la red, representando los dispositivos físicos que se van encontrando en el entorno.

5.5 Razonamiento

Cómo se ha mencionado en otros epígrafes, el típo de razonamiento que se llevará a cabo en el middleware diseñado es aquel que provea la capa de acceso a datos del nodo concreto que procese una consulta. En concreto, a día de hoy, esto se traduce en:

- tsc++ modificado: usa sesame u owlim depende de cómo se configure. Estos a su vez permiten razonamiento a nivel de RDFS y un dialecto de OWL cercano a OWL lite respectivamente.
- tscME: no permite razonar, sólo almacenar tripletas escritas. Esta es una limitación sobre la que se seguirá trabajando en sucesivos proyectos.

Por lo tanto, el razonamiento realizado por cada nodo concreto ya ha sido validado en el momento en el que se usa una librería externa.

En el módulo de coordinación semántica no se realiza razonamiento distribuido, siendo lo más parecido a ello la consulta distribuída *queryMultiple*, ya validada en la sección 5.2.

5.6 Test de integración

Para evaluar el rendimiento de la integración de los diferentes desarrollos llevados a cabo durante el proyecto, se han realizado una serie de evaluaciones descritas a continuación.

5.6.1 tscME

Para evaluar el rendimiento de las diferentes primitivas implementadas por tscME, se ha recreado un escenario en el que 5, 10 y 20 teléfonos se unen a 1, 5 y 10 espacios. Cada emulador, que dispone de un almacenamiento de 64MB, almacena 50 grafos de 5 tripletas cada uno repartidos a lo largo de todos los espacios. Como puede verse en la tabla 5.5, al realizar las diferentes mediciones se puede apreciar que lleva mayor tiempo parsear las tripletas, que esperar las respuestas. Al realizar el test en un dispositivo móvil real, los tiempos se incrementan entre 2 y 3 segundos.

Por otra parte, se ha medido el rendimiento de los dos tipos de repositorio de conocimiento de cada dispositivo: persistente y no-persistente. Como puede observarse en la tabla 5.6, en esta ocasión cada operación ha sido medida con 10, 50 o 100 grafos de 10 tripletas cada uno.

En una primera evaluación de los resultados, se puede observar que el rendimiento es bueno cuando las tripletas están almacenadas en el Record Store. La implementación en memoria muestra un buen rendimiento en todos los casos, la primitiva *query* es especialmente rápida en el caso de los 100 grafos, ya que las tripletas, además de estar almacenadas en los grafos a los que pertenecen, se encuentran en un grafo común utilizado para optimizar.

Kernels	5			10			20		
Spaces	1	5	10	1	5	10	1	5	10
read	0.23	0.22	0.26	3.48	2.99	3.04	10.01	9.97	9.8
take	0.2	0.21	0.28	3.40	2.89	2.61	10.28	9.93	11.07
query	0.4	0.27	0.24	7.05	3.66	3.34	24.83	11.9	10.56

Tabla 5.5.: Resultados de la evaluación de TscME (en segundos)

Número de grafos		10	50	100
RecordStore	write	0.006	0.006	0.006
	read	0.127	0.486	1.381
	take	0.125	0.473	1.401
	query	0.226	0.934	3.001
Memory	write	0.004	0.004	0.004
	read	0.003	0.004	0.008
	take	0.007	0.010	0.017
	query	0.002	0.003	0.005

Tabla 5.6.: Resultados de la evaluación de acceso a datos de TscME (en segundos).

5.6.2 Primitiva demand

En la tabla 5.7 se puede apreciar como la primitiva *demand*, al comportarse como una primitiva de red necesita tiempo para propagarse a través de la red.

Número de *demands* registradas		10	20	30
Versión anterior	write	0.004	0.004	0.004
Current version	demand	0.096	0.099	0.098
	write	0.006	0.015	0.027
	suggest	0.133	0.159	0.186

Tabla 5.7.: Rendimiento de la primitiva *demand* en un dispositivo móvil (en segundos).

5.6.3 SunSPOT gateway

Como puede verse en la tabla 5.8, el número de las SunSPOT conectadas no afecta al rendimiento de las primitivas *write* y *read(URL)*, ya que el gateway accede directamente a la URL de la SunSPOT determinada directamente. En cambio, en las primitivas *read(template)*, *take* y *query*, el número de SunSPOT conectadas afecta negativamente, ya que la información recolectada por el gateway es mayor.

Número de grafos		10			
Número de Spots		1	2	3	4
SunSpots	write	0.037	0.036	0.045	0.037
	read(URL)	0.020	0.017	0.029	0.028
	read(template)	0.198	0.304	0.567	0.586
	take	0.164	0.253	0.557	0.597
	query	0.170	0.282	0.396	0.507

Tabla 5.8.: Rendimiento del gatewayWOT (en segundos).

5.6.4 Escenario

Finalmente, se ha evaluado el tiempo necesitado para el siguiente escenario: dos ordenadores (uno de ellos conteniendo tres nodos y el otro con el nodo regulador), un dispositivo móvil (Nokia N95) y una SunSPOT. El tiempo necesario para cada acción del escenario puede verse en la tabla 5.9, tanto con el enfoque de servicios como con la primitiva *demand*. Todos los nodos utilizados en este escenarios han sido configurados para esperar un segundo a las respuestas.

Action	Demand based implementation	Service based implementation
Publicar la temperatura de una SunSPOT a través de la API REST	-	4.69
Descubrir nuevas localizaciones en el espacio	5.38	5.74
Actualizar la temperatura para una localización desconocida	1.91	3.67
Actualizar la temperatura para una localización conocida	1.18	2.03
Comprobar los cambios en la temperatura interior y exterior	11.06	10.5
Activación del ventilador desde que el *temperature manager* invoca su servicio	0.90	1.45

Tabla 5.9.: Mediciones del escenario propuesto (en segundos)

Como se puede apreciar, las principales diferencias entre los dos enfoques son:

- La obtención de información desde las SunSPOT se ha sustituido por el WOT gateway.
- La activación del ventilador es más rápida con el enfoque *demand*.
- El enfoque *demand* ha sido más fácil de desarrollar que el enfoque basado en servicios.

5.7 Análisis de resultados

Las validaciones llevadas a cabo en cada módulo verifican que la funcionalidad perseguida ha sido correctamente implementada. A su vez, los resultados obtenidos del escenario planteado demuestran que el middleware tiene un rendimiento adecuado para casos en los que el tiempo de respuesta no es crítico.

Además, cabe destacar que el módulo de coordinación está sujeto a distintas optimizaciones que serán objeto de futuros proyectos y tesis. Así, se deberían analizar en detalle aspectos como la escalabilidad y seguridad de la solución planteada, así como cómo afecta a nodos móviles y embebidos en términos de consumo de batería, utilidad del middleware para los desarrolladores,...

6. CONCLUSIONES

El resultado de la ejecución del proyecto ISMED en las anualidades 2009, 2009 y 2010 ha sido la provisión de una nueva plataforma middleware que facilita el desarrollo y despliegue de entornos inteligentes cooperativos equipados por dispositivos empotrados. Todos los módulos inicialmente planteados: descubrimiento, composición, razonamiento y aprendizaje han sido realizados. Sin embargo, ha sido el módulo de nexo de unión entre ellos, el módulo de modelado y coordinación semántica, realizado por la Universidad de Deusto, el que ha constituido mayor trabajo y en el que creemos se han realizado los avances más significativos. Prueba de ello es que ha dado lugar a dos publicaciones científicas en la conferencia TouchTheWeb 2010 y el journal UPGRADE, y está en proceso de elaboración un artículo detallado y extenso dirigido al journal of Network and Computer Applications. Es importante remarcar que la solución middleware final adaptada sigue un enfoque orientado a recursos más que servicios, lo que explica la solución adoptada en aspectos de composición más orientada a mash-ups de datos que de servicios.

Es también muy destacable el módulo de aprendizaje y el mecanismo de composición de servicios provistos por MGEP, que a su vez ha dado lugar a dos sendas tesis doctorales: "Learning frequent behaviours of the users in Intelligent Environments" por Asier Aztiria y "Enfoque integrado para el modelado, emparejamiento y composición sensible al contexto de servicios semánticos, basado en precondiciones y efectos, orientado a entornos inteligentes" por Aitor Urbieta. No obstante, es justo reseñar que el segundo investigador inició su tesis en el ámbito del proyecto ISMED aunque luego la continuó y finalizó en su actual puesto de trabajo, fuera de MGEP.

El middleware resultante del proyecto será la base de varias tesis doctorales actualmente en curso dentro de la Universidad de Deusto. Una distribución libre de tal software bajo licencia LGPL será publicada a comienzos de 2011 en la web del proyecto: `http://www.tecnologico.deusto.es/projects/ismed`. Este informe y todos los artículos científicos derivados de este proyecto también han sido y serán colgados de la citada web.

A. PUBLICACIONES DERIVADAS DEL TRABAJO EN ISMED

- Aitor Gómez-Goiri, Diego López de Ipiña: "A Triple Space-Based Semantic Distributed Middleware for Internet of Things". TouchTheWeb'10, International Workshop on Web-enabled Objects, Vienna (Austria), July 6, 2010.
- Aitor Gómez-Goiri, Mikel Emaldi, Diego López-de-Ipiña: "A Semantic Resource Oriented Middleware for Pervasive Environments". UPGRADE journal – The European Journal for the Informatics Professional, special issue on Internet Of Things, January 2011 (Submitted).
- Asier Aztiria, Alberto Izaguirre, Juan Carlos Augusto: "Learning patterns in Ambient Intelligence environments: A Survey". Artificial Intelligence Review, vol. 34, issue 1, pp. 35-51, Springer. May 23^{rd} , 2010. ISSN: 0269-2821, doi: 10.1007/s10462-010-9160-3.
- Asier Aztiria, Alberto Izaguirre, Rosa Basagoiti, Juan Carlos Augusto, Diane J. Cook: "Automatic Modeling of Frequent User Behaviours in Intelligent Environments". 6th International Conference on Intelligents Environments, July 19^{th} ,2010. ISBN: 978-0-7695-4149-5.
- Asier Aztiria, Juan Carlos Augusto, Rosa Basagoiti, Alberto Izaguirre: "Accurate Temporal Relationships in Sequences of User Behaviours in Intelligent Environments". International Symposium on Ambient Intelligence (ISAmI'2010), June 16^{th}, 2010. ISSN: 1867-5662. ISBN: 978-3-642-13267-4.

BIBLIOGRAFÍA (LIBROS Y ARTÍCULOS)

[1] The ambient assisted living joint programme. URL: `http://www.aal-europe.eu`.

[2] Explaining case based reasoning.

[3] http://iieg.essex.ac.uk/idorm2/index.htm.

[4] Increasing an individual's quality of life via their inteliigent home.

[5] Razonador basado en casos.

[6] D1.1 State-of-the-Art and requirements analysis. Technical report, TripCom, 2006.

[7] Rete algorithm. 2006.

[8] *Bossam*. 2008.

[9] *Crossbow, Wireless Sensor Network*. 2008.

[10] *DIG interface*. 2008.

[11] *DustNetworks, Wireless Sensor Network*. 2008.

[12] *Fact++*. 2008.

[13] *Gumstix Web Page*. 2008.

[14] *Gumstix Wiki*. 2008.

[15] *Hoolet*. 2008.

[16] *Jena Framework*. 2008.

[17] *JXTA Java Micro Edition Project*. 2008.

[18] *Kaon*. 2008.

[19] *Millennial, Wireless Sensor networks*. November 2008.

[20] *Pellet*. 2008.

[21] *Racer Pro*. 2008.

[22] *Sensicast, Wireless Sensor Network*. 2008.

[23] *SquidBee Project*. 2008.

[24] *Sun SPOT World*. 2008.

[25] *SweetRules*. 2008.

[26] E. Aarts. 365 days ambient intelligence in homelab, April 2003.

[27] E. Aarts. *The New Everyday. Views on Ambient Intelligence*. Uitgeverij 010 Publishers, 2003.

[28] E. Aarts. *Ambient Intelligence: A Multimedia Perspective*. 2004. ID: 1.

[29] E. Aarts. Ambient intelligence: A multimedia perspective. *IEEE Multimedia*, pages 12–19, 2004.

[30] E. Aarts, R. Harwig, and M. Schuurmans. Ambient intelligence. *The invisible future: the seamless integration of technology into everyday life*, 2001.

[31] E. Aarts and S. Marzano. The new everyday: Visions of ambient intelligence, 2003.

[32] Julio Abascal. *Ambient Intelligence for people with disabilities and elderly people.* 2004.

[33] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and Context-Awareness. pages 304–307. Springer-Verlag.

[34] Gregory D Abowd, Anind K Dey, Robert Orr, and Jason A Brotherton. Context-Awareness in wearable and ubiquitous computing. pages 179–180. Washington, DC, USA.

[35] T. Abraham and J. F. Roddick. Discovering meta-rules in mining temporal and spatio-temporal data.

[36] G. Acampora and V. Loia. *Using Fuzzy Technology in ambient intelligence environments.* 2005. ID: 19.

[37] G. Acampora and V. Loia. Using fuzzy technology in ambient intelligence environments, 2005.

[38] G. Acampora, V. Loia, M. Nappi, and S. Ricciardi. Ambient intelligence framework for context aware adaptive applications. *7th International Workshop on Computer Architecture for Machine Perception*, pages 327–332, 2005.

[39] G. Acampora, V. Loia, M. Nappi, and S. Ricciardi. Human-based models for smart devices in ambient intelligence, June 23-23 2005.

[40] Giovanni Acampora and Vincenzo Loia. Bridging the gap between services, devices and humans in ami environments. In *2005 IEEE International Conference on Industrial Technology, ICIT 2005*, volume 2005, pages 1257–1262, Hong Kong, Hong Kong, Dec 14-17 2005 2005. Dipartimento di Matematica e Informatica, Universita degli Studi di Salerno, 84084 Fisciano (Salerno), Italy, Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ 08855-1331, United States. URL: `http://dx.doi.org/10.1109/ICIT.2005.1600828`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: Proceedings of the IEEE International Conference on Industrial Technology.

[41] Rohit Aggarwal. Semantic web services languages and technologies: Comparison and discussion. *Technical Report LSDIS Lab, University of Georgia*, 2004.

[42] Rohit Aggarwal, Kunal Verma, John Miller, and William Milnor. Constraint driven web service composition in METEOR-S. pages 23–30. IEEE Computer Society.

[43] H. Aghajan, J.C. Augusto, and R. Lopez Cozar Delgado. *Human-centric interfaces for ambient intelligence.* Academic Press, 2009.

[44] R. Agrawal and R. Srikant. Mining sequential patterns. In *Pro. 11th International Conference on Data Engineering*, pages 3–14, 1995.

[45] R. Agrawal and R. Srikant. Mining sequential patterns, 1995.

[46] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[47] H. Ailisto, Petteri Alahuhta, V. Haataja, V. Kyllönen, and M. Lindholm. Structuring context aware applications: Five-Layer model and example case, position paper. *Models and concepts for ubiquitous computing, workshop at UbiComp*, 2002.

[48] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. Schmidt, A. Sheth, and K. Verma. Web service Semantics-WSDL-S. *A joint UGA-IBM Technical Note, version*, 1, 2005.

[49] R. Akkiraju, B. Sapkota, and February of. *Semantic annotations for WSDL and XML schema - Usage Guide (Working Group Note)*, volume 2008. 2007.

[50] Varol Akman and Mehmet Surav. Use of situation theory in context modeling. *Computational Intelligence*, 13(3):427–438, 1997. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[51] I. F Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[52] I F Akyldiz, W Su, Y Sankarasubramanian, and E Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40:102–114, 2002.

[53] Fahd Al-Bin-Ali. A multi-layered approach for modeling activities in smart spaces. In *Proceedings of the International Conference on Artificial Intelligence, IC-AI'04*, volume 2, pages 712–716, Las Vegas, NV, United States, Jun 21-24 2004 2004.

[54] A. Alamri, M. Eid, and A. El Saddik. Classification of the state-of-the-art dynamic web services composition techniques. *International Journal of Web and Grid Services*, 2(2):148–166, 2006.

[55] Safdar Ali and Stephan Kiefer. *microOR – A Micro OWL DL Reasoner for Ambient Intelligent Devices*, volume 5529/2009.

[56] R. A. Aliev and R. R. Aliev. *Soft Computing and Its Applications*. World Scientific, 2001.

[57] J. Allen. Towards a general theory of action and time. In *Artificial Intelligence*, volume 23, pages 123–154, 1984.

[58] José Manuel Alonso. *Estándares del W3C en BPM*. W3C Oficina Española, 2005.

[59] R. Alur. Timed automata. *Computer-Aided Verification*, 1633:8–22, 1999.

[60] R. Alur and D. L Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[61] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, and S. Thatte. Business process execution language for web services version 1.1. *Specification, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems*, 2003.

[62] Bernhard Angerer. *Space-Based Programming*. O'Reilly Media, 2003.

[63] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, and T. Payne. DAML-S: web service description for the semantic web. *The Semantic Web-ISWC*, 363, 2002.

[64] A. Ankolekar, M. Burstein, J. R Hobbs, O. Lassila, D. L Martin, S. A McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: semantic markup for web services. *Proceedings of the International Semantic Web Working Symposium*, 124, 2001.

[65] Knarig Arabshian and Henning Schulzrinne. GloServ: global service discovery architecture. *mobiquitous*, 00:319–325, 2004.

[66] Josep Lluis Arcos, Maarten Grachten, and Ramon Lopez De Mantaras. Extracting performers'behaviors to annotate cases in a cbr system for musical tempo transformations. In *5th International Conference on Case-Based Reasoning, ICCBR 2003*, volume 2689, pages 20–34, Trondheim, Norway, Jun 23-26 2003 2003. CSIC, Artif. Intell. Research Institute, Campus UAB, 08193 Bellaterra, Catalonia, Spain, Springer Verlag, Heidelberg, Germany. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved; T3: Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science).

[67] W. S Ark and T. Selker. A look at human interaction with pervasive computers. *IBM Systems Journal*, 38(4):504–507, 1999.

[68] A. Arkin. Business process modeling language (BPML). *BPMI.org*, 2002.

[69] A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C. K Liu, S. Thatte, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. *Working Draft.WS-BPEL TC OASIS, May*, 2005.

[70] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, and P. Takacsi-Nagy. Web service choreography interface (WSCI) 1.0. *Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems*, 2002.

[71] Sinuhe Arroyo and Miguel-Angel Sicilia. Design principles of a choreography conceptual framework. pages 83–89.

[72] J. C. Augusto. *Ambient Intelligence: the Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence*, pages 213–234. Intelligent Computing Everywhere. Springer London, 2007.

[73] J. C. Augusto. Past, present and future of ambient intelligence and smart environments. In *1st International Conference on Agents and Artificial Intelligence (ICAART)*, 2009.

[74] J. C. Augusto and D. J. Cook. Ambient intelligence: applications in society and opportunities for ai. In *20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

[75] J. C. Augusto, J. Liu, and L. Cheen. *Using Ambient Intelligence for Disaster Management*, pages 171–178. Knowledge-Based Intelligent Information and Engineering Systems. Springer Berlin / Heidelberg, 2006.

[76] J. C. Augusto and P. McCullagh. Ambient intelligence: Concepts and applications. In *Computer Science and Information Systems*, volume 4, pages 1–28. ComSIS Consortium, 2007.

[77] J. C. Augusto and C. D. Nugent. A new architecture for smart homes based on adb and temporal reasoning, 2004.

[78] J. C. Augusto and C. D. Nugent. The use of temporal reasoning and management of complex events in smart homes. In *Proccedings of European Conference on AI (ECAI 2004)*, pages 778–782. IO Press, 2004.

[79] J. C. Augusto and C. D. Nugent. The use of temporal reasoning and management of complex events in smart homes, 2004.

[80] J. C. Augusto and C. D. Nugent. *Designing Smart Homes. The Role of Artificial Intelligence.* Springer-Verlag, 2006.

[81] J. C. Augusto and C. D. Nugent, editors. *Designing Smart Homes. The Role of Artificial Intelligence.* Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[82] J. C. Augusto and C. D. Nugent. *Smart homes can be smarter*, pages 1–15. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[83] J. C. Augusto and C. D. Nugent. Smart homes can be smarter. In *Designing Smart Homes. The Role of Artificial Intelligence*, pages 1–15. Sch. of Comput. & Math., Ulster Univ., Jordanstown, UK, Springer-Verlag, 2006.

[84] J. C. Augusto and C. D. Nugent. Smart homes can be smarter. In *Designing Smart Homes. The Role of Artificial Intelligence*, pages 1–15, 2006.

[85] J. C. Augusto, C. D. Nugent, S. Martin, and C. Olphert. Software and konwledge engineering aspects of smart homes applied to health, 2005.

[86] J. C. Augusto and D. Shapiro. Artificial intelligence techniques for ambient intelligence. In *1st Workshop on Artificial Intelligence Techniques for Ambient Intelligence (Co-located event of ECAI 2006)*, 2006.

[87] J.C. Augusto. Ambient intelligence: The confluence of pervasive computing and artificial intelligence. In Alfons Schuster, editor, *Intelligent Computing Everywhere*, pages 213–234. Springer, 2007.

[88] J.C. Augusto and D. Cook. *Ambient Intelligence: applications in society and opportunities for AI. Tutorial Lecture Notes delivered at 20th International Joint Conference on Artificial Intelligence (IJCAI-07).* IJCAI, Hyderabad, India, January 2007.

[89] J.C. Augusto and C.D. Nugent. Smart homes can be smarter. In J. C. Augusto and C. D. Nugent, editors, *Introductory chapter to "Designing Smart Homes: The role of Artificial Intelligence"*, volume 4008, pages 1–15. Springer, 2006.

[90] A. Aztiria, J. C. Augusto, and A. Izaguirre. Spatial and temporal aspects for pattern representation and discovery in intelligent environments. In *Workshop on Spatial and Temporal Reasoning at 18th European Conference on Artificial Intelligence (ECAI 2008)*, 2008.

[91] A. Aztiria, J. C. Augusto, A. Izaguirre, and D. J. Cook. Learning accurate temporal relations from user actions in intelligent environments. In *Proceedings of the 3rd Symposium of Ubiquitous Computing and Ambient Intelligence*, volume 51/2009, pages 274–283, 2008.

[92] A. Aztiria, A. Izaguirre, R. Basagoiti, and J.C. Augusto. *Learning about preferences and common behaviours of the user in an intelligent environment*, pages 289–315. Behaviour Monitorin and Interpretation-BMI-Smart environments, Ambient Intelligence and Smart Environments. IOS Press, 2009.

[93] A. Aztiria, E. Saenz de Argandoña, C. Garcia, A. Arana, and A. Izaguirre. Aplication of artificial intelligence techniques to sheet metal forming processes global control. 2007.

[94] E. Babulak. Automated smart house 2015 via ubiquitous computing, 2006.

[95] J. Bacon, J. Bates, and D. Halls. Location-oriented multimedia. *IEEE Personal Communications*, 4(5):48–57, 1997. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[96] Magdalena Balazinska, Hari Balakrishnan, and David Karger. INS/Twine: a scalable Peer-to-Peer architecture for intentional resource discovery.

[97] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on Context-Aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[98] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, and S. Sharma. Web services conversation language (WSCL) 1.0. *W3C Note*, 2002.

[99] Sharad Bansal and Jose M Vidal. Matchmaking of web services based on the DAML-S service model. pages 926–927. ACM Press.

[100] J. E Bardram. The java context awareness framework (JCAF) - a service infrastructure and programming framework for Context-Aware applications. *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, 2005.

[101] T. S. Barger, D. E. Brown, and M. Alwan. Health-status monitoring through analysis of behavioral patterns. *IEEE Transactions on Systems, Man and Cybernetics, Part A (Systems and Humans)*, 35(1):22–7, 01 2005. URL: `http://dx.doi.org/10.1109/TSMCA.2004.838474`.

[102] A. Barros, M. Dumas, and P. Oaks. A critical overview of the web services choreography description language (ws-cdl). *BPTrends Newsletter*, 3(3), 2005.

[103] Alistair P Barros, Marlon Dumas, and Arthur H. M ter Hofstede. Service interaction patterns. pages 302–318. crossref: DBLP:conf/bpm/2005.

[104] J. Barton and T. Kindberg. *The Cooltown User Experience*. Hewlett-Packard Laboratories, 2001.

[105] John Barton and Tim Kindberg. *A Web-Based Nomadic Computing System*, volume 35. 2000. annote: "John Barton (Internet and Mobile Systems Laboratory); Tim Kindberg (Internet and Mobile Systems Laboratory);".

[106] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983.

[107] S. Battle. Semantic web services ontology (SWSO). *Member submission, W3C, September*, 2005.

[108] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, and D. McGuinness. Semantic web services framework (SWSF). *W3C Member Submission*, 9, 2005.

[109] J. Bauer. Identification and modeling of contexts for different information scenarios in air traffic. *Technische Universität Berlin*, 2003.

[110] Russell Beale. Supporting serendipity: Using ambient intelligence to augment user exploration for data mining and web browsing. *International Journal of Human Computer Studies*, 65(5):421–433, 2007. URL: `http://dx.doi.org/10.1016/j.ijhcs.2006.11.012`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[111] Sean Bechhofer and Peter F Patel-Schneider. *DIG 2.0: The DIG Description Logic Interface Overview*. 2006.

[112] Dave Beckett. Swad-europe deliverable 3.11: Developer workshop report 4 - workshop on semantic web storage and retrieval. Technical report, World Wide Web Consortium, 2004.

[113] R. Begg and R. Hassan. *Artificial neural networks in smart homes*, pages 146–164. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[114] Fabio Bellifimine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. *Jade - A White Paper*, volume 3. 2003.

[115] U. Bellur and N. C Narendra. Towards service orientation in pervasive computing systems. *International Conference on Information Technology: Coding and Computing*, 2:289–295, 2005.

[116] F. Bennett, T. Richardson, and A. Harter. Teleporting-making applications mobile. *Proceedings on Workshop on Mobile Computing Systems and Applications*, pages 82–84, 1994.

[117] D. Berardi. *Automatic Composition Services: Models, Techniques and Tools*. 2005.

[118] D. Berardi, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, J. Su, and S. Tabet. Semantic web services language (SWSL). *Technical report, Semantic Web Services Initiative*, 2005.

[119] D. Berardi, M. Gruninger, R. Hull, and S. McIlraith. Towards a first-order ontology for semantic web services. *Working notes of the W3C Workshop on Constraints and Capabilities for Web Services*, 2004.

[120] J O Berger. *Statistical Decisions*. Springer-Verlag, 1985.

[121] J. A Bergrstra and J. W Klop. Algebra of communicating processes with abstractions. *Theoretical Computer Science*, 37(1):77–121, 1985.

[122] Tim Berners-Lee, J. Hendler, and O. Lassila. *The Semantic Web*, volume 284. 2001.

[123] A. Bernstein and C. Kiefer. Imprecise RDQL: towards generic retrieval in ontologies using similarity joins. pages 1684–1689. ACM New York, NY, USA.

[124] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. MBP: a model based planner. *Proc.of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information*, 2001.

[125] Piergiorgio Bertoli, Alessandro Cimatti, Marco Pistore, and Paolo Traverso. A framework for planning with extended goals under partial observability. pages 215–225. crossref: DBLP:conf/aips/2003.

[126] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. pages 473–478. crossref: DBLP:conf/ijcai/2001.

[127] A. Bharathidasan and V. A. S Ponduru. Sensor networks: An overview. Technical report, 2003.

[128] A. Bhattacharya and S. K. Das. Lezi-update: an information-theoretic approach to track mobile users in pcs networks. In *Proceedings of 5th Annual Joint ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99)*, pages 1–12, 15-20 Aug. 1999 1999. URL: `http://dx.doi.org/10.1145/313451.313457`.

[129] D. Bianchini, V. De Antonellis, M. Melchiori, and D. Salvi. Semantic-Enriched service discovery. IEEE Computer Society Washington, DC, USA.

[130] Z. Zenn Bien and Hyong-Euk Lee. Effective learning system techniques for human-robot interaction in service environment. *Knowledge-Based Systems*, 20(5):439–456, 2007. URL: `http://dx.doi.org/10.1016/j.knosys.2007.01.005`.

[131] Walter Binder, Ion Constantinescu, and Boi Faltings. Decentralized orchestration of CompositeWeb services. pages 869–876. IEEE Computer Society.

[132] Fernando Bobillo, Juan Gomez-Romero, and Ramón Pérez-Pérez. Towards semantic web services: a brief over-view. IADIS Press.

[133] F. Boekhorst. Ambient intelligence: The next paradigm for consumer electronics: How will it affect silicon?, 2002.

[134] A. Boisvert and R. B. Rubio. Architecture for intelligent thermostats that learn from occupants'behavior. In *ASHRAE Transactions*, pages 124–130, 1999.

[135] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *COMP.NETWORKS ISDN SYST.*, 14(1):25–59, 1987.

[136] A. Bonarini. Anytime learning and adaptation of structured fuzzy behaviors, 1997.

[137] G. Booch, J. Rumbaugh, and I. Jacobson. *The unified modeling language user guide*. Addison-Wesley Reading Mass, 1999.

[138] L. Bordeaux, G. Salaun, D. Berardi, and M. Mecella. When are two web services compatible. *30th International Conference on Very Large Databases on 5th Workshop on Technologies for E-Services*, 3324:15–28, 2004.

[139] L. Botelho, A. Fernandez, B. Fries, M. Klusch, L. Pereira, T. Santos, P. Pais, and M. Vasirani. Service discovery. Birkhauser Verlag, Springer, 2008.

[140] André Bottaro, Johann Bourcier, Clement Escoffier, and Philippe Lalanda. Autonomic Context-Aware service composition. *2nd IEEE International Conference on Pervasive Services*, 2007.

[141] B. Bouzy and T. Cazenave. Using the object oriented paradigm to model context in computer go. *Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context*, 1997.

[142] K Brammer and G Siffling. *Kalman Bucy Filters*. Artech House, 1989.

[143] T. Bray, J. Paoli, and C. M Sperberg-McQueen. Extensible markup language (XML) 1.0. *W3C Recommendation*, 6, 2000.

[144] O. Brdiczka, P. Reignier, and J. Crowley. Detecting individual activities from video in a smart home. In *Proceedings of the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 363–370, 2007.

[145] O. Brdiczka, P. Reignier, and J. L. Crowley. Supervised learning of an abstract context model for an intelligent environment. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, volume 121, pages 259–264. ACM, 2005.

[146] C Brezeal and B Scassellati. A context-dependent attention system for a social robot. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1146–1151, 1999.

[147] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. *Sesame: A Generic Architecture for Storing and Querying RDF*.

[148] A. Brogi, S. Corfini, and R. Popescu. Composition-oriented service discovery.

[149] Yérom-David Bromberg and Valérie Issarny. INDISS: interoperable discovery system for networked services. pages 164–183. crossref: DBLP:conf/middleware/2005.

[150] P. J Brown. The stick-e document: a framework for creating context-aware applications. *Proceedings of Electronic Publishing*, 96:259–272, 1996.

[151] P. J Brown, J. D Bovey, and Xian Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, 1997.

[152] Antonio Bucchiarone and Stefania Gnesi. A survey on services composition languages and models. Andrea Polini.

[153] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 2004.

[154] P. Busetta, T. Kuflik, M. Merzi, and S. Rossi. Service delivery in smart environments by implicit organizations. pages 356–363.

[155] Christoph Bussler. A minimal triple space computing architecture. Technical report, Digital Enterprise Research Institute, 2005.

[156] Y. Cai. Empathic computing, 2006.

[157] Yang Cai. Ambient intelligence: From interaction to insight. *International Journal of Human Computer Studies*, 65(5):419–420, 2007. URL: `http://dx.doi.org/10.1016/j.ijhcs.2006.11.008`. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[158] V. Callaghan, A. Kameas, and D. Reyes, editors. *Proceedings of the 5th International Conference on Intelligent Environments*. IOSPress, 2009.

[159] E. Campo, S. Bonhomme, M. Chan, and D. Esteve. Learning life habits and practices: an issue to the smart home. In *International Conference on Smart Homes and health Telematic*, pages 355–358, 2006.

[160] Maria Celeste Campo. *Tecnologias middleware para el desarrollo de servicios en entornos de computacion ubicua*. 2004.

[161] L. Capra, W. Emmerich, and C. Mascolo. Reflective middleware solutions for Context-Aware applications. *Proceedings of REFLECTION*, 2192:126–133, 2001.

[162] L. Capra, S. Zachariadis, and C. Mascolo. Q-CAD: QoS and context aware discovery framework for adaptive mobile systems. pages 453–456. IEEE Computer Society Press.

[163] J. Cardoso and A. Sheth. *Semantic Web Services, Processes and Applications*. Springer Book Series on Semantic Web and Beyond: Computing for Human Experience, 2006.

[164] Jorge Cardoso, Amit P Sheth, John A Miller, Jonathan Arnold, and Krys Kochut. Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281–308, 2004.

[165] R. Cardoso, S. Ben Mokhtar, A. Urbieta, and V. Issarny. EVEY: enhancing privacy of service discovery in pervasive computing.

[166] R. Cardoso, S. Ben Mokhtar, A. Urbieta, and V. Issarny. Privacy-Aware service discovery for pervasive computing.

[167] K. Carey, D. Lewis, S. Higel, and V. Wade. Adaptive composite service plans for ubiquitous computing.

[168] N. Carretero and A. B. Bermejo. Inteligencia ambiental, 2005.

[169] Noelia Carretero and Ana Belén Bermejo. Inteligencia ambiental. Technical report, 2005.

[170] F. Casati, S. Ilnicki, and L. J Jin. *Adaptive and Dynamic Service Composition in EFlow*. Springer, 2000.

[171] F. Casati, M. Sayal, and M. C Shan. Developing e-services for composing e-services. *Proceedings of CAISE 2001*, 2001.

[172] G. Castellano, A.M. Fanelli, and C. Mencar. Generation of interpretable fuzzy granules by a double-clustering technique. *Arch. Contr. Sci.*, 12:397–410, 2002.

[173] J. L. Castro, J. J. Castro-Schez, and J. M. Zurita. Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems. *Fuzzy Sets and Systems*, 101(3):331–42, 02/01 1999. URL: `http://dx.doi.org/10.1016/S0165-0114(97)00105-X`.

[174] N. Cercone, A. An, and C. Chan. Rule-induction and case-based reasoning: hybrid architectures appear advantageous. *IEEE Transactions on Knowledge and Data Engineering*, 11:166–174, 1999.

[175] Girish B Chafle, Sunil Chandra, Vijay Mann, and Mangala Gowri Nanda. Decentralized orchestration of composite web services. pages 134–143. ACM Press.

[176] Dipanjan Chakraborty. *Service Discovery and Composition in Pervasive Environments*. 2004.

[177] Dipanjan Chakraborty and Anupam Joshi. Dynamic service composition: State-of-the-Art and research directions. *Technical Report TR-CS-01-19*, 19, 2001.

[178] Dipanjan Chakraborty, Anupam Joshi, Tim Finin, and Yelena Yesha. Service composition for mobile environments. *Journal on Mobile Networking and Applications, Special Issue on Mobile Services*, 10(4):435–451, 2005.

[179] D. Chalmers. *Contextual Mediation to Support Ubiquitous Computing*. 2002.

[180] M. Champion, C. Ferris, E. Newcomer, and D. Orchard. Web services architecture. *W3C Tehnical Report*, 2002.

[181] J. P. Chan and B. G. Batchelor. Learning method for the inspection of continuously repeated patterns, 1992.

[182] K. S May Chan, Judith Bishop, and Luciano Baresi. Survey and comparison of planning techniques for web services composition. Technical report, University of Pretoria, 2007.

[183] M. Chan, C. Hariton, P. Ringeard, and E. Campo. Smart house automation system for the elderly and the disabled. In *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics*, pages 1586–1589, 1995.

[184] Marie Chan, Cyril Hariton, Patrick Ringeard, and Eric Campo. Smart house automation system for the elderly and the disabled. volume 2, pages 1586–1589. IEEE, Piscataway, NJ, USA INSERM CJF, Toulouse, Fr. ID: 130; Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics.

[185] Y. Charif and N. Sabouret. An overview of semantic web services composition approaches. *Electronic Notes in Theoretical Computer Science*, 85(6), 2005.

[186] W. Cheetham, S. Shiu, and R. O. Weber. Soft case-based reasoning. *Knowledge Engineering Review*, 20(3):267–9, 2005. URL: `http://dx.doi.org/10.1017/S0269888906000579`.

[187] G. Chen and D. Kotz. Solar: An open platform for context-aware mobile applications.

[188] Guanling Chen and David Kotz. A survey of Context-Aware mobile computing research. Technical report, Dartmouth College, 2000.

[189] Harry Chen. *An Intelligent Broker Architecture for Context-Aware Systems*. 2002. Propuesta de Tesis de Harry Chen.

[190] Harry Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. 2004.

[191] Harry Chen, Tim Finin, and Anupam Joshi. Semantic web in a pervasive Context-Aware architecture. pages 33–40.

[192] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for Context-Aware pervasive computing environments. *Workshop Ontologies and Distributed Systems*, 2003.

[193] Harry Chen, Tim Finin, and Anupam Joshi. A context broker for building smart meeting rooms. *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium*, pages 53–60, 2004.

[194] Harry Chen, Tim Finin, and Anupam Joshi. *An Ontology for Context-Aware Pervasive Computing Environments*, volume 18. 2004.

[195] Harry Chen, Tim Finin, and Anupam Joshi. Semantic web in the context broker architecture. *Proceedings of the Second Annual IEEE International Conference on Pervasive Computer and Communications*, pages 277–286, 2004.

[196] Harry Chen, Tim Finin, and Anupam Joshi. The SOUPA ontology for pervasive computing. Springer, 2005.

[197] Harry Chen, Filip Perich, Dijapan Chakraborty, Tim Finin, and Anupam Joshi. Intelligent agents meet semantic web in a smart meeting room. *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 854–861, 2004.

[198] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. SOUPA: standard ontology for ubiquitous and pervasive applications. pages 258–267. crossref: DBLP:conf/mobiquitous/2004.

[199] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. Using OWL in a pervasive computing broker. *Workshop on Ontologies in Agent Systems (AAMAS 2003)*, 2003.

[200] Peter Pin-Shan Chen. The entity-relationship model-toward a unified view of data. *ACM Trans.Database Syst.*, 1(1):9–36, 1976.

[201] Keith Cheverst, Keith Mitchell, and Nigel Davies. Design of an object model for a context sensitive tourist GUIDE. *Computers and Graphics (Pergamon)*, 23(6):883–891, 1999. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[202] M. Chidambaram, M. Sundaram, M. Babu, R. Uthariariaraj, and R. Shriram. Resolving users'behavior modeling ambiguities in fuzzy-timed smart homes using only rfids. *IJCSNS International Journal of Computer Science and Network Security*, 6(11):179–184, 2006.

[203] R. Chinnici, M. Gudgin, J. J Moreau, J. Schlimmer, and S. Weerawarana. Web services description language (WSDL) version 2.0 part 1: Core language. *W3C Working Draft*, 26, 2004.

[204] J. Choi, D. Shin, and D. Shin. Research and implementation of the context-aware middleware for controlling home appliances. *IEEE Transactions on Consumer Electronics*, 51(1):301–306, 2005.

[205] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. *W3C Note*, 2001.

[206] H. I. Christensen. Intelligent home appliances, 2001.

[207] Eleni Christopoulou, Christos Goumopoulos, and Achilles Kameas. An ontology-based context management and reasoning process for UbiComp applications. pages 265–270. ACM Press.

[208] Eleni Christopoulou, Christos Goumopoulos, Ioannis Zaharakis, and Achilles Kameas. An ontology-based conceptual model for composing Context-Aware applications. pages 7–10.

[209] Eleni Christopoulou and Achilles Kameas. GAS ontology: an ontology for collaboration among ubiquitous computing devices. *International Journal of Human-Computers Studies*, 62(5):664–685, 2005.

[210] E. Chtcherbina and M. Franz. Peer-to-peer coordination framework (p2pc): Enabler of mobile ad-hoc networking for medicine, business, and entertainment. *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003)*, 2003.

[211] E. Cimpian, M. Moran, E. Oren, T. Vitvar, and M. Zaremba. Overview and scope of WSMX. *WSMO Working Draft v0.2*, 2(08), 2005.

[212] J. Clark and S. DeRose. XML path language (XPath) version 1.0. *W3C Recommendation*, 16:1999, 1999.

[213] Daniela Barreiro Claro, Patrick Albers, and Jin-Kao Hao. Approaches of web services composition - comparison between BPEL4WS and OWL-S. pages 208–213. crossref: DBLP:conf/iceis/2005.

[214] L. Clement, A. Hately, C. von Riegen, and T. Rogers. UDDI version 3.0. 2. *UDDI Spec Technical Committee Draft. Tc*, 220041019.

[215] M. H. Coen. Design principles for intelligent environments. In *Proceedings of the 1998 15th National Conference on Artificial Intelligence, AAAI*, pages 547–554. AAAI Press, 1998.

[216] Michael H Coen. *The Future of Human-Computer Interaction, or How I learned to stop worrying and love my Intelligent Room*, volume 14. 1999.

[217] C. Combi and R. Rossato. *Temporal constraints with multiple granularities in smart homes*, pages 35–56. Designing Smart Homes. The Role of Artificial Intelligence. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[218] JXTA Community. *JXTA Community Projects*. 2008.

[219] World Wide Web consortium. *Guía Breve de Servicios Web*. World Wide Web consortium, 2008.

[220] I. Constantinescu and B. Faltings. Efficient matchmaking and directory services. pages 75–81.

[221] D. Cook, J.C. Augusto, and V.R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, 2009.

[222] D. Cook and M. Schmitter-Edgecombe. Activity profiling using pervasive sensing in smart homes. *IEEE Transactions on Information Technology for Biomedicine*, 2008.

[223] D. J. Cook and S. K. Das. *Smart Environments: Technology, Protocols and Applications*. Wiley-Interscience, 2005.

[224] D. J. Cook and S. K. Das. How smart are our environments? an updated look at the state of the art. In *Pervasive and Mobile Computing*, volume 3, pages 53–73. Elsevier Science, 2007.

[225] D. J. Cook, M. Huber, K. Gopalratnam, and M. Youngblood. Learning to control a smart home environment. In *Innovative Applications of Artificial Intelligence*, 2003.

[226] D. J. Cook, M. Youngblood, and S. K. Das. *A multi-agent approach to controlling a smart environment*, pages 165–82. Designing Smart Homes. The Role of Artificial Intelligence. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[227] W. R Cook and J. Misra. A structured orchestration language. *Submitted for publication*, 2005.

[228] M. A Corella and P. Castells. Semantic-based taxonomic categorization of web services. pages 105–112.

[229] I. B. M Corporation. *BPEL for Java technology BPELJ*, volume 2006. 2006.

[230] I. B. M Corporation. *Websphere Software*, volume 2006. 2006.

[231] Microsoft Corporation. *Microsoft biztalk server*, volume 2006. 2006.

[232] Oracle Corporation. *Oracle bpel process manager*, volume 2006. 2006.

[233] S. Costantini, R. DellÁcqua, L. M. Pereira, and F. Toni. Towards a model of evolving agents for ambient intelligence, 2006.

[234] S. D Cotofana, S. Wong, and S. Vassiliadis. Embedded processors: Characteristics and trends. pages 17–24.

[235] T. Cottenier and T. Elrad. Adaptive embedded services for pervasive computing.

[236] National Research Council. Embedded everywhere, 2001.

[237] Fulvio Crivellaro and Gabriele Genovese. *uJena : Gestione di ontologie sui dispositivi mobili*. PhD thesis, 2007.

[238] Crossbow. *MICAz 2.4 GHz*. 2009.

[239] Steven E Czerwinski, Ben Y Zhao, Todd D Hodes, Anthony D Joseph, and Randy H Katz. An architecture for a secure service discovery service. pages 24–35. ACM Press.

[240] C. dÁmato, S. Staab, N. Fanizzi, and F. Esposito. Efficient discovery of services specified in description logics languages. pages 15–29.

[241] M. dÁquin, J. Lieber, and A. Napoli. Case-based reasoning within semantic web technologies. In *Proceedings*, Artificial Intelligence: Methodology, Systems, and Applications 12th International Conference, AIMSA 2006, pages 190–200, Varna, Bulgaria, 12-15 Sept. 2006 2006. LORIA, Nancy, France, Springer-Verlag. M1: Copyright 2006, The Institution of Engineering and Technology; T3: Artificial Intelligence: Methodology, Systems, and Applications. 12th International Conference, AIMSA 2006. Proceedings (Lecture Notes in Artificial Intelligence Vol. 4183).

[242] S. K. Das and Diane J. Cook. Designing and modeling smart environments. In *Proc. of International Sysmposium on Wireless, Mobile and Multimedia Networks*, pages 490–494, 2006.

[243] Sajal K Das and Diane J Cook. Designing and modeling smart environments. volume 2006, pages 490–494. Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States Center for Research in Wireless Mobility and Networking (CReWMaN), University of Texas, Arlington. ID: 55; Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved; T3: Proceedings - WoWMoM 2006: 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks.

[244] C. J Date. *A guide to the SQL standard*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[245] Fabrizio Davide, Pierpaolo Loreti, Massimiliano Lunghi, Giuseppe Riva, and Francesco Vatalaro. Communications through virtual technologies. pages 124–154, 2002.

[246] J. de Bruijn. Using Ontologies.Enabling knowledge sharing and reuse on the semantic web. *Technical Report DERI-2003-10-29*, 2003.

[247] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The web service modeling language WSML. *WSML Final Draft*, 16, 2005.

[248] R. L. de Mantaras and E. Plaza. Case-based reasoning: an overview. *AI Communications*, 10(1):21–9, 03 1997. M1: Copyright 1997, IEE.

[249] Gero Decker and Johannes Maria Zaha. Pattern based evaluation of WS-CDL. Technical report, Faculty of IT, Queensland University of Technology, Australia, 2006.

[250] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: ontology based access to distributed and semi-structured information. pages 351–69. Kluwer Academic Publishers Inst. fur Angewandte Inf. und Formale Beschreibungsverfahren, Karlsruhe Univ., Germany. M1: Copyright 2000, IEE.

[251] G Delapierre, H Grange, B Chambaz, and L Destannes. Polymer-based capacitive humidity sensor. *Sensors and Actuators*, 4(1):97–104, 1983.

[252] M. Delgado. Definición del modelo del negocio y del dominio utilizando razonamiento basado en casos, 2001.

[253] G. Demiris, M. Skubic, J. Keller, J. Rantz, D. Parker Oliver, M. A. Aud, J. Lee, K. Burks, and N. Green. Nurse participation in the design of user interfaces for a smart home system, 2006.

[254] G. Demiris, M. Skubic, M. Rantz, J. Keller, M. Aud, B. Hensel, and Z. He. Smart home sensors for the elderly: a model for participatory formative evaluation, 2006.

[255] A. K Dey. Supporting the construction of context-aware applications. *Dagstuhl seminar on Ubiquitous Computing*, 2001.

[256] Anind K Dey. *Understanding and Using Context*, volume 5. 2001.

[257] Anind K Dey, Gregory D Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2-4):97–166, 2001. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[258] Anind Kumar Dey. *Providing architectural support for building context-aware applications.* 2000. note: Director-Gregory D. Abowd.

[259] G. Diaz, J. J Pardo, M. E Cambronero, V. Valero, and F. Cuartero. Automatic translation of WS-CDL choreographies to timed automata. *Formal techniques for Computer Systems and Business Processes: Proceedings of the International Workshop on Web Services and Formal Methods*, pages 230–242, 2005.

[260] Gregorio Diaz, Maria-Emilia Cambronero, Juan Jose Pardo, Valentin Valero, and Fernando Cuartero. Automatic generation of correct web services choreographies and orchestrations with model checking techniques. page 186. crossref: DBLP:conf/aict/2006.

[261] Ian Dickinson. *The Semantic Web and Software Agents: Partners, or Just Neighbours?* 2004.

[262] Digi. *ConnectPort X Gateways.*

[263] Digi. *XBee Sensors.*

[264] Li Ding, Pranam Kolari, Zhongli Ding, Sasikanth Avancha, Tim Finin, Anupam Joshi, and C. S Tr. Using ontologies in the semantic web: A survey. Technical report, UMBC, 2005.

[265] F. Doctor, H. Hagras, and V. Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. In *IEEE Transactions on systems, man and cybernetics*, volume 35, pages 55–65, 2005.

[266] F. Doctor, H. Hagras, and V. Callaghan. *A fuzzy Embedded Agent-Based Approach for realizing Ambient Intelligence in Intelligent Inhabited Environments*, volume 35. 2005. ID: 7.

[267] A. Dogac, G. Laleci, and Y. Kabak. A context framework for ambient intelligence. *Proceedings of the eChallenges conference of the European Commission*, pages 2003–2010, 2003.

[268] J. Domingue, L. Cabral, F. Hakimpour, D. Sell, and E. Motta. IRS-III: a platform and infrastructure for creating WSMO-based semantic web services. *Proc.of the Workshop on WSMO Implementations*, 2004.

[269] J. S Dong, Y. Liu, J. Sun, and X. Zhang. Verification of computation orchestration via timed automata. *8th International Conference on Formal Engineering Methods*, 2006.

[270] J. Dooley, V. Callaghan, H. Hagras, P. Bull, and D. Rohlfing. Ambient intelligence - knowledge representation, processing and distribution in intelligent inhabited environments. In *2nd IET International Conference on Intelligent Environments, IE 06*, pages 51–59, 2006.

[271] S. Dornbush, J. English, T. Oates, Z. Segall, and A. Joshi. Xpod: A human activity aware learning mobile music player. In *Proceeding of the 2nd workshop on Artificial Intelligence Techniques for Ambient Intelligence*, pages 45–50, 2007.

[272] T. B Downing. *Java RMI: Remote Method Invocation.* IDG Books Worldwide, Inc., 1998.

[273] Droosl. *The Drools Rule engine*, volume 2006. 2006.

[274] D. Dubois, F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, and H. Prade. Case-based reasoning: a fuzzy approach. In *IJCAI'97 Workshop*, Fuzzy Logic in Artificial Intelligence, pages 79–90, Nagoya, Japan, 23-24 Aug. 1997 1999. Inst. de Recherche en Inf., Univ. Paul Sabatier, Toulouse, France, Springer-Verlag. M1: Copyright 1999, IEE; T3: Fuzzy Logic in Artificial Intelligence. IJCAI'97 Workshop. Selected and Invited Papers.

[275] D. Dubois, F. Esteva, P. Garcia, L. Godo, R. L. De Mantaras, and H. Prade. Fuzzy modelling of case-based reasoning and decision. Case-Based Reasoning Research and Development. Second International Conference on Case-Based Reasoning, ICCBR-97, pages 599–610. IRIT, Univ. Paul Sabatier, Toulouse, France, 25-27 July 1997 1997.

[276] D. Dubois, F. Esteva, P. Garcia, L. Godo, R. L. De Mantaras, and H. Prade. Fuzzy modelling of case-based reasoning and decision. In *Proceedings*, Case-Based Reasoning Research and Development. Second International Conference on Case-Based Reasoning, ICCBR-97, pages 599–610, Providence, RI, USA, 25-27 July 1997 1997. IRIT, Univ. Paul Sabatier, Toulouse, France, Springer-Verlag. M1: Copyright 1997, IEE; T3: Case-Based Reasoning Research and Development. Second International Conference on Case-Based Reasoning, ICCBR-97 Proceedings.

[277] D. Dubois, E. Hullermeier, and H. Prade. Fuzzy set-based methods in instance-based reasoning. *IEEE Transactions on Fuzzy Systems*, 10(3):322–32, 06 2002. URL: `http://dx.doi.org/10.1109/TFUZZ.2002.1006435`. M1: Copyright 2002, IEE.

[278] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J. C. Burgelman. Scenarios for ambient intelligence in 2010. Technical report, 2001. URL: `http://cordis.europa.eu/ist/istag-reports.htm`.

[279] H. Duman, H. Hagras, and V. Callaghan. Intelligent association exploration and exploitation of fuzzy agents in ambient intelligent environments. *Journal of Uncertain Systems*, 2(2):133–143, 2008.

[280] V. Duong, Q. Phung, H. Bui, and S. Venkatesh. Human behavior recognition with generic exponential family duration modeling in the hidden semi-markov model. In *18th International Conference on Pattern Recognition, ICPR 2006*, volume 3, pages 202–207, 2006.

[281] S. Dustdar and W. Schreiner. A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30, 2005.

[282] C. Ellis. Team automata for groupware systems. *Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge: the integration challenge*, pages 415–424, 1997.

[283] P. L. Emiliani and C. Stephanidis. Universal access to ambient intelligence environments: Opportunities and challenges for people with disabilities, 2005.

[284] C. Endres. A survey of software infrastructures and frameworks for ubiquitous computing. *Mobile Information Systems*, 1(1):41–80, 2005.

[285] Kynan Eng, Rodney J. Douglas, and Paul F. M. J. Verschure. An interactive space that learns to influence human behavior. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 35(1):66–77, 2005. URL: `http://dx.doi.org/10.1109/TSMCA.2004.838467`. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[286] The Open Source Bpel Engine. *Active BPEL (Version 2.0)*, volume 2006. 2006.

[287] M Ermes, J Parkka, J Mantyjarvi, and I Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled contitions. *IEEE Transactions on Information Technology in Biomedicine*, 12:20–26, 2008.

[288] L. Fan, N. Akhtar, K. A Chew, K. Moessner, and R. Tafazolli. Network composition: a step towards pervasive computing. *Fifth IEE International Conference on 3G Mobile Communication Technologies, 2004.3G 2004*, pages 198–202, 2004.

[289] C. Feier, D. Roman, A. Polleres, J. Domingue, M. Stollberg, and D. Fensel. Towards intelligent web services: Web service modeling ontology (WSMO). *Proceedings of the International Conference on Intelligent Computing (ICIC)*, pages 23–26, 2005.

[290] D. Fensel. Triple space computing. *ISWC 2004 workshop on Semantic Web Services: Preparing to Meet the World of Business Applications*, 2004.

[291] D. Fensel and C. Bussler. The web service modeling framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[292] Dieter Fensel. Triple-Space computing: Semantic web services based on persistent publication of information. In *Intelligence in Communication Systems*, pages 43–53. Springer Berlin / Heidelberg, 2004.

[293] Dieter Fensel, James A Hendler, Henry Lieberman, and Wolfgang Wahlster. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2003.

[294] A. Fernandez, A. Polleres, and S. Ossowski. Towards fine-grained service matchmaking by using concept similarity. pages 31–45.

[295] A. Ferrara. Web services: a process algebra approach. *Proceedings of the 2nd international conference on Service oriented computing*, pages 242–251, 2004.

[296] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. URL: `http://www.industrex.com/dynamic/reading/ics.uci.edu.fielding.dissertation.1col.pdf`.

[297] J. G. P Filho and M. van Sinderen. Web service architectures - semantics and context-awareness issues in web services platform. Technical report, Telematica Instituut, 2003.

[298] Tim Finin, Li Ding, Lina Zhou, and Anupam Joshi. *Social Networking on the Semantic Web*, volume 12. 2005.

[299] Maydene Fisher. *Introduction to Web Services*, volume 2006.

[300] Brad Fitzpatrick. *Memcached Official Site*. 2008.

[301] Foundation for Intelligent Physical Agents. *FIPA Abstract Architecture Specification*. 2002.

[302] UPnP Forum. UPnp device architecture v1.0.1 draft. 2003.

[303] Howard Foster, Sebastian Uchitel, Jeff Magee, and Jeff Kramer. Model-based verification of web service compositions. *Automated Software Engineering*, 00:152, 2003.

[304] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. volume 1193, pages 21–35. Springer Verlag, 1996. ftp://ftp.msci.memphis.edu/comp/caat/agentprog.ps.z.

[305] M. Friedemann and N. Mahmoud. *Pervasive Computing, First International Conference*. Springer-Verlag, 2002.

[306] Michael Friedewald, Olivier Da Costa, Yves Punie, Petteri Alahuhta, and Sirkka Heinonen. Perspectives of ambient intelligence in the home environment. *Telematics and Informatics*, 22(3):221–238, 2005. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[307] M. Friedwald and O. M. Da Costa. Science and technology roadmapping: Ambient intelligence in everyday life (ami@life), June 2003.

[308] M. Friedwald, O. M. Da Costa, Y. Punie, P. Alahuhta, and S. Heinonen. Perspectives of ambient intelligence in the home environment. In *Telematics and Informatics*, volume 22, pages 221–238. Pergamon Press, 2005.

[309] M. Friedwald, E. Vildjiounaite, and D. Wright. Safeguard in a world of ambient intelligence (swami), 2006.

[310] M. Frodigh, P. Johansson, and P. Larsson. Wireless ad hoc networking-The art of networking without a network. *Ericsson Review*, 4:248–263, 2000.

[311] X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL web services. *Proceedings of the 13th conference on World Wide Web*, pages 621–630, 2004.

[312] Keita Fujii and Tatsuya Suda. Dynamic service composition using semantic information. pages 39–48. ACM Press.

[313] Naoki Fukuta and Takayuki Ito. Comparing semantic web service frameworks in a context of auction services. *International Journal of Computer Science and Network Security*, 6(4):100–107, 2006.

[314] Thomas Gabel and Martin Riedmiller. Multi-agent case-based reasoning for cooperative reinforcement learners. In *8th European Conference on Case-Based Reasoning, ECCBR 2006*, volume 4106 NAI, pages 32–46, Fethiye, Turkey, Sep 4-7 2006 2006. Department of Mathematics and Computer Science, Institute of Cognitive Science, University of Osnabruck, 49069 Osnabruck, Germany, Springer Verlag, Heidelberg, D-69121, Germany. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved; T3: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[315] C. Le Gal, J. Martin, A. Lux, and J. L. Crowley. Smartoffice: Design of an intelligent environment. *IEEE Intelligent Systems*, 16(4):60–66, 2001.

[316] A. Galton. *Causal reasoning for alert generation in smart homes*, pages 57–70. Designing Smart Homes. The Role of Artificial Intelligence. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[317] M. Galushka, D. Patterson, and N. Rooney. *Temporal data mining for smart homes*, pages 85–108. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[318] M. Galushka, D. Patterson, and N. Rooney. Temporal data mining for smart homes. pages 85–108. Springer-Verlag, 2006. ID: 169; M1: Copyright 2006, The Institution of Engineering and Technology.

[319] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste. Project aura: Toward Distraction-Free pervasive computing. *IEEE Pervasive Computing*, 1(2):22–31, 2002.

[320] M. Garofalakis, K. P. Brown, M. J. Franklin, J. M. Hellerstein, D. Zhe Wang, E. Michelakis, L. Tancau, E. Wu, S. R. Jeffery, and R. Aipperspach. Probabilistic data management for pervasive computing: The data furnace project, 2006.

[321] M. R Genesereth and R. E Fikes. Knowledge interchange format, version 3.0 reference manual. 1994.

[322] Ion Kepa Gerrikagoitia. *Web Services como modelo de comunicaciones entre aplicaciones. Aplicabilidad, Interoperabilidad y Adopción*. 2006.

[323] M. Ghallab, D. S Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[324] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[325] F. Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, 16:279–305, 1993.

[326] Fausto Giunchiglia and Paolo Traverso. Planning as model checking. pages 1–20. Springer-Verlag.

[327] A. K. Goel and S. Craw. Design, innovation and case-based reasoning. *Knowledge Engineering Review*, 20(3):271–6, 2005. URL: `http://dx.doi.org/10.1017/S0269888906000609`. M1: Copyright 2006, The Institution of Engineering and Technology.

[328] D. Goncalves. Ubiquitous computing and ai towards an inclusive society, 2000.

[329] K. Gopalratnam and D.J. Cook. Active lezi: An incremental parsing algorithm for sequential prediction. *International Journal of Artificial Intelligence*, 14:917–930, 2004.

[330] Imanol Fernández Gorostizaga. Universidad de Deusto, 2008.

[331] B. Gottfried, H. W. Guesgen, and S. Hubner. *Spatiotemporal reasoning for smart homes*, pages 16–34. Designing Smart Homes. The Role of Artificial Intelligence. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[332] B. Gottfried, H. W. Guesgen, and S. Hubner. Spatiotemporal reasoning for smart homes. In *Designing Smart Homes. The Role of Artificial Intelligence*, pages 16–34, 2006.

[333] K. Gottschalk. Web services architecture overview: The next stage of evolution for E-Business. *IBM developerWorks*, 2000.

[334] K. D Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to web services architecture. *IBM Systems Journal*, 41(2):170–177, 2002.

[335] Paul Grace, Gordon S Blair, and Sam Samuel. ReMMoC: a reflective middleware to support mobile client. pages 1170–1187. crossref: DBLP:conf/coopis/2003.

[336] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 2008.

[337] Philip D Gray and Daniel Salber. Modelling and using sensed context information in the design of interactive applications. pages 317–336. Springer-Verlag.

[338] S. Grimm. Discovery - identifying relevant services. pages 211–244. Springer, 2007.

[339] M. D. Gross. Smart house and home automation technologies, 1998.

[340] European Commission I. S. T. Advisory Group. Scenarios for ambient intelligence in 2010. *Technical Report, EU Commission*, 2001. (ISTAG).

[341] European Commission I. S. T. Advisory Group. Ambient intelligence: from vision to reality. *Technical Report, EU Commission*, 2003. (ISTAG).

[342] European Commission I. S. T. Advisory Group. IST research content. *Technical Report, EU Commission*, 2003. (ISTAG).

[343] Object Management Group. CORBA: the common object request broker architecture and specification. 1995.

[344] Object Management Group. Meta object facility (MOF) specification v1.4. Technical report, 2004.

[345] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowl.Acquis.*, 5(2):199–220, 1993.

[346] M. Gruninger. A guide to the ontology of the process specification language. *Handbook on Ontologies.Sringer*, 2003.

[347] M. Gruninger and C. Menzel. Process specification language: Principles and applications. *AI Magazine*, 24(3):63–74, 2003.

[348] T. Gu, H. Pung, and D. Zhang. *A Middleware for Building Context-Aware Mobile Services.* 2004.

[349] Tao Gu, Hung Keng Pung, and Da Qing Zhang. *Toward an OSGi-Based Infrastructure for Context-Aware Applications*, volume 3. 2004.

[350] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *J.Netw.Comput.Appl.*, 28(1):1–18, 2005.

[351] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. An ontology-based context model in intelligent environments. pages 270–275.

[352] Zahia Guessoum. Adaptive agents and multiagent systems. *IEEE Distributed Systems Online*, 5(7):5, 2004. URL: `http://dx.doi.org/10.1109/MDSO.2004.10`. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[353] Dominique Guinard, Vlad Trifa, and Erik Wilde. A resource oriented architecture for the web of things. In *Proceedings of Internet of Things 2010 International Conference (IoT 2010)*, Tokyo, Japan, November 2010.

[354] Richard Moe Gustavsen. Condor - an application framework for mobility-based context-aware applications. *Concepts and models for ubiquitous computing workshop at Ubicomp 2002*, 2002.

[355] A Gómez-Goiri, D López de Ipiña, A Aztiria, A Urbieta, and J Bastida. Ismed: Intelligent semantic middleware for embedded devices, informe científico anual 2008. Technical report, Universidad de Deusto, Mondragon Unibertsitatea, 2008.

[356] H. Hagras, V. Callaghan, M. Colley, and G. Clarke. A hierarchical fuzzy-genetic multi-agent architecture for intelligent buildings online learning, adaptation and control. In *Information Sciences*, volume 150, pages 33–57, 2003. URL: `http://dx.doi.org/10.1016/S0020-0255(02)00368-7`.

[357] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Systems*, 19(6):12–20, 2004.

[358] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. *Creating an Ambient-Intelligence Environment Using Embedded Agents*, volume 19. 2004. ID: 10.

[359] F. Hakimpour, J. Domingue, E. Motta, L. Cabral, and Y. Lei. Integration of OWL-S into IRS-III. *1st AKT Workshop on Semantic Web Services*, 122, 2004.

[360] T. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann, 2001.

[361] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. *Proceedings of the Fourteenth Australasian database conference on Database technologies 2003*, 17:191–200, 2003.

[362] C. Hamblin. Instants and intervals. In J. Fraser, editor, *The study of time*, pages 325–331, 1972.

[363] Joohyun Han, Yongyun Cho, and Jaeyoung Choi. Context-Aware workflow language based on web services for ubiquitous computing. pages 1008–1017. crossref: DBLP:conf/iccsa/2005-2.

[364] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2-3):187–197, 2002. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[365] Ronny Haryanto. *Context-Awareness in Smart Homes to Support Independent Living*. 2005.

[366] E. O. Heierman and D. J. Cook. Improving home automation by discovering regularly occurring device usage patterns. In *Third IEEE International Conference on Data Mining*, pages 537–540, 2002.

[367] A. Held, S. Buchholz, and A. Schill. Modeling of context information for pervasive computing applications. *Proc.of the World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002 / ISAS 2002)*, 2002.

[368] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. volume 2414, pages 167–180. Springer-Verlag.

[369] Martin Hepp. Semantic web and semantic web services: Father and son or indivisible twins? *IEEE Internet Computing*, 10(2):85–88, 2006. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[370] Ramon Hervás, Jose Bravo, Salvador N Nava, and Gabriel Chavira. Interacción natural en ambientes inteligentes a través de roles en mosaicos de visualización. volume 1. José Bravo.

[371] Ramon Hervás, Salvador V Nava, Gabriel Chavira, and Jose Bravo. Modelado de contexto: Una ontología adaptativa al usuario en ambientes inteligentes. volume 1, pages 167–177. José Bravo.

[372] C. Hesselman, A. Tokmakoff, P. Pawar, and S. Iacob. Discovery and composition of services for Context-Aware systems. volume 4272, pages 67–81.

[373] E. F Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications, 2003.

[374] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz. Sensing techniques for mobile interaction. pages 91–100. Association for Computing Machinery Microsoft Research, Redmond, WA 98052, United States. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[375] Sebastian Hinz, Karsten Schmidt, and Christian Stahl. Transforming BPEL to petri nets. pages 220–235. crossref: DBLP:conf/bpm/2005.

[376] C. A. R Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[377] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-Awareness on mobile devices - the hydrogen approach. page 292.

[378] R. Hogg, J. McKean, and Allen Craig. *Introduction to Mathematical Statistics*, pages 359–364. Pearson Prentice Hall, 2005.

[379] G. J Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison Wesley Publishing Company, 2003.

[380] Jason I Hong. The context fabric: an infrastructure for context-aware computing. volume 2, pages 554–555. ACM Press.

[381] I. Horrocks. DAML+OIL: a reason-able web ontology language. pages 2–13. Springer-Verlag Dept. of Comput. Sci., Manchester Univ., UK. M1: Copyright 2002, IEE.

[382] I. Horrocks, P. F Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: a semantic web rule language combining OWL and RuleML. *W3C Member Submission*, 21, 2004.

[383] M. Huber. Cse 4392 / cse 5392 - smart home technologies, 2006.

[384] Richard Hull and Jianwen Su. Tools for composite web services: a short overview. *SIGMOD Rec.*, 34(2):86–95, 2005.

[385] Gwang hun Kim, Do hyun Kim, XuanTung Hoang, and Younghee Lee. Group-aware service discovery using effect ontology for conflict resolution in ubiquitous environment.

[386] Gwang hun Kim, Do hyun Kim, XuanTung Hoang, Younghee Lee, and Gab soo Lee. Semantic service discovery using effect ontology for appliance service in ubiquitous environment.

[387] Pertti Huuskonen. Run to the hills! ubiquitous computing meltdown. In Juan Carlos Augusto and Daniel Shapiro, editors, *Advances in Ambient Intelligence*, volume 164 of *Frontiers in Artificial Intelligence and Applications*, pages 157 – 172. IOS Press, 2007.

[388] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen. Experiences in using CC/PP in Context-Aware systems. *Proceedings of the 4th International Conference on Mobile data Management (MDM 2003)*, 2893:224–235, 2003.

[389] Jadwiga Indulska and Peter Sutton. Location management in pervasive systems. pages 143–151. Australian Computer Society, Inc.

[390] M. Ingstrup and K. M Hansen. Palpable assemblies: Dynamic service composition in ubiquitous computing. *Proceedings of Software Engineering and Knowledge Engingeering*, 2005, 2005.

[391] V. Issarny, D. Sacchetti, F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, and A. Talamona. Developing ambient intelligence systems: A solution based on web services. *Automated Software Engineering*, 12(1):101–137, 2005.

[392] V. Issarny, D. Sacchetti, F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, and A. Talamona. Developing ambient intelligence systems: A solution based on web services, 2005.

[393] R.M. van Eijk J. van Diggelen, R.J. Beun and P.J. Werkhoven. Efficient semantic information exchange for ambient intelligence. *The computer journal*, 2009.

[394] Michel Jaczynski. Framework for the management of past experiences with time-extended situations. In *Proceedings of the 1997 6th International Conference on Information and Knowledge Management, CIKM'97*, pages 32–39, Las Vegas, NV, USA, Nov 10-14 1997 1997. INRIA Sophia-Antipolis, Sophia Antipolis, Fr, ACM, New York, NY, USA. URL: `http://dx.doi.org/10.1145/266714.266851`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: International Conference on Information and Knowledge Management, Proceedings.

[395] M. C Jaeger, G. Rojec-Goldmann, C. Liebetruth, G. Muhl, and K. Geihs. Ranked matching for service descriptions using OWL-S. *Kommunikation in verteilten Systemen (KiVS 2005), Informatik Aktuell*, pages 91–102, 2005.

[396] V. R. Jakkula and D. J. Cook. Using temporal relations in smart environment data for activity prediction. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[397] V. R. Jakkula, A. S. Crandall, and D. J. Cook. Knowledge discovery in entity based smart environment resident data using temporal relation based data mining. In *7th IEEE International Conference on DataMining*, pages 625–630, 2007.

[398] J. Janecek. Efficient SOAP processing in embedded systems. page 128. IEEE Computer Society.

[399] J.S.R. Jang. Anfis: Adaptive-network-based fuzzy inference system. *IEEE Systems, man and cybernetics*, 23:665–684, 1993.

[400] Yang-Seung Jeon, Eun-Ha Song, Minyi Guo, Laurence Tianruo Yang, Young-Sik Jeong, Jin-Tak Choi, and Sung-Kook Han. Ontology-Based composition of web services for ubiquitous computing. pages 559–568. crossref: DBLP:conf/ispa/2006w.

[401] Li Jiang, Da-You Liu, and Bo Yang. Smart home research. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, volume 2, pages 659–663, 2004.

[402] Long Jin, Yongmi Lee, Sungbo Seo, and Keun Ho Ryu. Discovery of temporal frequent patterns using tfp-tree. In *7th International Conference on Advances in Web-Age Information Management, WAIM 2006*, volume 4016 NCS, pages 349–361, Hong Kong, China, Jun 17-19 2006 2006.

[403] D. B Johnson. Routing in ad hoc networks of mobile hosts. pages 158–163. IEEE Computer Society, Santa Cruz, CA.

[404] Gerald Kaefer, Reiner Schmid, Guenter Prochart, and Reinhold Weiss. Framework for dynamic Resource-Constrained service composition for mobile ad hoc networks. *8th Annual Conference on Ubiquitous Computing, Workshop on System Support for Ubiquitous Computing*, 2006.

[405] L. Kagal, V. Korolev, H. Chen, A. Joshi, and T. Finin. Centaurus: A framework for indoor mobile services. *International Conference on Distributed Computing Systems*, 2001.

[406] A. Kainulainen, M. Turunen, J. Hakulinen, E. P. Salonen, P. Prusi, and L. Helin. A speechbased and auditory ubiquitous office environment. In *10th International Conference on Speech and Computer (SPECOM)*, pages 231–234, 2005.

[407] D. N Kalofonos, F. D Reynolds, and T. R Nrc. Task-Driven End-User programming of smart spaces using mobile devices. Technical report, Nokia, 2006.

[408] A. Kameas, S. Bellis, I. Mavrommati, K. Delaney, M. Colley, and A. Pounds-Cornish. An architecture that treats everyday objects as communicating tangible components.

[409] Saehoon Kang, Woohyun Kim, Dongman Lee, and Younghee Lee. Group context-aware service discovery for supporting continuous service availability.

[410] D. Karagiannis. BPMS: business process management systems. *ACM SIGOIS Bulletin*, 16(1):10–13, 1995.

[411] K. Karimi and H. J. Hamilton. Temporal rules and temporal decision trees: A c4.5 approach. Technical Report CS-2001-02, 2001. URL: `http://flash.lakeheadu.ca/~kkarimi/pubs/tech-2001-02.pdf`.

[412] N. Kasabov. Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. volume 31, pages 902–918. IEEE Systems, Man, and Cybernetics Society, 2001.

[413] F. Kaufer and M. Klusch. Performance of hybrid WSML service matching with WSMO-MX: preliminary results. pages 63–77.

[414] F. Kaufer and M. Klusch. WSMO-MX: a logic programming based hybrid service matchmaker. IEEE CS Press.

[415] H. Kautz, D. Fox, O. Etzioni, G. Borriello, and L. Arnstein. An overview of the assisted cognition project. In *Proceedings of the AAAI Workshop on Automation as Caregiver*, pages 60–65. AAAI Press, 2002.

[416] N. Kavantzas, D. Burdett, and G. Ritzinger. Web services choreography description language version 1.0 (W3C working draft). *W3C*, 2004.

[417] D. K Kaynar, N. A Lynch, R. Segala, and F. W Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lecture on Computer Science, 2006.

[418] Raman Kazhamiakin, Paritosh K Pandya, and M. Pistore. Modelling and analysis of time-related properties in web service compositions. *In Proceedings of the 1st International Workshop onEngineering Service Compositions*, 2005.

[419] Raman Kazhamiakin, Paritosh K Pandya, and Marco Pistore. Timed modelling and analysis in web service compositions. pages 840–846. crossref: DBLP:conf/IEEEares/2006.

[420] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic location of services. pages 38–49. Springer.

[421] U. Keller, H. Lausen, and M. Stollberg. On the semantics of functional descriptions of web services. volume 4011, page 605. Springer.

[422] M. Khedr and A. Karmouch. Negotiating context information in Context-Aware systems. 19(6):21–29, 2004.

[423] M. Khedr and A. Karmouch. ACAI: agent-based context-aware infrastructure for spontaneous applications. *J.Netw.Comput.Appl.*, 28(1):19–44, 2005.

[424] C. Kiefer and A. Bernstein. The creation and evaluation of iSPARQL strategies for matchmaking. volume 5021, page 463. Springer.

[425] B. Kiepuszewski, A. H. M ter Hofstede, W. M. P van der Aalst, and Fit-Tr Qut Technical report. Fundamentals of control flow in workflows (Revised version). Technical report, Queensland University of Technology, 2002.

[426] M. Kifer and G. Lausen. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. pages 134–146. ACM Press New York.

[427] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the Association for Computing Machinery*, 42(4):741–843, 1995. M1: Copyright 1995, IEE.

[428] Woohyun Kim, Saehoon Kang, Younghee Lee, Dongman Lee, and Inyoung Ko. Activity Policy-Based service discovery for pervasive computing. pages 756–768. crossref: DBLP:conf/edbtw/2006.

[429] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, and H. Morris. People, places, things: Web presence for the real world. *Mobile Networks and Applications*, 7(5):365–376, 2002.

[430] M. Klein and B. Konig-Ries. Coupled signature and specification matching for automatic service binding. Springer.

[431] Mark Klein and Abraham Bernstein. Toward High-Precision service retrieval. *IEEE Internet Computing*, 8(1):30–36, 2004.

[432] M. Klusch. Semantic web service coordination. Birkhauser Verlag, Springer, 2008.

[433] M. Klusch and B. Fries. Hybrid OWL-S service retrieval with OWLS-MX: benefits and pitfalls. pages 47–61.

[434] M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with OWLS-MX. pages 915–922. ACM.

[435] M. Klusch and P. Kapahne. Semantic web service selection with SAWSDL-MX. pages 3–17.

[436] Jonathan Knudsen. *Getting Started with JXTA for J2ME*. 2002.

[437] C. Lee. S Ko, W. Lee S Lee, and A. Helal. Context-Aware service composition for mobile network environments.

[438] A. Kofod-Petersen. Challenges in case-based reasoning for context awareness in ambient intelligence systems. In *1st Workshop on Case-based Reasoning and Context Awareness*, 2006.

[439] A. Kofod-Petersen and A. Aamodt. Contextualised ambient intelligence through case-based reasoning. Advances in Case-Based Reasoning. 8th European Conference, ECCBR 2006, pages 211–25, Fethiye, Turkey, 4-7 Sept. 2006 2006.

[440] Anders Kofod-Petersen and Marius Mikalsen. An architecture supporting implementation of Context-Aware services. pages 31–42. HIIT Publications.

[441] A. Kofos-Petersen and A. Aamodt. Contextualised ambient intelligence through case-based reasoning, 2006.

[442] C. Y Kong, Cho-Li Wang, and Francis C. M Lau. Ontology mapping in pervasive computing environment. pages 1014–1023. crossref: DBLP:conf/euc/2004.

[443] J. Kopecky, E. M Simperl, and D. Fensel. Semantic web offer discovery. pages 3–13.

[444] J. Kopecký. Aligning WSMO and WSDL-S. *WSMO working draft D30v0*, 1(5), 2005.

[445] Jacek Kopecký, Dumitru Roman, Matthew Moran, and Dieter Fensel. Semantic web services grounding. page 127. crossref: DBLP:conf/aict/2006.

[446] Teemu Koponen and Teemupekka Virtanen. A service discovery: A service broker approach. *hicss*, 09:90284b, 2004.

[447] D. Kourtesis and I. Paraskakis. Combining SAWSDL, OWL-DL and UDDI for semantically enhanced web service discovery. volume 5021, pages 614–628. Springer.

[448] A. Koutsorodi, E. Adamopoulou, K. Demestichas, and M. Theologou. User profiling and preference modelling in 4g terminals, 2006.

[449] Gerhard Kramler and Werner Retschitzegger. Specification of interorganizational workflows - a comparison of approaches. Technical report, Institute of Software Technology and Interactive Systems, Business Informatics Group, Vienna University of Technology, 2002.

[450] Reto Krummenacher, Michael Fried, and Daniel Blunder. tsc++ user guide. Technical report, STI Innsbruck, 2005.

[451] Reto Krummenacher, Elena Simperl, Doug Foxvog, Vassil Momtchev, Dario Cerizza, Lyndon Nixon, Davide Cerri, Brahmananda Sapkota, Kia Teymourian, Philipp Obermeier, Daniel Martin, Hans Moritsch, Omair Shafiq, and David de Francisco. D6.5 towards a scalable triple space. Technical report, TripCom, 2008.

[452] L. Kukolich and R. Lipmann. Lnknet user's guide.

[453] A. Kulkarni. A reactive behavioral system for the intelligent room.

[454] N. Kushwaha, M. Kim, D. Y. Kim, and W. Cho. An intelligent agent for ubiquitous computing environments: Smart home ut-agent. In *Proceedings of the 2nd IEEE Workshop on Software Technologies for future Embedded and Ubiquitous Systems*, pages 157–159, 2004.

[455] Neeraj Kushwaha, Minkoo Kim, Dong Yoon Kim, and We-Duke Cho. An intelligent agent for ubiquitous computing environments: Smart home UT-AGENT. pages 157–159. IEEE Computer Society, Los Alamitos, CA 90720-1314, United States Grad. Sch. of Info. and Commun., Ajou University, Suwon, South Korea. ID: 122; Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: Proceedings - Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems.

[456] U. Kuster, H. Lausen, and B. Konig-Ries. Evaluation of semantic service Discovery-A survey and directions for future research. pages 37–53.

[457] Ulrich Kuster, Mirco Stern, and Birgitta König-Ries. A classification of issues and approaches in automatic service composition.

[458] OhByung Kwon. Multi-agent system approach to context-aware coordinated web services under general market mechanism. *Decision Support Systems*, 41(2):380–399, 2006. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[459] U. Küster, B. König-Ries, M. Stern, and M. Klein. DIANE: an integrated approach to automated service discovery, matchmaking and composition. pages 1033–1042. ACM Press New York, NY, USA.

[460] Ontotext Lab. Fact sheet. Technical report, Sirma Group Corp., 2007.

[461] M. Lamming and M. Flynn. Forget-me-not: Intimate computing in support of human memory. *Proceedings of FRIEND21, Int'l Symp. Next Generation Human Interface*, pages 94–103, 1994.

[462] Jason I. Hong Landay and James A. An infrastructure approach to Context-Aware computing. *Human-Computer Interaction*, 16(2-4):287–303, 2001.

[463] R. Lara, M. A Corella, and P. Castells. A flexible model for web service discovery. pages 51–66.

[464] R. Lara, D. Roman, A. Polleres, and D. Fensel. A conceptual comparison of WSMO and OWL-S. *Proceedings of the European Conference on Web Services (ECOWS 2004)*, 2004.

[465] K. G Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):134–152, 1997.

[466] O. Lassila. Semantic web, quo vadis?

[467] O. Lassila and R. R Swick. Resource description framework (RDF) model and syntax specification. *W3C Recommendation*, 22:2004–2003, 1999.

[468] Ora Lassila. *Semantic Web, quo vadis?* Nokia Research Center, Cambridge, MA, 2006.

[469] Ora Lassila and Mark Adler. Semantic gadgets: Ubiquitous computing meets the semantic web. pages 363–376.

[470] Patrick Laube and Ross S. Purves. An approach to evaluating motion pattern detection techniques in spatio-temporal data. *Computers, Environment and Urban Systems*, 30(3):347–374, 2006. URL: `http://dx.doi.org/10.1016/j.compenvurbsys.2005.09.001`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[471] H. Lausen and D. Innsbruck. *Semantic Annotations for WSDL (SAWSDL)*, volume 2006. 2006.

[472] D. N Le and A. E. S Goh. FuzMOD: a fuzzy multi-ontology web service discovery system. pages 197–203.

[473] D. N Le, T. M Hang, and A. E. S Goh. MOD - a Multi-Ontology discovery system. pages 13–17.

[474] D. Leake. Cbr in context: The present and future, 1996.

[475] D. Leake, A. Maguitman, and T. Reichherzer. *Cases, context, and comfort: opportunities for case-based reasoning in smart homes*, pages 109–131. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[476] D. Leake, A. Maguitman, and T. Reichherzer. Cases, context, and comfort: opportunities for case-based reasoning in smart homes. pages 109–31. Springer-Verlag, 2006. ID: 165; M1: Copyright 2006, The Institution of Engineering and Technology.

[477] Choonhwa Lee and Sumi Helal. Context attributes: An approach to enable context-awareness for service discovery. page 22. IEEE Computer Society.

[478] George Lee, Peyman Faratin, Steven Bauer, and John Wroclawski. A User-Guided cognitive agent for network service selection in pervasive computing environments. page 219. IEEE Computer Society.

[479] Sun Young Lee, Jong Yun Lee, and Byung Il Lee. Service composition techniques using data mining for ubiquitous computing environments. *International Journal of Computer Science and Network Security*, 6(9):110–117, 2006.

[480] Yugyung Lee, Soon Ae Chun, and James Geller. Web-Based semantic pervasive computing services. *IEEE Intelligent Informatics Bulletin*, 4:4–15, 2004.

[481] H. J Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B Scherl. GOLOG: a logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1):59–83, 1997.

[482] F. Leymann. Web services flow language (WSFL), version 1.0. *IBM Software Group*, 2001.

[483] D. Li, L. Jiang, and J. S. Deogun. Temporal knowledge discovery with infrequent episodes, 2002.

[484] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. *Int.J.Electron.Commerce*, 8(4):39–60, 2004.

[485] Yan Li, Simon Chi-Keung Shiu, Sankar Kumar Pal, and James Nga-Kwok Liu. Case-base maintenance using soft computing techniques. In *2003 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1768–1773, Xián, China, Nov 2-5 2003 2003. Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong, Institute of Electrical and Electronics Engineers Inc. URL: `http://dx.doi.org/10.1109/ICMLC.2003.1259783`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: International Conference on Machine Learning and Cybernetics.

[486] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Behavior recognition in assisted cognition. In *Proceedings of the IAAA-04 Workshop on Supervisory Control of Learning and Adaptive Systems*, pages 41–42, 2004.

[487] M. Likhachev and R. C. Arkin. Spatio-temporal case-based reasoning for behavioral selection. volume 2, pages 1627–1634, Seoul, May 21-26 2001 2001. URL: `http://dx.doi.org/10.1109/ROBOT.2001.932844`.

[488] Maxim Likhachev, Michael Kaess, and Ronald C. Arkin. Learning behavioral parametrization using spatio-temporal case-based reasoning. In *2002 IEEE International Conference on Robotics and Automation*, volume 2, pages 1282–1289, Washington, DC, United States, May 11-15 2002 2002. Mobile Robot Laboratory, College of Computing, Georgia Institute of Technology, Atlanta, GA, United States, Institute of Electrical and Electronics Engineers Inc. URL: `http://dx.doi.org/10.1109/ROBOT.2002.1014719`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: Proceedings - IEEE International Conference on Robotics and Automation.

[489] H. Lim, Y. Teo, P. Mukherjee, V. Lam, W. Wong, and S. See. Sensor grid: Integration of wireless sensor networks and the grid. *IEEE Conference on local computer networks*, 30:91–99, 2005.

[490] L. Lin and I. B Arpinar. Discovering semantic relations between web services using their pre and Post-Conditions. pages 237–238.

[491] L. Lin and I. B Arpinar. Discovery of semantic relations between web services. pages 357–364. IEEE Computer Society Washington, DC, USA.

[492] W. Lin, M. A. Orgun, and G. J. Willians. Mining temporal patterns from health care data, 2002.

[493] M. Lindwer, D. Marculescu, T. Basten, R. Zimmennann, R. Marculescu, S. Jung, E. Cantatore, P. Res, and N. Eindhoven. Ambient intelligence visions and achievements: linking abstract ideas to real-world concepts. *Design, Automation and Test in Europe Conference and Exhibition*, pages 10–15, 2003.

[494] J. Liu, J. C. Augusto, and H. Wang. Considerations on uncertain spatio-temporal reasoning in smart home systems. In *7th International Conference on Applied Artificial Intelligence (FLINS 2006)*. World Scientific, 2006.

[495] J. Liu and V. Issarny. Signal strength based service discovery (S3D) in mobile ad hoc networks.

[496] J. Liu and V. Issarny. QoS-aware service location in mobile ad hoc networks. *Proceedings of the 5th IEEE International Conference on Mobile Data Management (MDM'04)*, pages 224–235, 2004.

[497] S. Liu, R. Khalaf, and F. Curbera. From DAML-S processes to BPEL4WS. *14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications*, pages 77–84, 2004.

[498] A. Lopez, L. Sanchez, F. Doctor, H. Hagras, and V. Callaghan. An evolutionary algorithm for the off-line data driven generation of fuzzy controllers for intelligent buildings. In *IEEE Transactions on systems, man and cybernetics*, volume 1, pages 42–47, 2004.

[499] R. Lopez de Mantaras and E. Armengol. Machine learning from examples: inductive and lazy methods. *Data and Knowledge Engineering*, 25(1):99–123, 03 1998. URL: `http://dx.doi.org/10.1016/S0169-023X(97)00053-0`.

[500] E. F. LoPresti. Assistive technology for cognitive rehabilitation: State of the art, 2004.

[501] K. Lorincz, D. J Malan, T. R. F Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor networks for emergency response: challenges and opportunities. *Pervasive Computing, IEEE*, 3(4):16–23, 2004.

[502] M. Martinez Luaces, C. Gayoso, and S. Suarez. Intelligent virtual environments: Operating conditioning and observational learning in agents using neural networks. In *2nd International Conference on intelligent environments*, pages 127–134, 2006.

[503] Ramón Hervás Lucas. *Modelado de contexto para la visualización de información en ambientes inteligentes*. PhD thesis, 2008.

[504] S. Luhr, G. West, and S. Venkatesh. Recognition of emergent human behaviour in a smart home: A data mining approach. *Pervasive and Mobile Computing*, 3(2):95–116, 2007. URL: `http://dx.doi.org/10.1016/j.pmcj.2006.08.002`.

[505] N. A Lynch. Input/Output automata: Basic, timed, hybrid, probabilistic, dynamic. *Proceedings of the 14th International Conference on Concurrency Theory*, 3:191–192, 2003.

[506] N. A Lynch and M. R Tuttle. *An Introduction to Input/output Automata*. Laboratory for Computer Science, Massachusetts Institute of Technology, 1988.

[507] Kalle Lyytinen and Youngjin Yoo. Issues and challenges in ubiquitous computing. *Communications of the ACM*, 45(12):62–65, 2002.

[508] A. López, L. Sánchez, F. Doctor, H. Hagras, and V. Callaghan. *An Evolutionary Algorithm for the Off-Line Data Driven Generation of Fuzzy Controllers for Intelligent Buildings*, volume 1. 2004. ID: 9.

[509] T. Ma, Y. Kim, Q. Ma, M. Tang, and W. Zhou. Context-aware implementation based on cbr for smart home. volume 4, pages 112–115, Montreal, QC, Canada, Aug 22-24 2005 2005. URL: `http://dx.doi.org/10.1109/WIMOB.2005.1512957`.

[510] Zakaria Maamar, Soraya Kouadri Mostefaoui, and Hamdi Yahyaoui. Toward an Agent-Based and Context-Oriented approach for web services composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):686–697, 2005.

[511] Guilherme Bertoni Machado, Frank Siqueira, Robinson Mittmann, and Carlos Augusto Vieira e Vieira. Embedded systems integration using web services. page 18. IEEE Computer Society.

[512] E. Maeda and Y. Minami. Steps towards ambient intelligence, 2006.

[513] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, 2002.

[514] Shalil Majithia, David W Walker, and W. A Gray. A framework for automated service composition in Service-Oriented architectures. pages 269–283. crossref: DBLP:conf/esws/2004.

[515] Qusay H Mamoud. *Getting Started With JavaSpaces Technology: Beyond Conventional Distributed Programming Paradigms*. 2005.

[516] H. De Man. Ambient intelligence: Gigascale and nanoscale realities, 2005.

[517] D. J Mandell and S. A McIlraith. A Bottom-Up approach to automating web service discovery, customization, and semantic translation. *Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web*, 2003.

[518] Daniel J Mandell and Sheila A McIlraith. Adapting BPEL4WS for the semantic web: The Bottom-Up approach to web service interoperation. pages 227–241. crossref: DBLP:conf/semweb/2003.

[519] J. Mankoff and B. Schilit. Supporting knowledge workers beyond the desktop with palplates. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 550–551, 1997.

[520] William C Mann and Sandra A Thompson. Rhetorical structure theory: A theory of text organization. volume 8, pages 243–281. USC Information Sciences Institute, 1987.

[521] H. Mannila, H. Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–89, 1997. URL: `http://dx.doi.org/10.1023/A:1009748302351`. M1: Copyright 1998, IEE.

[522] Frank Manola and Eric Miller. *RDF Primer*. 2004. howpublished: "World Wide Web Consortium, Recommendation REC-rdf-primer-20040210".

[523] R. L. De Mantaras, P. Cunningham, and P. Perner. Emergent case-based reasoning applications. *Knowledge Engineering Review*, 20(3):325–8, 2005. URL: `http://dx.doi.org/10.1017/S0269888906000658`. M1: Copyright 2006, The Institution of Engineering and Technology.

[524] R. L. De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson. Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review*, 20(3):215–40, 2005. URL: `http://dx.doi.org/10.1017/S0269888906000646`. M1: Copyright 2006, The Institution of Engineering and Technology.

[525] J Manyika and H Durrant-Whyte. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Ellis Horwood, 1994.

[526] D. Marculescu, N. H. Zamora, P. Stanley-Marbell, and R. Marculescu. Fault-tolerant techniques for ambient intelligent distributed systems, 2003.

[527] Dave Marshall. Distributed reasoning. 2010. URL: `http://www.cs.cf.ac.uk/Dave/AI2/node103.html#SECTION00012100000000000000`.

[528] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, and T. Payne. OWL-S: semantic markup for web services. *W3C Member Submission*, 22, 2004.

[529] T. Martin, B. Majeed, B. E. Lee, and N. Clarke. Fuzzy ambient intelligence for next generation telecare, July 16-21 2006.

[530] Trevor Martin. Fuzzy ambient intelligence in home telecare. In *19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006*, volume 4031 NAI, pages 12–13, Annecy, France, Jun 27-30 2006 2006. Department of Engineering Mathematics, University of Bristol, Bristol BS8 1TR, United Kingdom, Springer Verlag, Heidelberg, D-69121, Germany. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[531] S. Marzano. The home of the future: Smarter but simple. In *http://www.design.philips.com/about/design/designnewscenter/news/pressbackgrounders/article-15027.page*.

[532] Ryusuke Masuoka, Yannis Labrou, Bijan Parsia, and Evren Sirin. *Ontology-Enabled Pervasive Computing Applications*, volume 18. 2003.

[533] Ryusuke Masuoka, Bijan Parsia, and Yannis Labrou. Task computing - the semantic web meets pervasive computing. pages 866–881. crossref: DBLP:conf/semweb/2003.

[534] U. Maurer, A. Rowe, A. Smailagic, and D. Siewiore. Location and activity recognition using ewatch: A wearable sensor platform, 2006.

[535] U. Maurer, A. Smailagic, D. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, pages 99–102, 2006.

[536] B. McBride. Jena: a semantic web toolkit. *IEEE Internet Computing*, 6(6):55–9, 2002. M1: Copyright 2002, IEE.

[537] J. McCan. A design process for the research and development of smart clothes with embedded technologies with potential to enhance quality of life for older people, 2006.

[538] H. McCarthy. Notes on formalizing context. volume vol.1, pages 555–60. Morgan Kaufmann Publishers Dept. of Comput. Sci., Stanford Univ., CA, USA.

[539] J. McCarthy and S. Buvac. Formalizing context (Expanded notes). *Working Papers of the AAAI Fall Symposium on Context in Knoledge Representation and Natural Language*, 1997.

[540] D. McDermott. PDDL-the planning domain definition language. *The AIPS-98 Planning Competition Comitee*, 1998.

[541] D. McDermott. Estimated-Regression planning for interactions with web services. *Proceedings of the AI Planning Systems Conference*, 2002.

[542] D. McDermott. DRS: a set of conventions for representing logical languages in RDF. 2004.

[543] R. McGrath. *Semantic Infrastructure for a Ubiquitous Computing Environment.* 2005.

[544] Deborah L McGuinness and Frank Van Harmelen. *OWL Web Ontology Language Overview.* 2004.

[545] S. McIlraith and D. Mandell. Comparison of DAML-S and BPEL4WS. 2002.

[546] S. McIlraith and T. Son. Adapting golog for composition of semantic web services. *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning*, pages 482–493, 2002.

[547] Sheila A McIlraith, Tran Cao Zeng, and Honglei Zeng. Semantic web services. *IEEE Intelligent Systems and Their Applications*, 16(2):46–53, 2001.

[548] B. Medjahed, A. Bouguettaya, and A. K Elmagarmid. Composing web services on the semantic web. *The VLDB Journal The International Journal on Very Large Data Bases*, 12(4):333–351, 2003.

[549] R. Mehta, D. J. Cook, and L. B. Holder. Identifying inhabitants of an intelligent environment using a graph-based data mining system, 2002.

[550] Wolfgang Meier. eXist: an open source native XML database. volume 2593, pages 169–183. Springer LNCS Series.

[551] I. Meiri. Temporal reasoning: A constraint-based approach. Technical Report R-173, 1992.

[552] Jan Mendling and Martin Muller. A comparison of BPML and BPEL4WS. pages 305–316. crossref: DBLP:conf/bxml/2003.

[553] Jan Mendling, Gustaf Neumann, and Markus Nuttgens. A comparison of XML interchange formats for business process modelling. pages 129–140. crossref: DBLP:conf/emisa/2004.

[554] I. Millard, D. De Roure, and N. Shadbolt. The use of ontologies in contextually aware environments. pages 42–47.

[555] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1989.

[556] Cai Min and Frank Martin. *RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network*. ACM, 2004. 988760 650-657.

[557] A. Mingkhwan, P. Fergus, O. AbuelmaÁtti, M. Merabti, B. Askwith, and M. B Hanneghan. Dynamic service composition in home appliance networks. *Multimedia Tools and Applications*, 29(3):257–284, 2006.

[558] T. M. Mitchell. *Machine Learning*. The McGraw-Hill and MIT Press, 1997.

[559] N. Mitra. SOAP version 1.2 part 0: Primer. *W3C*, 2003.

[560] S. Ben Mokhtar. *Semantic Middleware for Service-Oriented Pervasive Computing*. 2007.

[561] S. Ben Mokhtar, Damien Fournier, Nikolaos Georgantas, and Valérie Issarny. Context-Aware service composition in pervasive computing environments. pages 129–144. crossref: DBLP:conf/rise/2005.

[562] S. Ben Mokhtar, N. Georgantas, and V. Issarny. Ad hoc composition of user tasks in pervasive computing environments. *Proceedings of the 4th international Workshop on Software Composition co-located with ETAPS'05*, 3628, 2005.

[563] S. Ben Mokhtar, N. Georgantas, and V. Issarny. Cocoa: Conversation-based service composition in pervasive computing environments. *Proceedings of the IEEE International Conference on Pervasive Services*, 2006.

[564] S. Ben Mokhtar, N. Georgantas, and V. Issarny. COCOA: COnversation-based service composition in PervAsive computing environments with QoS support. *Journal Of System and Software*, 80(12):1941–1955, 2007.

[565] S. Ben Mokhtar, Anupam Kaul, Nikolaos Georgantas, and Valérie Issarny. Efficient semantic service discovery in pervasive computing environments. pages 240–259.

[566] S. Ben Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers. EASY: efficient SemAntic service DiscoverY in pervasive computing environments with QoS and context support. *Journal Of System and Software*, 81(5):785–808, 2008.

[567] S. Ben Mokhtar, P. G Raverdy, R. Cardoso, A. Urbieta, and N. Georgantas. Discovering social services in pervasive environments with privacy.

[568] S. Ben Mokhtar, P. G Raverdy, A. Urbieta, and R. Cardoso. Interoperable semantic & syntactic service matching for ambient computing environment.

[569] S. Ben Mokhtar, P. G Raverdy, A. Urbieta, and R. Cardoso. Interoperable semantic & syntactic service matching for ambient computing environments. *International Journal of Ambient Computing and Intelligence (IJACI)*, 2009.

[570] S. Ben Mokhtar, P. G Raverdy, A. Urbieta, R. Cardoso, N. Georgantas, and V. Issarny. Interoperable Semantic-Syntactic service discovery in Service-Oriented overlays for pervasive environments.

[571] T. P Moran and P. Dourish. Introduction to this special issue on Context-Aware computing. *Human-Computer Interaction*, 16(2, 3 and 4):87–95, 2001.

[572] Ghita Kouadri Mostefaoui, Jacques Pasquier-Rocha, and Patrick Brezillon. Context-aware computing: A guide for the pervasive computing community. pages 39–48. Institute of Electrical and Electronics Engineers Inc., New York, NY 10016-5997, United States University of Fribourg, Software Engineering Group, CH-1700 Fribourg, Switzerland. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[573] S. K Mostefaoui, A. Tafat-Bouzid, and B. Hirsbrunner. Using context information for service discovery and composition. *Proceedings of the Fifth International Conference on Information Integration and Web-based Applications and Services*, 3:15–17, 2003.

[574] E. Motta. An overview of the OCML modelling language. *8th Workshop on Knowledge Engineering Methods and Languages*, pages 21–22, 1998.

[575] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. IRS-II: a framework and infrastructure for semantic web services. *2nd International Semantic Web Conference*, pages 20–23, 2003.

[576] M. C. Mozer. The neural network house: an environment that adapts to its inhabitants. In M. Coen, editor, *Proceedings of the AAAI Spring Symposium on Intelligent Environments*, pages 110–114. AAAI Press, 1998.

[577] M. C. Mozer. *Lessons from an Adaptive Home*, pages 273–298. Smart Environments: Technology, Protocols and Applications. Wiley-Interscience, 2004.

[578] M. C Mozer, R. H Dodier, M. Anderson, L. Vidmar, R. F Cruickshank, and D. Miller. *The neural network: an overview.* 1995. ID: 53.

[579] M. C. Mozer, R. H. Dodier, M. Anderson, L. Vidmar, R. F. Cruickshank, and D. Miller. *The neural network house: an overview*, pages 371–380. Current trends in connectionism. Erlbaum, 1995.

[580] M Muehlenbrock, O Brdiczka, D Snowdon, and J Meunier. Learning to detect user activity and availability from a variety of sensor data. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*, 2004.

[581] H. Mugge, T. Rho, M. Winandy, M. Won, A. B. Cremers, P. Costanza, and R. Englert. Towards context-sensitive intelligence.

[582] M. Muhlenbrock, O. Brdiczka, D. Snowdon, and J. L. Meunier. Learning to detect user activity and availability from a variety of sensor data. In *PerCom 2004. Proceeding of the Second IEEE Annual Conference on Pervasive Computing and Communications*, pages 13–22, 14-17 March 2004 2004.

[583] M. E. Muller. Can user models be learned at all? inherent problems in machine learning for user modelling. In *Knowledge Engineering Review*, volume 19, pages 61–88. Cambridge University Press, 2004.

[584] Martin Murth, Gerson Joskowicz, Kühn Eva, Dario Cerizza, David de Francisco, Alessandro Ghioni, Reto Krummenacher, Daniel Martin, Lyndon Nixon, Nuria Sanchez, Brahmananda Sapkota, Omair Shafiq, and Daniel Wutke. D6.2 triple space reference architecture. Technical report, TripCom, 2008.

[585] N. Mustapha, M. N. Sulaiman, M. Othman, and M. H. Selamat. Fast discovery of long patterns for association rules. *International Journal of Computer Mathematics*, 80(8):967–976, 2003. URL: `http://dx.doi.org/10.1080/0020716031000112376`.

[586] A. Muñoz, A. Vera, J. A. Botía, and A. F. Gómez Skarmeta. Defining basic behaviours in ambient intelligence environments by means of rule-based programming with visual tools. In *Proceeding of the 1st Workshop on Artificial Intelligence Techniques for Ambient Intelligence*, pages 12–16, 2006.

[587] E. D Mynatt, M. Back, R. Want, M. Baer, and J. B Ellis. Designing audio aura. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 566–573, 1998.

[588] E. D Mynatt, M. Back, R. Want, and R. Frederick. Audio aura: light-weight audio augmented reality. *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 211–212, 1997.

[589] Jin Nakazawa, Hideyuki Tokuda, W. Keith Edwards, and Umakishore Ramachandran. A bridging framework for universal interoperability in pervasive systems. *icdcs*, 00:3, 2006.

[590] Srini Narayanan and Sheila A McIlraith. Simulation, verification and automated composition of web services. pages 77–88. ACM Press.

[591] D Nau, Ilghami O, Kuter U, Murdock JW, Wu D, and Fusun Y. Shop2: An htn planning system. journal of artificial intelligence research. 379-404, 2003.

[592] A. Ndiaye, P. Gebhard, M. Kipp, M. Klesen, M. Schneider, and W. Wahlster. Ambient intelligence in edutainment: Tangible interaction with life-like exhibit guides, 2005.

[593] Robert Neches, Richard Fikes, Tim Finin, Tom Gruber, Ramesh Patil, Ted Senator, and William R Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.

[594] Q. Ni and M. Sloman. An ontology-enabled service oriented architecture for pervasive computing. volume Vol. 2, pages 797–8. IEEE Comput. Soc Dept. of Comput., Imperial Coll., London, UK.

[595] Qun Ni. Service composition in ontology enabled service oriented architecture for pervasive computing.

[596] Shaylor Nik, N. Simon Douglas, and R. Bush William. A java virtual machine architecture for very small devices. *SIGPLAN Not.*, 38(7):34–41, 2003. 780738.

[597] Lyndon J B Nixon, Philipp Obermeier, Sebastian Dill, Janne Saarela, and Pasi Tiitinen. D3.4 distributed semantic query tool for triple space. Technical report, TripCom.

[598] Lyndon J B Nixon, Philipp Obermeier, Omair Sharif, Janne Saarela, and Vassil Momtchev. D3.3 semantic matching in distributed spaces. Technical report, TripCom, 2008.

[599] Lyndon J B Nixon, Kia Teymourian, Reto Krummenacher, Hans Moritsch, Vassil Momtchev, Alessandro Ghioni, and Adi Schütz. D2.4 semantic clustering and Self-Organization in triple space. Technical report, TripCom, 2008.

[600] Lyndon JB Nixon, Daniel Martin, Daniel Wutke, Martin Murth, Reto Krummenacher, Brahmananda Sapkota, Zhangbing Zhou, Hans Moritsch, Christian Schreiber, Omair Shafiq, Germán Toro del Valle, Davide Cerri, and Vassil Momtchev. D6.3 platform API specification for interaction between all components. Technical report, TripCom, 2008.

[601] Brian D Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R Walker. Agile application-aware adaptation for mobility. pages 276–287. ACM Press.

[602] T. Di Noia, E. Di Sciascio, and M. Donini. A Non-Monotonic approach to semantic matchmaking and request refinement in E-Marketplaces. pages 81–96.

[603] H. O. Nyongesa and S. Maleki-dizaji. User modelling using evolutionary interactive reinforcement learning, 2006.

[604] University of Essex. ispace. 2008.

[605] Nicole Oldham, Kunal Verma, Amit P Sheth, and Farshad Hakimpour. Semantic WS-agreement partner selection. pages 697–706. crossref: DBLP:conf/www/2006.

[606] Ontotext. SwiftOWLIM system documentation. Technical report, 2007.

[607] Openstorm. *Service Orchestrator*, volume 2006. 2006.

[608] ORDI. *Ontology Representation and Data Integration*. 2008.

[609] C. Ouyang, W. M. P van der Aalst, S. Breutel, M. Dumas, A. H. M ter Hofstede, H. M. W Verbeek, and B. P. M. Center Report BPM. Formal semantics and analysis of control flow in WS-BPEL. Technical report, BPMcenter.org, 2005.

[610] M. Owen and J. Raj. BPMN and business process management: Introduction to the new business process modeling standard. *Popkin Software*, 2003.

[611] M. I. T Oxygen. *Project Oxygen, Pervasive, Human-centered Computing*.

[612] Pinar Ozturk. Towards a knowledge-level model of context and context use in diagnostic problems. *Applied Intelligence*, 10(2-3):123–137, 1999. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[613] Pinar Ozturk and Agnar Aamodt. Context model for knowledge-intensive case-based reasoning. *International Journal of Human Computer Studies*, 48(3):331–355, 1998. URL: `http://dx.doi.org/10.1006/ijhc.1997.0174`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[614] P-Grid. *The P-Grid Project*, volume 2008.

[615] B. Padmanabhan and A. Tuzhilin. Pattern discovery in temporal databases: A temporal logic approach, 1996.

[616] S. M Pahveli, A. Matomo, and I. Kojima. SPARQL-based OWL-S service matchmaking. pages 35–50.

[617] T. Paine. *Agents and the Semantic Web Activity*, volume 5. 2004.

[618] M. Pantic, A. Pentland, A. Nijholt, and T. Huang. Human computing and machine understanding of human behavior: A survey. In *Proceedings of the 8th international conference on Multimodal interfaces*, pages 239–248. ACM, 2006.

[619] K. Panton, C. Matuszek, D. Lenat, D. Schneider, M. Witbrock, N. Siegel, and B. Shepard. Common sense reasoning- from cyc to intelligent assistant, 2006.

[620] M. Paolucci, A. Ankolekar, N. Srinivasan, and K. Sycara. The DAML-S virtual machine. *2nd International Semantic Web Conference*, 2870:290–305, 2002.

[621] Massimo Paolucci, Takahiro Kawamura, Terry R Payne, and Katia P Sycara. Semantic matching of web services capabilities. pages 333–347. Springer-Verlag.

[622] Michael P Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, and Bernd J Krümer. Service-Oriented computing: A research roadmap. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.

[623] S. Parasuraman, V. Ganapathy, and B. Shirinzadeh. Behavior coordination and selection using situation context-dependent method for behavior based robot navigation using ai techniques for real world environments, 2003.

[624] J. Parra, J. Bilbao, A. Urbieta, and E. Azketa. Standard multimedia protocols for localization in "Seamless handover.ªpplications. Springer Verlag.

[625] T. Partala, V. Surakka, and T. Vanhala. Real-time estimation of emotional experiences from facial expressions. *Interacting with Computers*, 18(2):208–226, 2006.

[626] J. Pascoe. The stick-e note architecture: extending the interface beyond the user. *Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 261–264, 1997.

[627] Jason Pascoe. Adding generic contextual capabilities to wearable computers. pages 92–99. IEEE Computer Society. note: Online proceedings available from http://iswc.gatech.edu/.

[628] Abhijit A Patil, Swapna A Oundhakar, Amit P Sheth, and Kunal Verma. Meteor-s web service annotation framework. pages 553–562. ACM Press.

[629] D. J. Patterson, D. Fox, D. Kautz, K. Fishkin, M. Perkowitz, and M. Philipose. Contextual computer support for human activity. volume 3, pages 71–72, Stanford, CA, United States, Mar 22-24 2004 2004.

[630] J. Peer. Web service composition as AI planning-a survey. Technical report, University of St.Gallen, 2005.

[631] Chris Peltz. Web service orchestration and choreography: a look at WSCI and BPEL4WS. *Web Services Journal*, 2003.

[632] Chris Peltz. Web services orchestration: a review of emerging technologies, tools and standards. *Technical report, HP Technical white paper*, 2003.

[633] Chris Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.

[634] M. A. Perez, L. Susperregi, I. Maurtua, A. Ibarguren, and B. Sierra. Software agents for ambient intelligence based manufacturing. In *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, page 6, Prague, Czech Republic, 15-16 June 2006 2006.

[635] Filip Perich. A service for aggregating and interpreting contextual information. Technical report, 2002.

[636] C. E Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, 1994.

[637] C. E Perkins and E. M Royer. Ad-hoc on-demand distance vector routing. 1999.

[638] D. Petrelli. E. not, c. strapparava, o. stock, and m. zancanaro. modeling context is like taking pictures. *Proceedings of Workshop on Context Awareness (CHI 2000)*, 2000.

[639] C. A Petri. *Kommunikation mit Automaten*. 1962.

[640] J. Petzold, F. Bagci, W. Trumler, and T. Ungerer. Next location prediction within a smart office building, 2005.

[641] J Pineau, M Montemerlo, M Pollack, N Roy, and S Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3-4), 2003.

[642] M. E. Pollack. Intelligent technology for an aging population: The use of ai to assist elders with cognitive impairment. *AI Magazine*, 26(2):9–24, 2005.

[643] A. Polleres, H. Boley, M. Kifer, and October. *RIF Datatypes and Built-Ins 1.0*, volume 2008. 2008.

[644] S. R Ponnekanti and A. Fox. SWORD: a developer toolkit for web service composition. *Proceedings of the Eleventh World Wide Web Conference, Web Engineering Track*, 2002.

[645] Hossein Pourreza and Peter Graham. On the fly service composition for local interaction environments. page 393. IEEE Computer Society.

[646] C. Preist. A conceptual architecture for semantic web services. pages 395–409. Springer.

[647] Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. pages 148–159. crossref: DBLP:conf/eusai/2004.

[648] G. Privat and T. Flury. Position-Based interaction for indoor ambient intelligence environments. volume 2875, pages 192–207. Springer.

[649] Blitz Project. *The Blitz Project*. 2008.

[650] I. S. T. Amigo Project and Ist. Amigo d3.2 amigo middleware core: Prototype implementation and documentation. Technical report, 2006.

[651] Meteor-S Project. *METEOR-S Semantic Web Services and Processes*, volume 2006. 2006.

[652] Plastic Project and Plastic Deliverable D. Middleware specification and architecture. Technical report, 2007.

[653] Sirena Project. volume 2008. 2008.

[654] Soda Project. volume 2008. 2008.

[655] Y. Punie. A social and technological view of ambient intelligence in everyday life: What bends the trend. *Key deliverable, The European Media and Technology in Everyday Life Network (EMTEL)*, 2003.

[656] A. Qasem, J. Heflin, and H. Muñoz-Avila. Efficient source discovery and service composition for ubiquitous computing environments.

[657] Lirong Qiu, Zhongzhi Shi, and Fen Lin. Context optimization of AI planning for services composition. pages 610–617. IEEE Computer Society.

[658] M. S Raisinghani, A. Benoit, J. Ding, M. Gomez, K. Gupta, V. Gusila, D. Power, and O. Schmedding. Ambient intelligence: Changing forms of Human-Computer interaction and their social implications. *Journal of Digital Information*, 5(4), 2004.

[659] Andry Rakotonirainy, Jadwiga Indulska, and Karen Henricksen. Generating context management infrastructure from High-Level context models. pages 1–6. annote: .ᴬndry Rakotonirainy (School of Information Technology and Electrical Engineering, The University of Queensland and Distributed Systems Technology Centre, karen, jaga andryitee.uq. edu.au); Jadwiga Indulska (School of Information Technology and Electrical Engineering, The University of Queensland and Distributed Systems Technology Centre, karen, jaga andryitee.uq. edu.au); Karen Henricksen (School of Information Technology and Electrical Engineering, The University of Queensland and Distributed Systems Technology Centre, karen, jaga andryitee.uq. edu.au);".

[660] B. Raman and R. H Katz. An architecture for highly available wide-area service composition. *Computer Communications*, 26(15):1727–1740, 2003.

[661] Jorge C. G. Ramirez, Diane J. Cook, Lynn L. Peterson, and Dolores M. Peterson. Temporal pattern discovery in course-of-disease data. *IEEE Engineering in Medicine and Biology*, 19(4):63–71, 2000. URL: `http://dx.doi.org/10.1109/51.853483`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[662] Carlos Ramos, Juan Augusto, , and Daniel Shapiro. Ambient intelligence - the next step for artificial intelligence (guest editorsíntroduction to the special issue on ambient intelligence). *IEEE Intelligent Systems*, 23(2):15–18, Mar/Apr 2008.

[663] A. Ranganathan, J. Al-Muhtadi, and R. H Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, pages 62–70, 2004.

[664] A. Ranganathan and S. McFaddin. Using workflows to coordinate web services in pervasive computing environments. *Proceedings of the IEEE International Conference on Web Services*, pages 288–295, 2004.

[665] Anand Ranganathan and Roy H Campbell. A middleware for Context-Aware agents in ubiquitous computing environments. volume 2672, pages 143–161. crossref: DBLP:conf/middleware/2003.

[666] Anand Ranganathan and Roy H Campbell. Autonomic pervasive computing based on planning. *Proceedings International Conference on Autonomic Computing*, pages 80–87, 2004.

[667] J. Rao and X. Su. A survey of automated web service composition methods. *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition*, 2004.

[668] S. P. Rao and D. J. Cook. Predicting inhabitant action using action and task models with application to smart homes. *International Journal on Artificial Intelligence Tools (Architectures, Languages, Algorithms)*, 13(1):81–99, 2004.

[669] P. G Raverdy, V. Issarny, R. Chibout, and Agnes de La Chapelle. A Multi-Protocol approach to service discovery and access in pervasive environments.

[670] Pierre-Guillaume Raverdy, Oriana Riva, Agnes de La Chapelle, Rafik Chibout, and Valerie Issarny. Efficient context-aware service discovery in Multi-Protocol pervasive environments. page 3. IEEE Computer Society.

[671] R. Razavi, J. F. Perrot, and N. Guelfi. Adaptive modeling: An approach and a method for implementing adaptive agents, 2005.

[672] W. Reisig and G. Rozenberg. Lectures on petri nets II: applications. *Lecture notes in computer science*, 1492, 1998.

[673] W. Reisig, G. Rozenberg, and G. Roxenberg. Lectures on petri nets i: Basic models. *Lecture notes in computer science*, 1491, 1998.

[674] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[675] P. Remagnino and G. L. Foresti. Ambient intelligence: A new multidisciplinary paradigm. In *IEEE Transactions on systems, man and cybernetics Part A: Systems and Humans*, volume 35, pages 1–6. IEEE Systems, Man, and Cybernetics Society, 2005.

[676] P. Remagnino and G. L Foresti. *Ambient Intelligence: A New Multidisciplinary Paradigm*, volume 35. 2005. ID: 21.

[677] E. Rich. User modelling via stereotypes, 1979.

[678] Alistair Riddoch, Nick Gibbins, and Steve Harris. *Sitio web del proyecto 3store*. 2008.

[679] J. Riemer, F. Martin-Recuerda, Y. Ding, M. Murth, B. Sapkota, R. Krummenacher, O. Shafiq, D. Fensel, and E. K\ühn. Triple space computing: Adding semantics to space-based computing. *The Semantic Web–ASWC 2006*, page 300–306, 2006.

[680] Johannes Riemer. Implementation of CORSO extension (Intranet, internet, Multi-Port, notification, firewall, security). Technical report, Vienna University of Technology, 2007.

[681] Marcia Riley. *Ubiquitous Computing: An Interesting New Paradigm*, volume 2006. 1997.

[682] E. G Riva, F. Vatalaro, F. Davide, and M. Alcaniz. *Ambient Intelligence*. IOS Press, 2005.

[683] G. Riva, P. Loreti, M. Lunghi, F. Vatalaro, and F. Davide. Presence 2010: The emergence of ambient intelligence. *Being There: Concepts, effects and measurement of user presence in synthetic environments*, pages 59–82, 2003.

[684] F. Rivera-Illingworth, V. Callaghan, and H. Hagras. A neural network agent based approach to activity detection in ami environments. In *IEEE International Workshop on Intelligent Environments*, pages 92–99, 2005.

[685] D. Rocchesso and R. Bresin. *Emerging Sounds for Dissapearing Computers*, pages 233–255. The Dissapearing Computer. Springer-Verlag, 2007.

[686] D. Roman. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[687] D. Roman, H. Lausen, and U. Keller. Web service modeling ontology standard (WSMO-standard). *Working Draft D2v0*, 2, 2004.

[688] D. Roman, L. Vasiliu, and C. Bussler. Orchestration in WSMO. *WSMO working draft*, 2004.

[689] Manuel Roman and Roy H Campbell. Gaia: enabling active spaces. pages 229–234. ACM Press.

[690] Manuel Roman, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H Campbell, and Klara Nahrstedt. Gaia: a middleware platform for active spaces. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):65–67, 2002.

[691] Manuel Roman, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H Campbell, and Klara Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 01(4):74–83, 2002.

[692] Sydney Rosario, Albert Benveniste, Stefan Haar, Claude Jard, and Report Technical. Net systems semantics of web services orchestrations modeled in orc. Technical report, Istitut de REcherche en Informatique et Systèmes Alèatories, 2006.

[693] David De Roure, Nicholas R Jennings, and Nigel R Shadbolt. The semantic grid: Past, present, and future. *Proceedings of the IEEE*, 93(3):669–680, 2005. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[694] W. Roy and H. Andy. Active badges and personal interactive computing objects. *IEEE Transactions of Consumer Electronics*, pages 91–102, 1992.

[695] M. Rudary, S. Singh, and M. E. Pollack. Adaptive cognitive orthotics: Combining reinforcement learning and constraint-based temporal reasoning. In *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, pages 719–726, Banff, Alta, Canada, Jul 4-8 2004 2004.

[696] W. A Ruh, F. X Maginnis, and W. J Brown. *Enterprise Application Integration: A Wiley Tech Brief*. John Wiley, 2001.

[697] S.J. Russell and P. Norvig. *Artificial Intelligence: A modern approach, 2nd edition*. Prentice Hall, 2003.

[698] U. Rutishauser, J. Joller, and R. Douglas. *Control and Learning of Ambience by an intelligent building.* 2004. ID: 22.

[699] U. Rutishauser, J. Joller, and R. Douglas. Control and learning of ambience by an intelligent building. In *IEEE on Systems, man and cybernetics: a special issue on ambient intelligence*, pages 121–132. IEEE Systems, Man, and Cybernetics Society, 2005.

[700] U. Rutishauser, J. Trindler, and R. Douglas. Unsupervised learning and control provide ambient intelligence to smart buildings, 2004.

[701] N. Ryan. ConteXtML: exchanging contextual information between a mobile client and the FieldNote server. 1999.

[702] N. S Ryan, J. Pascoe, and D. R Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. Tempus Reparatum.

[703] Tomasz Rybicki. *MobileSpaces - JavaSpaces for mobile devices.* PhD thesis, 2004.

[704] N. Sadeh, E. Chan, and L. Van. MyCampus: an Agent-Based environment for Context-Aware mobile services.

[705] N. M. Sadeh, F. L. Gandom, and O. B. Kwon. Ambient intelligence: The mycampus experience. Technical Report CMU-ISRI-05-123, ISRI, 2005.

[706] N. M Sadeh, F. Gandon, O. B Kwon, and Isri Cmu. Ambient intelligence: The MyCampus experience. Technical report, Carnegie Mellon University Technical ReportCMU-ISRI-05-123. June, 2005.

[707] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *IEEE Computer*, 36:25–31, 2003.

[708] G. Salaun, L. Bordeaux, and M. Schaerf. Describing and reasoning on web services using process algebra. *Proceedings of the IEEE International Conference on Web Services*, 00, 2004.

[709] E. P. Salonen, M. Turunen, J. Hakulinen, L. Helin, P. Prusi, and A. Kainulainen. Distributed dialogue management for smart terminal devices. In *Interspeech 2005*, pages 849–852, 2005.

[710] Michael Samulowitz, Florian Michahelles, and Claudia Linnhoff-Popien. CAPEUS: an architecture for Context-Aware selection and execution of services. pages 23–40. Kluwer, B.V.

[711] M. Sanchez-Marre, U. Cortes, M. Martinez, J. Comas, and I. Rodriguez-Roda. An approach for temporal case-based reasoning: Episode-based reasoning. In *6th International Conference on Case-Based Reasoning, ICCBR 2005*, volume 3620, pages 465–476, Chicago, IL, United States, Aug 23-26 2005 2005.

[712] J. M. Sanders. Sensing the subtleties of everyday life, 2000.

[713] L.A. SanMartin, V.M. Pelaez, R. Gonzalez, and A.M. Campos. Environmental user preference learning for smart homes. In *Proceedings of the 5th International Conference on Intelligent Environments*, pages 177–184, 2009.

[714] Brahmananda Sapkota, Doug Foxvog, Daniel Wutke, Daniel Martin, Martin Murth, Omair Shafiq, Andrea Turati, Emanuele Della Valle, Nuria Sanchez, and Jacek Kopecky. D4.1 architectural integration of triple spaces with web service infrastructures. Technical report, TripCom, 2007.

[715] Brahmananda Sapkota, Edward Kilgarriff, and Christoph Bussler. Role of triple space computing in semantic web services. In *Frontiers of WWW Research and Development - APWeb 2006*, pages 714–719. 2006. 10.1007/11610113_63.

[716] Brahmananda Sapkota, Vassil Momtchev, and Omair Shafiq. D1.3 High-Performance storage implementation. Technical report, TripCom, 2008.

[717] M. Satyanarayanan. Pervasive computing: Vision and challenges, 2001.

[718] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4):10–17, 2001. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[719] M. Satyanarayanan. A catalyst for mobile and ubiquitous computing. *IEEE Pervasive Computing*, 1(1):2–5, 2002.

[720] M. Satyanarayanan. *Challenges in Implementing a Context-Aware System*, volume 3. 2002.

[721] Bill Schilit. *A Context-Aware System Architecture for Mobile Distributed Computing*. 1995.

[722] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. pages 85–90. IEEE, Los Alamitos, CA, USA Columbia Univ, New York, NY, USA. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[723] Bill N Schilit, Norman Adams, Rich Gold, M. M Tso, and Roy Want. The PARCTAB mobile computing system. *Workstation Operating Systems, 1993.Proceedings., Fourth Workshop on*, pages 34–39, 1993.

[724] Bill N Schilit and Marvin M Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[725] C. Schlenoff. *The Process Specification Language (PSL) Overview and Version 1.0 Specification*. US Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 2000.

[726] A. Schmidt. Interactive Context-Aware systems interacting with ambient intelligence. 2005.

[727] A. Schmidt and K. Van Laerhoven. How to build smart appliances? *IEEE Personal Communications*, 8(4):66–71, 2001. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[728] Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, Kristof Van Laerhoven, and Walter Van de Velde. Advanced interaction in context. pages 89–101. Springer-Verlag.

[729] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers and Graphics (Pergamon)*, 23(6):893–901, 1999. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[730] Michael Schneider, Alexander Kröner, Julio C. Encinas Alvarado, Andres García Higuera, Juan C. Augusto, Diane J. Cook, Veikko Ikonen, Pavel Cech, Peter Mikulecký, Achilles Kameas, and Vic Callaghan. 1st international workshop on rfid technology: Concepts, practices and solutions. included in workshops proceedings of the 5th international conference on intelligent environments, 2009.

[731] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. *Proc.of the Int.Conference on Advanced Information Systems Engineering (CAiSE)*, 2000.

[732] J. Scicluna, A. Polleres, and D. Roman. Ontology-based choreography and orchestration of WSMO. *WSMO Working Draft*, 2005.

[733] L. Serafini and M. Homola. *Modular Knowledge Representation and Reasoning in the Semantic Web. Semantic Web Information Management.* 2010.

[734] L. Serafini, A. Tamilin, and A. Drago. *Distributed reasoning architecture for the semantic web. The Semantic Web: Research and Applications.*

[735] N. Shadbolt. Ambient intelligence, 2003.

[736] N. Shadbolt. Ambient intelligence. *IEEE Intelligent Systems*, 18(4):2–3, 2003.

[737] R. Shapiro. A comparison of XPDL, BPML, and BPEL4WS. *Draft document*, 2002.

[738] H. Sharp, Y. Rogers, and J. Preece. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley and Sons Ltd., 2007.

[739] M. Sheshagiri, M. desJardins, and T. Finin. A planner for composing services described in DAML-S. *Workshop on Planning for Web Services, International Conference on Automated Planning and Scheduling*, 2003.

[740] M. Sheshagiri, N. M Sadeh, and F. Gandon. Using semantic web services for Context-Aware mobile applications. *MobiSys - Workshop on Context Awareness*, 2004.

[741] A. Sheth. Semantic web process lifecycle: Role of semantics in annotation, discovery, composition and orchestration. *Invited Talk, WWW 2003 Workshop on E-Services and the Semantic Web*, 20, 2003.

[742] S. C. K. Shiu, D. S. Yeung, C. H. Sun, and X. Z. Wang. Transferring case knowledge to adaptation knowledge: An approach for case-base maintenance. *Computational Intelligence*, 17(2):295–314, 2001. URL: `http://dx.doi.org/10.1111/0824-7935.00146`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[743] N. A. Sigrimis, K. G. Arvanitis, and R. S. Gates. Learning technique for a general purpose optimizer. *Computers and Geotechnics*, 26(2):83–103, 2000. URL: `http://dx.doi.org/10.1016/S0266-352X(99)00046-4`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[744] N. A. Sigrimis, K. G. Arvanitis, and R. S. Gates. Learning technique for a general purpose optimizer. *Computers and Geotechnics*, 26(2):83–103, 2000. URL: `http://dx.doi.org/10.1016/S0266-352X(99)00046-4`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[745] R. Simpson, D. Schreckenghost, E. F. LoPresti, and N. Kirsch. *Plans and planning in smart homes*, pages 71–84. Designing Smart Homes. The Role of Artificial Intelligence. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[746] Richard Simpson, Debra Schreckenghost, Edmund F LoPresti, and Ned Kirsch. Plans and planning in smart homes. In Juan Augusto and Chris Nugent, editors, *Designing Smart Homes: The role of Artificial Intelligence*. Springer Verlag, 2006.

[747] Abhishek Singh and Michael Conway. Survey of context aware frameworks analysis and criticism. Technical report, 2006.

[748] M. P Singh and M. N Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley and Sons, 2005.

[749] E. Sirin and B. Parsia. Planning for semantic web services. *Semantic Web Services Workshop at 3rd International Semantic Web Conference*, 2004.

[750] E. Sirin, B. Parsia, and J. Hendler. Template-based composition of semantic web services.

[751] K. Sivashanmugam. The METEOR-S framework for semantic web process composition. *Master's Thesis*, 2003.

[752] K. Sivashanmugam, J. A Miller, A. P Sheth, and K. Verma. Framework for semantic web process composition. *International Journal of Electronic Commerce*, 9(2):71–106, 2004.

[753] Joan M Smith. Standard generalized markup language and related standards. *Computer Communications*, 12(2):80–84, 1989. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[754] Michael K Smith, Chris Welty, and Deborah L McGuinness. *OWL Web Ontology Language Guide*. 2004.

[755] L. Snidaro, C. Micheloni, and C. Chiavedale. Video security for ambient intelligence, 2005.

[756] Zhexuan Song, Yannis Labrou, and Ryusuke Masuoka. Dynamic service discovery and management in task computing. *Mobiquitous*, 00:310–318, 2004.

[757] R. Srikant. Fast algorithms for mining association rules, 1994.

[758] R. Srikant and R. Agrawal. Mining sequential patterns: generalizations and performance improvements. In *Proceedings of 5th Conference on Extended Database Technology (EDBT'96)*, pages 3–17, Avignon, France, 25-29 March 1996 1996. IBM Almaden Res. Center, San Jose, CA, USA, Springer-Verlag. M1: Copyright 1996, IEE; T3: Advances in Database Technology - EDBT '96. 5th International Conference on Extending Database Technology. Proceedings.

[759] A. Srinivasan, D. Bhatia, and S. Chakravarthy. Discovery of interesting episodes in sequence data. In *21st Annual ACM Symposium on Applied Computing*, volume 1, pages 598–602, Dijon, France, 23-27 April 2006 2006. ACM.

[760] B. Srivastava and J. Koehler. Web service Composition-Current solutions and open problems. *ICAPS 2003 Workshop on Planning for Web Services*, pages 28–35, 2003.

[761] B. Srivastava and J. Koehler. Planning with Workflows-An emerging paradigm for web service composition. *Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[762] Frank Stajano. Security for ubiquitous computing. volume 3506, page 2. Springer Verlag, Heidelberg, D-69121, Germany Computer Laboratory, University of Cambridge, Cambridge, United Kingdom. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[763] V Stanford. Biosignals offer potential for direct interfaces and health monitoring. *IEEE Pervasive Computing*, 3:99–103, 2004.

[764] V. Stankovski and J. Trnkoczy. *Application of decision trees to smart homes*, pages 132–45. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[765] D. Stokic, U. Kirchhoff, and H. Sundmaeker. Ambient intelligence in manufacturing industry: control system point of view. In *Proceedings of the Eighth IASTED International Conference on Control and Applications*, pages 63–8, Montreal, Que., Canada, 24-26 May 2006 2006.

[766] M. Stollberg and S. Arroyo. WSMO tutorial. *WSMO Deliverable*, 2005.

[767] M. Stollberg, U. Keller, H. Lausen, and S. Heymans. Two-Phase web service discovery based on rich functional descriptions. volume 4519, pages 99–113. Springer.

[768] Peter Stone and Manuela Veloso. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000. URL: `http://dx.doi.org/10.1023/A:1008942012299`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

[769] Thomas Strang. *Service Interoperability in Ubiquitous Computing Environments*. 2003.

[770] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey.

[771] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. CoOL: a context ontology language to enable contextual interoperability. volume 2893, pages 236–247.

[772] Maria A Strimpakou, Ioanna G Roussaki, and Miltiades E Anagnostou. A context ontology for pervasive service provision. pages 775–779. IEEE Computer Society.

[773] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[774] N. C. Narendra Sattanathan Subramanian and Zakaria Maamar. ConWeSc: context based semantic web services composition.

[775] L. Susperregi, I. Maurtua, C. Tubío, M. A. Pérez, I. Segovia, and B. Sierra. Una arquitectura multiagente para un laboratirio de inteligencia ambiental en fabricación.

[776] L. Susperregi, I. Maurtua, C. Tubío, M. A Pérez, I. Segovia, and B. Sierra. *Una arquitectura multiagente para un Laboratirio de inteligencia ambiental en Fabricación*. 2004. ID: 26.

[777] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *ACM SIGMOD Record*, 28(1):47–53, 1999.

[778] K. Sycara, J. Lu, M. Klusch, and S. Widoff. Matchmaking among heterogeneous agents on the internet.

[779] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1):27–46, 2003.

[780] Sakari Tamminen, Antti Oulasvirta, Kalle Toiskallio, and Anu Kankainen. Understanding mobile contexts. *Personal Ubiquitous Comput.*, 8(2):135–143, 2004.

[781] T. Tamura. A smart house for emergencies in the elderly, 2006.

[782] Joo Geok Tan, Daqing Zhang, Xiaohang Wang, and Heng Seng Cheng. Enhancing semantic spaces with Event-Driven context interpretation. volume 3468, pages 80–97. crossref: DBLP:conf/pervasive/2005.

[783] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Proceedings of Pervasive*, pages 158–175, 2004.

[784] Sasu Tarkoma, Ramya Balu, Jaakko Kangasharju, Miika Komu, Mika Kousa, Tancred Lindholm, Mikko Mäkelä, Marko Saaresto, Kristian Slavov, and Kimmo Raatikainen. State of the art in enablers for applications in future mobile wireless internet. Technical report, Helsinki Institute for Information Technology, 2004. type: HIIT Publication; key: HIIT.

[785] M. H ter Beek, C. A Ellis, J. Kleijn, and G. Rozenberg. Synchronizations in team automata for groupware systems. *Computer Supported Cooperative Work*, 12(1):21–69, 2003.

[786] Maurice ter Beek, Antonio Bucchiarone, and Stefania Gnesi. Web service composition approaches: From industrial standards to formal methods. *Second International Conference on Internet and Web Applications and Services*, page 15, 2007.

[787] Maurice H ter Beek, Antonio Bucchiarone, and Stefania Gnesi. A survey on service composition approaches: From industrial standards to formal methods. *ISTI-CNR Technical report*, 2006.

[788] V. Terziyan and O. Kononenko. Semantic web enabled web services: State-of-Art and industrial ChAllenges. *Int. Conference on Web Services Europe*, pages 183–197, 2003.

[789] V. T Thai. *A Survey on Ambient Intelligence in Manufacturing Environment*. 2005.

[790] C. K Tham and R. Buyya. SensorGrid: integrating sensor networks and grid computing. *CSI Communications*, 29:24–29, 2005.

[791] S. Thatte. XLANG: web services for business process design. *Microsoft Corporation*, 2001.

[792] D. Tidwell. Web services-the web's next revolution. *IBM developerWorks*, 2001.

[793] I. Toma, B. Sapkota, J. Scicluna, J. M Gomez, D. Roman, and D. Fensel. A P2P discovery mechanism for web service execution environment. In *Second WSMO Implementation Workshop*, 2005.

[794] David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services.

[795] Paolo Traverso and Marco Pistore. Automated composition of semantic web services into executable processes. pages 380–394. crossref: DBLP:conf/semweb/2004.

[796] Georgios Trimponias, Chan Le Duc, Antoine Zimmermann, and Simon Schenk. Reasoning over distributed networked ontologies and data sources.

[797] Tripcom. *Triplespace Computing: Large-Scale Integrated Knowledge Applications*. 2008.

[798] B. A Truong, Y. K Lee, and S. Y Lee. Modeling uncertainty in context-aware computing. pages 676–681.

[799] Aphrodite Tsalgatidou and Thomi Pilioura. An overview of standards and related technology in web services. *Distributed and Parallel Databases*, 12(2-3):135–162, 2002.

[800] M. Turunen, J. Hakulinen, A. Kainulainen, A. Melto, and T. Hurtig. Design of a rich multimodal interface for mobile spoken route guidance. In *Proceedings of Interspeech 2007 - Eurospeech*, pages 2193–2196. 2007.

[801] Luxio Ugarte. *¿Sinfonía o Jazz?: Koldo Saratxaga y el modelo Irizar: un modelo basado en las personas*. Granica, 2004.

[802] A. Urbieta. Service composition and matchmaking of embedded semantic-devices in ubiquitous computing environments.

[803] A. Urbieta. Modelado de contexto basado en ontologías para composición dinámica de servicios web semánticos en plataformas embebidas aplicadas a entornos de inteligencia ambiental. Technical report, 2006.

[804] A. Urbieta. Etxetresnek gustukoago dute jazza jotzea sinfonia jotzea baino. *Elhuyar*, 241:108–113, 2008.

[805] A. Urbieta, E. Azketa, I. Gomez, J. Parra, and N. Arana. Analysis of effects- and preconditions-based service representation in ubiquitous computing environments.

[806] A. Urbieta, E. Azketa, I. Gomez, J. Parra, and N. Arana. Bridging the gap between services and context in ubiquitous computing environments using an effects- and conditions-based model. Springer Verlag.

[807] A. Urbieta, E. Azketa, I. Gomez, J. Parra, and N. Arana. Towards effects-based service description and integration in pervasive environments.

[808] A. Urbieta, G. Barrutieta, J. Parra, and A. Uribarren. Estado del arte de composición dinámica de servicios en entornos de computación ubicua.

[809] A. Urbieta, G. Barrutieta, J. Parra, and A. Uribarren. A survey of dynamic service composition approaches for ambient systems. ACM, ICST.

[810] A. Urbieta, C. Gómez, G. Barrutieta, M. Gonzalez, J. Abaitua, J. Díaz, and C. Lamsfus. SEMTEK: an advanced research project about agents'theory applied to the context semantic adaptation.

[811] Mike Uschold and Michael Gruninger. Ontologies: principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–136, 1996. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[812] A. Vainio, M. Vaaltonen, and J. Vanala. Proactive fuzzy control and adaptation methods for smart homes. *IEEE Intelligent Systems*, 23(2):42–49, Mar/Apr 2008.

[813] A. M. Vainio, M. Valtonen, and J. Vanhala. Proactive fuzzy control and adaptation methods for smart homes. *IEEE Intelligent Systems*, 23(2):42–49, 2008.

[814] Mathieu Vallee, Fano Ramparany, and Laurent Vercouter. Dynamic service composition in ambient intelligence environments: a multi-agent approach.

[815] Mathieu Vallee, Fano Ramparany, and Laurent Vercouter. Flexible composition of smart device services. pages 165–171. CSREA Press. crossref: DBLP:conf/csreaPSC/2005.

[816] Mathieu Vallee, Fano Ramparany, and Laurent Vercouter. A Multi-Agent system for dynamic service composition in ambient intelligence environments. pages 165–171.

[817] W. M. P van der Aalst. Pi calculus versus petri nets: Let us eat "humble pierather than further inflate the "Pi hype". *Unpublished paper*, 2004.

[818] W. M. P van der Aalst, M. Dumas, A. H. M ter Hofstede, and P. Wohed. Pattern-Based analysis of BPML (and WSCI). *QUT Technical Report FIT-TR-2002-05*, 2002.

[819] Wil M. P van der Aalst, Marlon Dumas, and Arthur H. M ter Hofstede. Web service composition languages: Old wine in new bottles? *Euromicro Conference, 2003. Proceedings. 29th*, 00:298, 2003.

[820] Wil M. P van der Aalst, Arthur H. M ter Hofstede, Bartek Kiepuszewski, and Alistair P Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[821] W.M.P. van der Aalst, A.J.M.M. Weitjers, and L. Maruster. Workflow mining discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1128–1142, 2004.

[822] M. van Doorn and A. P. de Vries. *Co-Creation in Ambient Narratives*, pages 103–129. Ambient Intelligence for Every life, Lectures Notes in Computer Science 3964. Springer Berlin / Heidelberg, 2006.

[823] M. van Doorn and A. P de Vries. *Co-Creation in Ambient Narratives*. 2006. ID: 32.

[824] T.L.M. van Kasteren and B.J.A. Kröse. Bayesian activity recognition in residence for elders. In *Proceedings of the 3rd International Intelligent Environments Conference*, pages 209–212, 2007.

[825] E. J van Loenen. Ambient intelligence: Philips'vision. *Presentation at ITEA*, 2003.

[826] Juan Ignacio Vazquez. *A Reactive Behavioural Model for Context-Aware Semantic Devices*. PhD thesis, 2007.

[827] Juan Ignacio Vazquez, Diego Lopez de Ipina, and Iñigo Sedano. SOAM: an environment adaptation model for the pervasive semantic web. *Lecture notes in computer science ICCSA 2006*, 3983:108–117, 2006.

[828] Juan Ignacio Vazquez, Diego Lopez de Ipina, and Iñigo Sedano. SoaM: a web-powered architecture for designing and deploying pervasive semantic devices. *International Journal of Web Information Systems*, 2007.

[829] K. Verma, R. Akkiraju, and R. Goodwin. Semantic matching of web service policies. *SDWP Workshop*, 2005.

[830] Kunal Verma. *Configuration and adaptation of semantic web processes*. 2006.

[831] Kunal Verma, Kaarthik Sivashanmugam, Amit Sheth, Abhijit Patil, Swapna Oundhakar, and John Miller. METEOR-S WSDI: a scalable infrastructure of registries for semantic publication and discovery of web services. *Journal of Information Technology and Management, Special Issue on Universal Global Integration*, 6(1):17–39, 2005.

[832] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J. A Stankovic. An assisted living oriented information system based on a residential wireless sensor network. *Proceedings of the Distributed Diagnosis and Home Healthcare Conference*, 2006.

[833] M. Vukovic and P. Robinson. Adaptive, planning based, web service composition for context awareness. *Proceedings of the Second International Conference on Pervasive Computing*, 2004.

[834] W3C. *Composite Capabilities / Preferences Profile*, volume 2006. 2006.

[835] Monica Wachowicz. How can knowledge discovery methods uncover spatio-temporal patterns in environmental data? In *Data Mining and Knowledge Discovery: Theory, Tools, and Technology II*, volume 4057, pages 221–229, Orlando, FL, USA, Apr 24-Apr 25 2000 2000. Wageningen UR, Neth, Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, USA. URL: `http://dx.doi.org/10.1117/12.381736`. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved; T3: Proceedings of SPIE - The International Society for Optical Engineering.

[836] R. Waldinger. Web agents cooperating deductively. *Proceedings of Formal Approaches to Agent-Based Systems 2000*, 1871:250–262, 2000.

[837] L. X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples, 1992.

[838] L.X. Wang. The mw method completed: A flexible system approach to data mining. *IEEE Transactions fuzzy systems*, 11:768–782, 2003.

[839] L.X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Systems, man and cybernetics*, 22:1414–1427, 1992.

[840] R. C Wang, Y. C Chang, and R. S Chang. Design issues of semantic service discovery for ubiquitous computing. pages 880–885.

[841] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using OWL. pages 18–22. crossref: DBLP:conf/percom/2004w.

[842] Xiaohang Wang, Jin Song Dong, ChungYau Chin, SankaRavipriya Hettiarachchi, and Daqing Zhang. Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39, 2004.

[843] R. Want, B. N Schilit, N. I Adams, R. Gold, K. Petersen, D. Goldberg, J. R Ellis, and M. Weiser. An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–43, 1995.

[844] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Trans.Inf.Syst.*, 10(1):91–102, 1992.

[845] Roy Want and Trevor Pering. System challenges for ubiquitous and pervasive computing. pages 9–14. Association for Computing Machinery, New York, NY 10036-5701, United States Intel Research, Santa Clara, CA 95052, United States. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[846] Wapforum. *User Agent Profile (UAProf)*, volume 2006. 2006.

[847] C.J.C.H. Watkins and P. Dayan. Q learning. *Machine Learning*, 8:279–292, 1992.

[848] W. Weber. Ambient intelligence: Industrial research on a visionary concept, 2003.

[849] A.J.M.M. Weijters and W.M.P. van der Aalst. Process mining discovering workflow models from event-based data. In *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pages 283–290, 2001.

[850] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

[851] M. Weiser. Hot topics: Ubiquitous computing. *IEEE Computer*, 26:71–72, 1993.

[852] Mark Weiser. *The Computer for the 21st century*, volume 256. 1991.

[853] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, 1993.

[854] L. Wen, W.M.P. van der Aalst, J.Wang, and J. Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, 2007.

[855] S. A White. Business process modeling notation (BPMN) version 1.0. *Business Process Management Initiative, BPMI.org, May*, 2004.

[856] Wikipedia and February. *List of ad-hoc routing protocols*, volume 2007. 2007.

[857] B. C. Williams and P. P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of National Conference on Artificial Intelligence*, volume 2, pages 971–8, Portland, OR, USA, 4-8 Aug. 1996 1996. NASA Ames Res. Center, Moffett Field, CA, USA, MIT Press. M1: Copyright 1997, IEE; T3: Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference.

[858] Terry Winograd. Architectures for context. *Human-Computer Interaction*, 16(2/4):401–419, 2001.

[859] Nirmalie Wiratunga, Susan Craw, Bruce Taylor, and Genevieve Davis. Case-based reasoning for matching smarthouse technology to people's needs. *Knowledge-Based Systems*, 17(2):139–146, 2004. URL: `http://dx.doi.org/10.1016/j.knosys.2004.03.009`.

[860] P. Wisner and D. N Kalofonos. A framework for End-User programming of smart homes using mobile devices.

[861] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed.* Elsevier, 2005.

[862] P. Wohed, W. M. P van der Aalst, M. Dumas, and A. H. M ter Hofstede. Pattern-Based analysis of BPEL4WS. *Department of Computer and Systems Sciences, Stockholm University/The Royal Institute of Technology, Sweden*, 2003.

[863] Petia Wohed, Wil M. P van der Aalst, Marlon Dumas, and Arthur H. M ter Hofstede. Analysis of web services composition languages: The case of BPEL4WS. pages 200–215. crossref: DBLP:conf/er/2003.

[864] M. Wojciechowski. Discovering and processing sequential patterns in databases, 2000.

[865] W. H Wolf and W. Wolf. *Computers as Components: Principles of Embedded Computing Systems Design.* Morgan Kaufmann, 2001.

[866] R F Wolffenbuttel, K M Mahmoud, and P L Regtien. Compliant capacitive wrist sensor for use in industrial robots. *IEEE Transactions on Instrumentation and Measurements*, 39:991–997, 1990.

[867] W. S. Wong. Analysis of temporal and behavioral rules in smart home technologies working report 3, July 2003.

[868] M. Wooldridge and N. R Jennings. The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4):563–592, 1999.

[869] C. Wu and H. Aghajan. Using context with statistical relational models - object recognition from observing user activity in home environment. In *Workshop on Use of Context in Vision Processing (UCVP), ICMI-MLMI*, 2009.

[870] D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia. Automatic web services composition using SHOP2. *Twelfth World Wide Web Conference*, 2003.

[871] Huadong Wu, Mel Siegel, and Sevim Ablay. Sensor fusion for context understanding. volume 1, pages 13–17. Institute of Electrical and Electronics Engineers Inc Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, United States. Compilation and indexing terms, Copyright 2006 Elsevier Inc. All rights reserved.

[872] Chang Xu and S. C Cheung. Inconsistency detection and resolution for context-aware middleware support. pages 336–345. ACM Press.

[873] Kun Yang and Alex Galis. Policy-Driven mobile agents for Context-Aware service in next generation networks. pages 111–120. crossref: DBLP:conf/mata/2003.

[874] Stephen S Yau, H. Davulcu, S. Mukhopadhyay, H. Gong, D. Huang, P. Singh, and F. Gelgi. Automated situation-aware service composition in Service-Oriented computing. *International Journal on Web Services Research*, 4(4):59–82, 2007.

[875] Stephen S Yau and Junwei Liu. Functionality-Based service matchmaking for Service-Oriented architecture. pages 147–154. IEEE Computer Society.

[876] Stephen S Yau and Junwei Liu. Hierarchical situation modeling and reasoning for pervasive computing. pages 5–10. IEEE Computer Society.

[877] Stephen S Yau and Junwei Liu. Incorporating situation awareness in service specifications. pages 287–294. IEEE Computer Society.

[878] J. Ye, L. Coyle, and S. Dobson. Resolving uncertainty in context integration and abstraction: context integration and abstraction. pages 131–140. ACM New York, NY, USA.

[879] X. Yi and K. J Kochut. A CP-nets-based design and verification framework for web services composition. *Web Services, 2004.Proceedings.IEEE International Conference on*, pages 756–760, 2004.

[880] S. R. Young, R. C. Williges, T. L. Smith-Jackson, and Jiyoung Kwahk. Comparison of design requirements methods for monitoring eating patterns of seniors in a smart house. In *Human Factors and Ergonomics Society Conference*, pages 844–8, Baltimore, MD, USA, 30 Sept.-4 Oct. 2002 2002. Dept. of Ind. & Syst. Eng., Virginia Polytech. Inst. & State Univ., Blacksburg, VA, USA, Human Factors and Ergonomics Soc. M1: Copyright 2003, IEE; T3: Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting.

[881] G. M. Youngblood, D. J. Cook, and L. B. Holder. Managing adaptive versatile environments. In *IEEE International Conference on Pervasive Computing and Communications*, 2005.

[882] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[883] S. Zaidenberg, P. Reignier, and J. L. Crowley. Reinforcement learning of context models for a ubiquitous personal assistant. In *Proceedings of the 3rd Symposium of Ubiquitous Computing and Ambient Intelligence*, volume 51/2009, pages 254–264, 2008.

[884] A. M Zaremski and J. M Wing. Specification matching of software components. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(4):333–369, 1997.

[885] Amy Moormann Zaremski and Jeannette M Wing. Specification matching of software components. pages 6–17. ACM Press.

[886] J. Zhang, C. K Chang, J. Y Chung, and S. W Kim. WS-Net: a petri-net based specification model for web services. *Proceedings IEEE International Conference on Web Services, 2004*, pages 420–427, 2004.

[887] T. Zhang and B. Brugge. Empowering the user to build smart home applications, 2004.

[888] Y. Zhang, Y. Hou, Z. Huang, H. Li, and R. Chen. A Context-Aware AmI system based on MAS model. pages 703–706.

[889] G. Zhao, B. Luo, and J. Ma. Matching case history patterns in case-based reasoning, 2006.

[890] Weibin Zhao and Henning Schulzrinne. Enhancing service location protocol for efficiency, scalabiliy and advanced discovery. *J.Syst.Softw.*, 75(1-2):193–204, 2005.

[891] Fen Zhu, Matt W Mutka, and Lionel M Ni. Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, 04(4):81–90, 2005.

[892] Feng Zhu, Matt Mutka, and Lionel Ni. Splendor: A secure, private, and Location-Aware service discovery protocol supporting mobile services. page 235. IEEE Computer Society.

[893] Feng Zhu, Matt Mutka, and Lionel Ni. PrudentExposure: a private and user-centric service discovery protocol. *percom*, 00:329, 2004.

[894] Michael zur Muehlen. *Workflow-based Process Controlling*. Logos-Verl., 2004.

[895] Michael zur Muehlen, Jeffrey V Nickerson, and Keith D Swenson. Developing web services choreography standards: the case of REST vs. SOAP. *Decision Support Systems*, 40(1):9–29, 2005.

[896] I. Zuriarrain. Inteligencia ambiental, 2006.