Proyecto de Grado

Integración de Ontologías desde el punto de vista de Minería Ontológica y del Paradigma de Arquitecturas Orientadas a Servicios.

Ing. Carlos Ramón Rangel Piña.

Tutores:

José L. Aguilar
Mariela Cerrada
Junior Altamiranda

Universidad de los Andes, Facultad de Ingeniería

División de Estudios de Postgrado

Programa de Estudios de Postgrado en Computación

Integración de Ontologías desde el punto de vista de Minería Ontológica y del Paradigma de Arquitecturas Orientadas a Servicios.

Ing. Carlos Ramón Rangel Piña.

Proyecto de Grado –Postgrado de Computación, 128 Páginas.

Escuela de Ingeniería de Sistemas, Universidad de Los Andes, 2015.

Resumen: La Integración de Ontologías busca la unificación o enlazado del conocimiento distribuido de varias ontologías. Para ello se consideraran aspectos de la minería ontológica, tales como la mezcla de ontologías o enlazado de ontologías. Hasta ahora, los trabajos que intentan realizar la integración de ontologías manejan estos procesos de manera separada; además, la mezcla que se encuentra con mayor frecuencia en la literatura conlleva a un enriquecimiento de una de las dos ontologías, es decir se copia en ella el conocimiento proveniente de los nodos alineados con la otra ontología, dejando el resto de los nodos no alineados de esa otra ontología por fuera de la ontología resultante de la mezcla. Este proyecto presenta un esquema de integración autonómico de ontologías basado en servicios, para lo cual se propone un middleware de gestión inteligente de los mecanismos de integración ontológica, definidos como servicios de minería ontológica. Adicionalmente, en este trabajo se analiza el proceso de mezcla, y se definen dos tipos de mezclas (mezcla fuerte y mezcla débil). En particular, se diseña un servicio (algoritmo) para el caso de la mezcla fuerte, que intenta resolver el problema planteado de los algoritmos de mezcla que dejan conocimiento por fuera. Finalmente, en el trabajo también se propone otro servicio (algoritmo) para la selección automática de mecanismos de alineación de ontologías (debido a que existen muchos en la literatura), basado en el algoritmo de optimización de las Colonias de Abejas ABC. Los algoritmos fueron probados en diferentes experimentos, obteniendo buenos resultados, en especial el algoritmo de mezcla fuerte pudo ser

comparado con la mezcla clásica y otros algoritmos de mezcla existente en la literatura, obteniéndose resultados bastante prometedores.

Ontology Integration seeks the unification of distributed knowledge or link multiple ontologies. To do so, ontological aspects of ontology mining, such as the merge of ontologies or link ontologies are considered. So far, studies aiming to realize the integration processes of ontologies, handled this separately. In addition, the merge, which is most frequently found in the literature, leads to an enrichment of one of the two ontologies, i.e. the knowledge from the nodes aligned with the other ontology, are been copied, leaving the knowledge from the other nodes unaligned outside the resulting merged ontology. This project presents a scheme of autonomic integration of ontologies, based on services, for which is defined an intelligent middleware management mechanisms for the ontological integration, prepared as ontological mining services. Additionally, in this study, the merge process is analyzed, and two types of merge (strong merge and weak merge) are defined. In particular, a service (algorithm) for the case of strong merge, tries to resolve the problem posed by traditional merge algorithms, which left knowledge outside merge. Finally, other service work (algorithm) for automatic selection of ontology alignment mechanisms (because there are many in the literature), based on the ABC optimization algorithm, an artificial bee colonies is also proposed. The algorithms were tested in different experiments, obtaining good results, especially strong merge algorithm, which could be compared to the classic merge and other algorithms of merge of ontologies in the literature, yielding in very promising results.

Palabras claves: Mezcla de Ontologías, Integración de Ontologías, Minería Semántica, Middleware Reflexivo, Colonias de Abejas, Alineamiento Ontológico, Minería Ontológica, Computación Autonómica.

Índice

Índice de Tablas 8
Índice de Figuras 9
Capítulo 1 11
1.1. Introducción 11
1.2. Formulación del problema 12
1.3. Justificación y relevancia del trabajo 14
1.4. Objetivos 15
1.4.1. Objetivo General. 15
1.4.2. Objetivos específicos. 15
1.5. Antecedentes 16
1.5.1. Mezcladores de ontologías 16
1.5.2. Enlazado de ontologías 21
1.5.3. *Middleware* para desarrollo de DM 23
1.6. Organización de la Tesis 25
Capítulo 2 27
Marco Teórico 27
2.1 Minería Semántica (*Semantic Mining (SM)*) 27
2.1.1 Minería Ontológica (*Ontology Mining (OM)*) 29
2.2 Middleware Reflexivo y Computación Autonómica 48
2.2.1. Arquitectura Orientada en Servicios (SOA) 52
2.3 Colonias de Abejas Artificiales (ABC) 55
Capítulo 3. 58
Algoritmos de Minería Ontológica 58
3.1. Sistema de recomendación de Alineamiento de Ontologías usando ABC 58
3.2. Algoritmos de Mezcla de Ontologías 62

Capítulo 4 77
Diseño del Middleware orientado a la integración autonómica de ontologías 77
4.1. Caracterización del Middleware 77
4.2. Arquitectura del Middleware 81
Monitor 83
Analizador 84
Planificador 88
Ejecutor 90
Capítulo 5 91
5.1. Protocolo experimental para el sistema de recomendación de técnicas de Alineamiento de Ontologías usando ABC 91
5.1.1. Datos de Prueba 91
5.1.2. Parámetros, Medidas de Rendimiento y Técnicas a Evaluar 93
5.1.3. Resultados Obtenidos 95
5.1.4. Análisis de resultados 103
5.2. Protocolo experimental para el Sistema de Mezcla Fuerte de Ontologías 105
5.2.1 Datos de Prueba 106
5.2.2. Medidas de Rendimiento 106
5.2.3. Resultados Obtenidos 106
5.2.4. Análisis de resultados 112
5.3. Protocolo experimental para el Middleware reflexivo de integración de ontologías. 114
5.3.1. Caso basado en usuario 115
5.3.2. Caso basado en eventos 117
5.3.3. Análisis 118
Capítulo 6 120
Conclusiones y Recomendaciones 120
6.1. Conclusiones 120

6.2. Recomendaciones 122

Referencias Bibliográficas 124

Índice de Tablas

Tabla 5.1 Parámetros a considerar en cada uno de los experimentos con el algoritmo de recomendación de técnicas de alineamiento de ontologías. 94
Tabla 5.2 Ontologías de Automóviles. Tamaño de la Colonia al doble de la cantidad de técnicas de alineamientos. 96
Tabla 5.3 Ontologías de Automóviles. Tamaño de la Colonia igual al de la cantidad de técnicas de alineamientos. 97
Tabla 5.4 Ontologías de Automóviles. Tamaño de la Colonia a la mitad de la cantidad de técnicas de alineamientos. 97
Tabla 5.5 Ontologías de la Anatomía del ojo. Tamaño de la Colonia al doble de la cantidad de técnicas de alineamientos. 98
Tabla 5.6 Ontologías de la Anatomía del Ojo. Tamaño de la Colonia igual al de la cantidad de técnicas de alineamientos. 99
Tabla 5.7 Ontologías de la Anatomía del Ojo. Tamaño de la Colonia a la mitad de la cantidad de técnicas de alineamientos. 100
Tabla 5.8 Ontologías del Hardware de un Computador. Tamaño de la Colonia al doble de la cantidad de técnicas de alineamientos. 101
Tabla 5.9 Ontologías del Hardware de un Computador. Tamaño de la Colonia igual al de la cantidad de técnicas de alineamientos. 102
Tabla 5.10 Ontologías del Hardware de un Computador. Tamaño de la Colonia a la mitad de la cantidad de técnicas de alineamientos. 102
Tabla 5.11 Comparación de resultado con [34] 114

Índice de Figuras

Capítulo 1.

Fig. 1.1 COM caso A 17
Fig. 1.2 COM caso B.....19
Fig. 1.3 COM caso C 20
Fig. 1.4 COM caso D 21

Capítulo 2.

Fig. 2.1 Mezcla de A y B 42
Fig. 2.2 Enlazado de A y B 47
Fig. 2.3 Arquitectura de la computación autonómica 50
Fig. 2.4 Arquitectura MAPE.....51
Figura 2.5 ESB. Imagen extraída de http://en.wikipedia.org/wiki/Enterprise_service_bus
.....55

Capítulo 3.

Fig. 3.1 Ciclo de las abejas para encontrar un buen alineamiento.....60
Fig. 3.2 Mezcla débil donde no es dejado conocimiento por fuera 63
Fig. 3.3 Mezcla débil donde un concepto de B no es agregado a C 65
Fig. 3.4 Mezcla fuerte donde se incorpora un concepto de B que no fue agregado a C 66
Fig. 3.5 Mezcla débil donde un concepto de B ancestro no es agregado a C 67
Fig. 3.6 Mezcla fuerte donde el concepto ancestro de B es agregado a C.....68
Fig. 3.7 Mezcla débil donde solo los nodos hojas son alineados 69
Fig. 3.8 Mezcla fuerte donde solo los nodos hojas son alineados.....70
Fig. 3.9 Proceso mezcla débil tradicional, ejemplo mezcla fuerte 75
Fig. 3.10 Nuevo proceso mezcla fuerte.....76

Capítulo 4.

Fig. 4.1 Middleware reflexivo para la integración de ontologías 82
Fig. 4.2 Ciclo MAPE del Middleware reflexivo para la integración de ontologías 83

Capítulo 5.

Fig. 5.1. Ontologías de Automóviles.....92
Fig. 5.2 Ontologías sub anatomía del cuerpo humano 92
Fig. 5.3 Ontologías de computadores.....93
Fig. 5.4 Mezcla de las ontologías sobre automóviles europeos. 108
Fig. 5.5 Mezcla de las ontologías de la anatomía del ojo. 110
Fig. 5.6 Mezcla de las ontologías sobre hardware de computadores. 111
Fig. 5.7 Comportamiento del Middleware para el caso de mezcla fuerte por petición del usuario. 117
Fig. 5.8 Ciclo MAPE para el caso de enlazado débil basada en eventos. 118

Capítulo 1

1.1. Introducción

En el presente trabajo se propone desarrollar un esquema de Integración de Ontologías, el cual busca la unificación en una sola ontología del conocimiento distribuido en varias ontologías. Para ello se consideraran aspectos tales como *mezcla de ontologías* de un mismo dominio con conceptos equivalentes, o *enlazado de ontologías* que pueden estar dentro de un mismo dominio o no. Para este esquema de integración autonómico, se propone un Middleware de gestión inteligente que consta con servicios para realizar la integración de ontologías tales como: mezcla, enlazado, alineamiento o la selección automática de mecanismos de alineación de ontologías (algoritmo de optimización de las Colonias de Abejas ABC). El enfoque a proponer será realizado desde la perspectiva de la minería ontológica (*Ontology Mining*, por su denotación en inglés). La minería ontológica forma parte del ámbito de la minería semántica (*Semantic Mining)*, que consiste en aplicar técnicas de Minería de Datos (conocida por su nombre en inglés *Data Mining (DM)*) para la extracción o creación de contenido semántico desde diferentes fuentes (ontologías, bases de datos, grafos anotados Semánticamente extraídos de la web, o cualquier fuente que permita extraer conocimiento semántico). Dentro del campo de *Semantic Mining* se pueden mencionar las siguientes áreas de estudio: Minería de Datos Semántica (*Semantic Data Mining*, por su denotación en Inglés), Minería de Web Semántica (*Semantic Web Mining*, por su denotación en Inglés) y Minería Ontológica (*Ontology Mining*, por su denotación en Inglés).

Para el problema de integración de ontologías visto desde la Minería Ontológica, algunos conceptos importantes a considerar son: conceptos equivalentes (se refieren al mismo contenido semántico, pueden diferir y estar escritos en diferentes idiomas, o simplemente

con diferente representación), distancia semántica entre conceptos (valor asignado para definir que tanto se parecen dos conceptos), detección de incoherencias (relaciones que contradicen otras relaciones), entre otros.

1.2. Formulación del problema

Hasta ahora, los trabajos que intentan realizar la integración de ontologías manejan los procesos de mezcla como en [1, 18] y de enlazado como en [4, 15] de manera separada, no existe una plataforma de servicio que manejara ambos esquemas de mezcla y enlazado. Por otra parte, sólo se han desarrollado mezcladores de ontologías de manera semiautomática, o automática pero dejando conocimiento de las ontologías a ser mezcladas por fuera de la mezcla. Además, no se toma en cuenta el enlazado entre ontologías al momento de realizar una integración, que se requiere en ontologías que están dentro de una misma área pero no manejan el mismo conocimiento, es decir manejan dominios complementarios. Dichos dominios, al ser integrados dan como resultado un conocimiento más completo dentro de un área, o dentro de áreas diferentes con conceptos comunes. Así, al realizar la mezcla o el enlazado de manera separada, se puede estar dejando conocimiento que perfectamente se podrían complementar.

Un ejemplo clásico donde ocurre esto es cuando el alineamiento (proceso para encontrar correspondencias de similitud semántica, entre los conceptos y relaciones de diferentes ontologías, ya sea con la finalidad de mezclar o enlazar) entre dos ontologías es pobre (no se alinean muchos nodos entre las ontologías, es decir el porcentaje de nodos alineados no es alto en relación a la cantidad de nodos de las ontologías de entrada, esto puede depender de los expertos o del problema), es decir no se alinearon gran cantidad de conceptos y/o propiedades. Si hacemos sólo la mezcla clásica (mezcla que se encuentra con mayor frecuencia en la literatura, que para efectos de este trabajo la llamaremos mezcla débil), ocurriría un enriquecimiento leve de una de las dos ontologías, es decir se copian en

una de las ontologías el poco conocimiento de la otra ontología proveniente de los nodos alineados (conceptos y propiedades), esto deja conocimiento de una de las ontologías por fuera de la ontología mezcla resultado. Por otro lado, si consideramos el enlazado automático (sin la ayuda de un experto, en el ámbito de este trabajo lo llamaremos enlazado débil), entonces sólo se tendría una pequeña ontología de enlace (la cual sirve para navegar y razonar entre ambas ontologías) con los pocos nodos alineados, ya que al no contar con la ayuda de un experto, no se crean conceptos nuevos y relaciones nuevas, que mejoran la ontología intermedia de enlace. En este sentido, el proponer un mecanismo inteligente de gestión de los dos procesos, nos permitirá obtener resultados, tales como, cuál de los dos procesos es más pertinente realizar (mezcla o enlazado, con sus variaciones de fuerte o débil que se exponen en capítulos más adelante), mezclar las ontologías sin dejar conocimiento por fuera (mezcla fuerte), alinear ontologías sin especificar una técnica de similitud semántica, entre otras cosas.

Así, este proyecto presenta un esquema de integración autonómico de ontologías basado en servicios, que explora todos los aspectos mencionados anteriormente, para lo cual se propone hacer una gestión inteligente de los mecanismos de integración ontológica, tales como los de alineamiento, mezclado y enlazado, definidos como servicios. Adicionalmente, en este trabajo se analiza el proceso de mezcla, se definen dos tipos de mezclas (mezcla fuerte y mezcla débil), y se diseña un servicio para el caso de mezcla fuerte. Así también, para el proceso de enlazado, se diferencian dos tipos de enlazados, fuerte con la ayuda del experto, y débil sin la ayuda de un experto. Finalmente, en el trabajo se propone otro servicio para la evaluación y selección automática (sin la intervención de un experto del área del dominio) de mecanismos de alineación, basado en el algoritmo de optimización de las Colonias de Abejas ABC.

1.3. Justificación y relevancia del trabajo

Hasta ahora, es relativamente poca la importancia que se le da al potencial que existe en unificar varias ontologías para manejar un conocimiento integrado. Algunos intentos por hacer esa integración son solo basados en la mezcla de ellas, y se realizan de manera semiautomática (con la supervisión de un experto del área del dominio). Otros solo se preocupan por enlazarlas, dejando por fuera aspectos como que puede existir conocimiento repetido, o incluso llegar a generar inconsistencias. El aporte del presente trabajo será desligar al experto del dominio de conocimiento al momento de integrar ontologías, gestionando de manera inteligente (escogiendo si realizar mezcla o enlazado, y el tipo: fuerte o débil) los procesos de mezclar ontologías que estén en el mismo dominio de conocimiento, y de enlazar ontologías que manejan dominios complementarios, lo que permite tener un marco de integración completo de ontologías.

Realizar una integración completa de ontologías permite tener todo el conocimiento unificado en un mismo lugar (en una misma ontología), lo cual, para algunos procesos puede ser de gran utilidad, ya que al tener el conocimiento distribuido, pueden aumentar los tiempos de consultas entre las ontologías. Para realizar el proyecto se usará el paradigma SOA (*Service Oriented Architecture*), que permite la reutilización de componentes, y se diseñara un middleware autonómico para gestionar el proceso de integración de forma inteligente (sin la intervención de un experto del área de conocimiento).

Por otro lado, es fundamental diferenciar los dos tipos de enfoques de mezcla que actualmente existen, por los tipos de enriquecimientos que permiten. En el caso clásico, una ontología es enriquecida con la información de la otra ontología si sus conceptos existen en esa segunda ontología (lo llamaremos mezcla débil), mientras que el otro caso es cuando se hace una fusión completa de las ontologías con todo el conocimiento contenido en ambas (lo llamaremos mezcla fuerte). En este trabajo se considera esa diferencia, y se propone un

algoritmo para el caso de mezcla fuerte, ya que en la literatura existen diversos algoritmos de mezcla débil, y a nuestro entender, ninguno de mezcla fuerte.

Finalmente, en la literatura existen también diversos algoritmos de alineamiento de ontologías, cada uno con sus ventajas y desventajas, con sus dominios de aplicación, etc. En general, escoger cual algoritmo de alineación es más idóneo en un contexto dado es un problema NP-Completo (esto es porque no es determinístico y además no se puede resolver en tiempo polinómico). Para resolver ese problema, en este trabajo se propone un enfoque de selección automática de mecanismos de alineación, basado en el modelo de Colonia de Abejas denotado ABC, por sus siglas en inglés.

1.4. Objetivos

1.4.1. Objetivo General.

Diseñar procesos de integración de ontologías, enmarcados en una arquitectura autonómica basada en servicios.

1.4.2. Objetivos específicos.

a) Diseñar una middleware reflexivo basado en servicios para la Integración Automática de Ontologías.
b) Especificar los servicios requeridos basados en *Ontology Mining*.
c) Analizar el estado del arte en algoritmos de mezcla y enlazado.
d) Diseñar un algoritmo de mezcla fuerte.
e) Diseñar un algoritmo para la selección automática de alineamiento.
f) Especificar casos de estudio para validar el proceso de integración semántica automática, y los algoritmos de mezcla fuerte y selección automática de alineamiento, propuestos.

1.5. Antecedentes

Las investigaciones en integración de ontología presentan trabajos muy concretos, partiendo siempre que se conoce lo que se va a realizar, ya sea mezclar, como por ejemplo la investigación de Sun, H. [18], o enlazar, como en Mechouche, A. [15] o Zagal R., [21]. Por otro lado, también actualmente se están proponiendo middlewares para tareas de minería de datos, con el fin de optimizar diferentes aspectos de ese proceso, como su ejecución, sus operaciones de E/S, entre otros. En las siguientes secciones hablaremos de trabajos en esos ámbitos.

1.5.1. Mezcladores de ontologías

En el trabajo de Sun, H. [18] se analiza el proceso clásico para mezclar ontologías: mapeo de ontologías, alineamiento de ontologías y mezcla de ontologías. Sun, H. [18] presenta un algoritmo que define la representación del conocimiento basado en grafos, y usa un mecanismo de inferencia *bottom-up* para la comparación de los conceptos basado en un método heurístico semi-automático. Dado que existe una heurística, este algoritmo puede realizar juicios equivocados, así que no se desliga de un experto de dominio para confirmar las relaciones candidatas a ser equivalentes.

Uno de los principales problemas es no poder desligar la necesidad del experto de la mezcla al momento de la inferencia. Cuevas [1] propone un algoritmo para realizar la mezcla de manera automática. Dicho trabajo se aplica para mezclar dos ontologías en forma automática para la obtención de una tercera ontología, tomando en cuenta aspectos como las inconsistencias (que una relación contradiga a otra relación, dentro de una misma ontología), los sinónimos, las contradicciones, y las discrepancias entre las ontologías, de tal manera que el resultado es bastante bueno. Para mezclar las ontologías el algoritmo propuesto es el clásico, a diferencia que para alinear los conceptos utilizan COM, un método

para calcular el concepto más cercano en otra ontología (normalmente se denota cómo el Concepto Más Similar (CMS)). El cálculo de CMS es expresado por Cuevas [1] basado en cuatro casos. Supone que CA sea un concepto o nodo en la ontología A y PA su predecesor. COM busca encontrar el concepto más parecido CB a CA en la ontología B, donde PB es el predecesor del concepto CB. Los cuatro casos que propone el algoritmo son los siguientes:

1. Caso A: el concepto CA coincide con CB en B y los predecesores PA y PB también

En el Caso A, dados CA y PA, COM busca en B por dos conceptos, CB y PB, de manera que la definición de PB coincide con la mayoría de las palabras que definen PA, y la mayoría de las palabras que definen CB coinciden con la definición de CA. En ese caso retorna CB (conocido como CMS), tal como se ilustra en la Fig. 1.1.

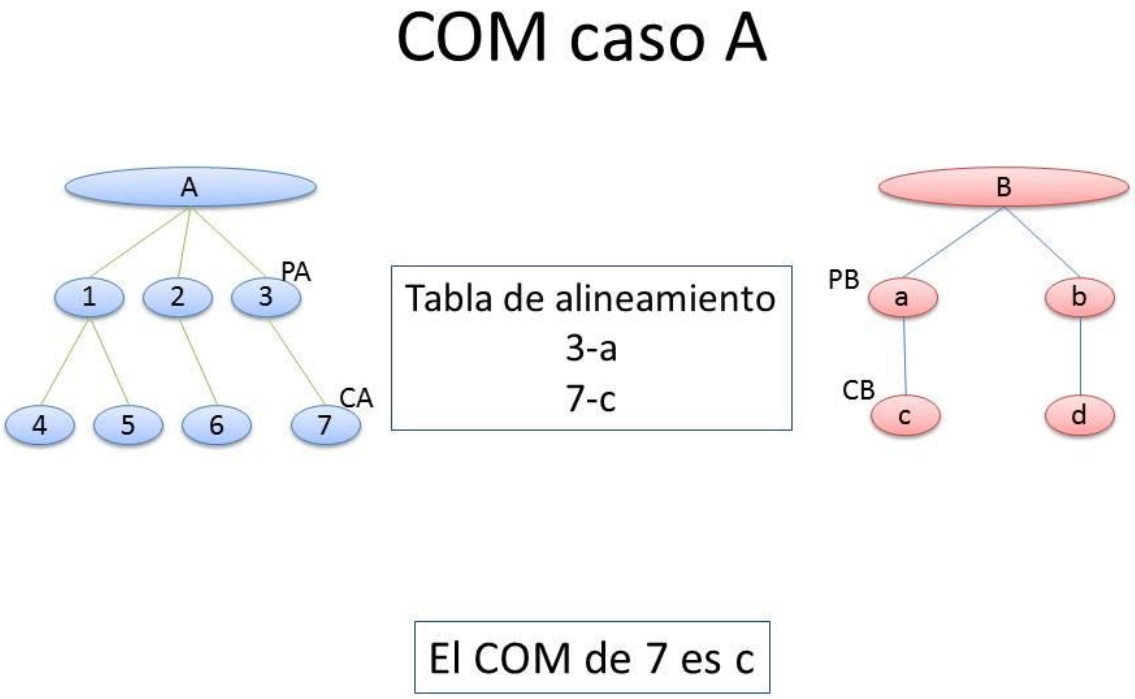

Fig. 1.1 COM caso A

2. Caso B: PA coincide con PB, pero no hay coincidencia entre CA y CB.

PB se encuentra, pero no CB. En este caso, COM de forma recursiva con PA como parámetro, pasa a confirmar que PB es un predecesor de CA. Si un primo de PB (PB') pasa a ser la raíz de la ontología (OBRoot), entonces el algoritmo termina sin éxito. Si eso no sucede, entonces CA se busca en B a través de cada hijo de PB (ese hijo debe coincidir con la mayoría de sus propiedades de CA). Si el candidato CB tiene hijos, se verifica que coincidan con los hijos de CA. Si un CB' se encuentra con las propiedades esperadas de CA, el algoritmo termina devolviendo con éxito CB'. De lo contrario, COM intenta encontrar CB' entre los hermanos de PB. Si eso no sucede, se busca entre los nietos de PB. Si CB' no se encuentra, entonces el valor más cercano a CA es un hijo desconocido (que no está presente) de PB, por lo tanto, COM devuelve "hijo de PB" (lo que significa que un hijo PB que no existe todavía en B es el concepto más similar a CA), un ejemplo de este caso es ilustrado en la Fig. 1.2.

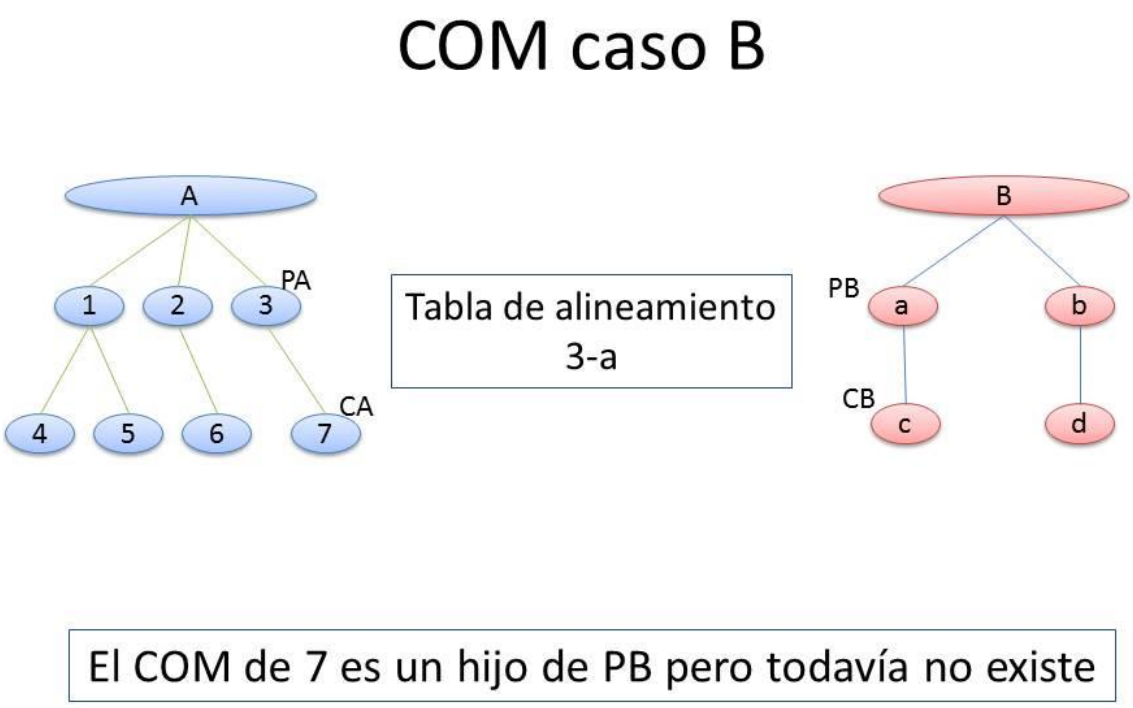

Fig. 1.2 COM caso B

3. Caso C: CA coincide con CB, pero no hay ninguna coincidencia entre PA y PB.

Si CB se encuentra, pero no PB, entonces COM comprueba si el abuelo de CB en B es similar a PA, o si el bisabuelo de CB en B es similar a PA. Si este es el caso, entonces el concepto más similar de PA en B es el abuelo o bisabuelo de CB y se termina el algoritmo. Si no se encuentra, entonces COM verifica si la mayoría de las relaciones y los valores de CA coinciden con los de CB y si la mayoría de los hijos CA coinciden con la mayoría de los hijos de CB. Si las propiedades y los hijos coinciden, entonces la respuesta es CB y el algoritmo termina, a pesar de que PB no se ha encontrado en B. Si sólo una parte de las propiedades y los hijos coincide entonces la respuesta es "Probablemente CB" y se termina el algoritmo. Si no hay propiedades ni los hijos son iguales, entonces la respuesta es "no existe" y el algoritmo concluye, tal como muestra la Fig. 1.3.

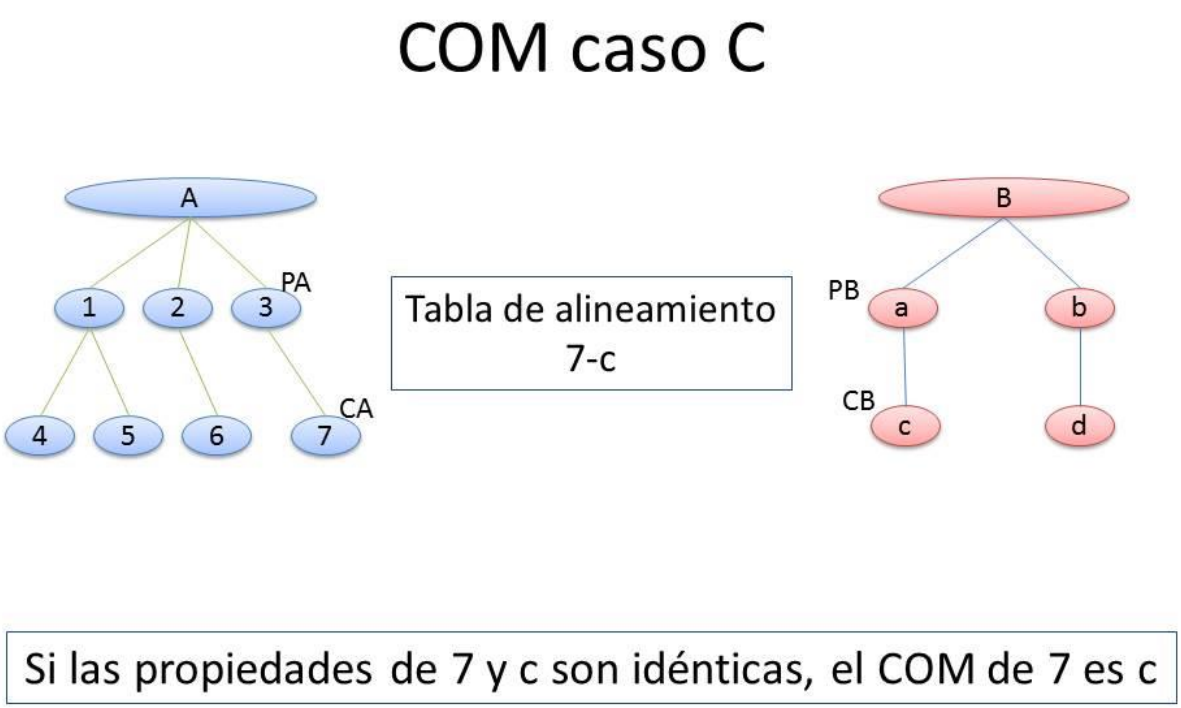

Fig. 1.3 COM caso C

4. Caso D: CA no coincide con el CB y PA no coincide con PB.

Si CB no existe y tampoco PB, entonces la respuesta de la COM es "no existe " y termina el algoritmo.

Este trabajo de Cuevas [1] deja por fuera conocimiento al mezclar, al no copiar conceptos y propiedades que no tiene relaciones con los conceptos alineados. Existen muchos otros trabajos en el área, como [1, 34, 36, 37], pero el de Cuevas [1] es el que más se asemeja a lo requerido por este estudio, a pesar de dejar por fuera conocimiento al mezclar, como también ocurre prácticamente con todos los algoritmos de mezcla de la actualidad. Este trabajo propone el enfoque de mezcla fuerte, en el que se pretende no dejar conocimiento sin ser mezclado, ilustrado en la Fig. 1.4.

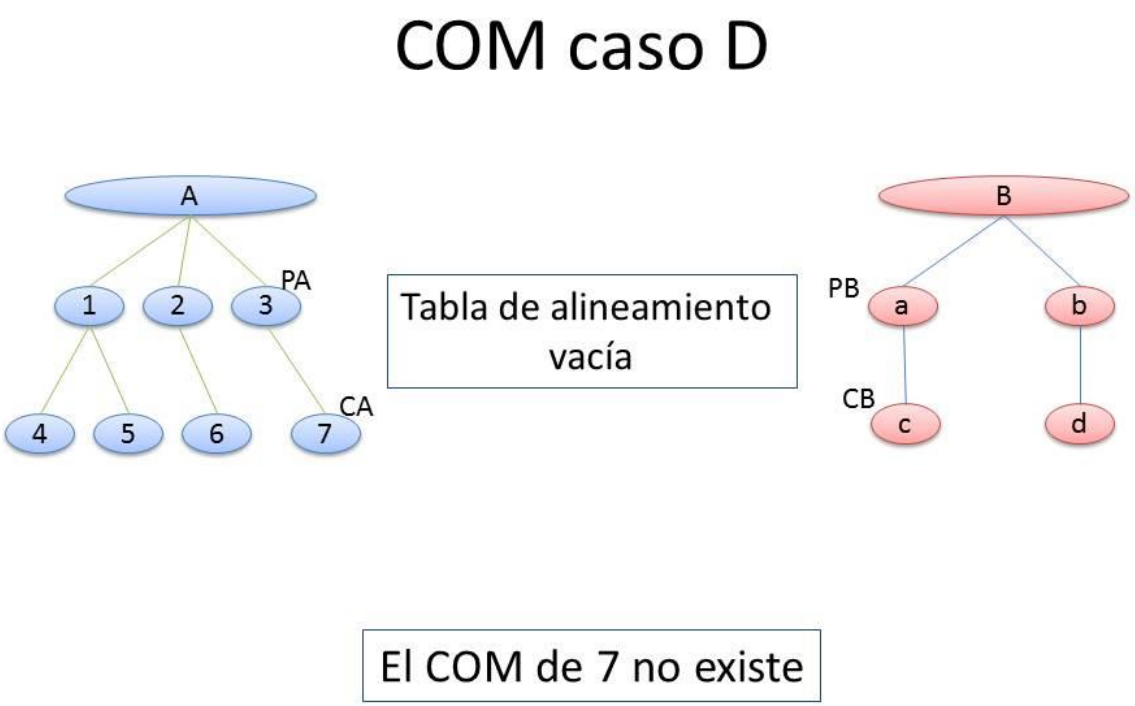

Fig. 1.4 COM caso D

1.5.2. Enlazado de ontologías

El objetivo del trabajo de Niang [4], es la creación de una ontología de dominio para proporcionar un vocabulario conceptual en común a los miembros de una comunidad virtual de usuarios (como médicos, turismo, banca, agricultura, entre otros). La identificación y definición de los conceptos que describen el conocimiento del dominio requiere un cierto consenso. Algunos expertos tienen su propia opinión sobre el dominio, y con ella describe su propio vocabulario. Por lo tanto, para llegar a un consenso se debe reflejar una visión común del dominio, esto puede ser una tarea difícil, y aún más difícil si los miembros están geográficamente dispersos. Una forma es utilizando los elementos del dominio: cuerpo de textos, taxonomías, fragmentos de una ontología, y explotarlos como base para definir gradualmente la ontología de dominio.

En este trabajo utilizan una aproximación utilizando técnicas de *Matching* de ontologías (alineamientos de ontologías) para la construcción de una ontología de dominio a la medida, a partir de una taxonomía de dominio general y varias piezas de conocimiento dadas por diferentes socios. La estrategia usada es el diseño de un mediador, para llegar a un acuerdo con cada socio en sus fragmentos de conocimientos que serán proporción a la ontología de dominio compartido, y de conciliar los diversos fragmentos mediante la vinculación de sus conceptos. Como ontología del mediador, en el caso de estudio utilizaron una taxonomía pública que existe para describir los dominios de agricultura, silvicultura, pesca, alimentación y ámbitos relacionados (como el medio ambiente), llamada AGROVOC3. La ontología resultante del dominio combina las dos características siguientes: (i) una taxonomía general vinculada al dominio de aplicación considerado, (ii) completada (enlazada) con las relaciones y propiedades procedentes de los fragmentos de conocimiento de los socios. La propuesta en [4] depende de un experto para la toma de decisiones cruciales, como la creación de nuevos conceptos, que no se contemplan en ninguna de las ontologías a ser enlazadas.

Por otro lado, Parundekar [35] dice que la vinculación de datos estructurados desde diferentes fuentes se puede realizar mediante declaraciones de equivalencia, como owl:sameAs (relación en formato owl, para identificar correspondencia de identidad entre conceptos y entre relaciones). En ese trabajo se describe un enfoque de generación de relaciones de equivalencia y subsunciones entre clases de ontologías de diferentes fuentes de datos, apoyadas por las declaraciones de equivalencia existentes. También, en [35] son capaces de generar una jerarquía de las clases derivadas, o generar nuevas clases desde las fuentes ontológicas. El enfoque ha sido utilizado por ontologías en el dominio geoespacial, genético y en zoología, entre otros. El algoritmo ha logrado descubrir más de 800 equivalencias y de 29.000 relaciones de subconjuntos entre dos ontologías.

Otros trabajos de enlazado de ontologías son [4, 35, 39], los más significativos para nuestro trabajo son los que vienen de ser descritos. Ahora bien, esos trabajos tienen sus limitaciones, [4] depende de un experto para la toma de decisiones cruciales, y [35] ha fijado un número ideal de diez pares de instancias alineadas para proponer un enlace entre dos ontologías, de tal manera que la eficacia del enfoque debe ser verificada cuando las fuentes son pequeñas o muy grandes.

1.5.3. *Middleware* para desarrollo de DM

La cantidad de datos disponibles para el análisis se ha disparado en los últimos años, haciendo la escalabilidad de las implementaciones de minería de datos un factor crítico. Con este fin, se han desarrollado versiones paralelas de la mayor parte de las técnicas de minería de datos conocidas. Sin embargo, la experiencia y el esfuerzo requeridos actualmente en implementación, mantenimiento y optimización del rendimiento de una aplicación de minería de datos en paralelo es un impedimento grave.

En general, en la literatura existen muchos trabajos sobre middlewares reflexivos y autonómicos [22, 26, 41], así como muchas aplicaciones middlewares para entornos empresariales [25, 30, 33], pero las dos más cercanas a nuestro trabajo, que tienen que ver con middlewares para procesamiento de grandes cantidades de datos, son las presentadas en este apartado, solo que las mismas no consideran aspectos semánticos, cómo lo son las ontologías dentro de una empresa, ni son autonómicas al momento de guiar el proceso de análisis semántico de grandes volúmenes de datos. Por eso, este estudio propone un Middleware reflexivo autonómico (interfaz para servicios, con la capacidad de razonar y actuar sobre si misma) para el análisis semántico de grandes volúmenes de datos.

Jin y Agrawal [5] presentan el diseño y evaluación de rendimiento inicial de un middleware para permitir el rápido desarrollo de aplicaciones de minería de datos en

paralelo. Este middleware puede ayudar a aprovechar el paralelismo en memoria, al tiempo que permite el procesamiento eficiente de los datos residentes en disco. El middleware se basa en la observación de que las versiones paralelas de varias técnicas de minería de datos conocidos comparten una estructura relativamente similar. Este middleware explota en paralelo diferentes nodos de un clúster (utilizan pase de mensajes para la comunicación) y diferentes procesadores en un nodo (que comparten una memoria común). Se permite un rendimiento alto de E/S de disco, minimizando el tiempo de búsqueda. Por lo tanto, se puede utilizar para desarrollar aplicaciones de minería de datos paralelas eficientes, que operan en grandes conjuntos de datos. Ese middleware explota tanto la memoria compartida como la memoria distribuida. Por lo tanto, es especialmente adecuado para grupos de estaciones de trabajo SMP. En concreto, demostraron que: 1) la paralelización de memoria distribuida alcanza una eficiencia muy alta, y 2) la paralelización de memoria compartida logra un buen rendimiento si la aplicación no tiene muchas E/S.

Otro ejemplo interesante de plataforma de middleware de procesamiento de grandes volúmenes de datos, con procesamiento paralelo y distribuido, es la propuesta de Hitachi [36]. Hitachi es una compañía proveedora de una gran gama de diferentes soluciones, y una de ellas es el Servicio de Procesamiento Distribuido para Big Data. El middleware de procesamiento paralelo y distribuido de Hitachi posee características que incluyen la facilidad para el procesamiento por lotes [36]. Hitachi también ofrece servicios de consultoría sobre cómo analizar y utilizar datos grandes, cómo usar PaaSs (plataforma como servicio) como el de ellos, cómo usar los productos de Hitachi para Soluciones de Cloud Computing, etc.

En la revisión del estado de arte para el desarrollo de este trabajo, los trabajos previos han dejado espacios para continuar estudios, por ejemplo, [1] no incluye todo el conocimiento de las ontologías de entrada en el proceso de mezcla, [4] deja decisiones para

el experto del dominio, y [5] no provee servicios para el análisis semántico de la información en grandes volúmenes de datos.

El enfoque propuesto en este trabajo, apunta a solucionar esas brechas, planteando un diseño de un middleware integrador de ontologías, el cual desliga al usuario lo más posible de las tomas de decisión, además con dos servicios nuevos (que no se encontraron en la revisión del estado del arte), de mezclar sin dejar por fuera conocimiento (mezcla fuerte), y de seleccionar de manera automática el mecanismo de alineamiento a usar (utilizando el algoritmo basado en Colonia de Abejas ABC). Por supuesto, el middleware también permite que otros servicios puedan ser usados, como los servicios existentes de mezcla débil, de enlazado (débil y fuerte), de mezcla vía conceptos, de invocación directa a servicios de alineación, entre otros.

1.6. Organización de la Tesis

El capítulo 1 realiza una introducción al proyecto, planteando el problema y presentando los objetivos del trabajo, para culminar con algunos antecedentes en el área.

El capítulo 2 presenta las bases teóricas del proyecto, entre las cuales están: el área de minería semántica, con conceptos específicos, tales como: mezcla de ontologías, enlazado de ontologías y alineamiento de ontologías. También se presentan aspectos teóricos vinculados a los middleware reflexivos y a las colonias de abejas artificiales, por ser usados en nuestro trabajo.

El capítulo 3 presenta las herramientas de Minería Ontológica. En particular, se presenta el diseño e implementación de un sistema de recomendación de Alineamiento de Ontologías usando ABC, y un sistema de Mezcla de Ontologías.

El capítulo 4 presenta el Diseño del *Middleware* orientado a la integración autonómica de ontologías. El cual abarca el diseño del ciclo MAPE del *Middleware* reflexivo.

El capítulo 5 presenta los experimentos analiza los resultados. Comprende las pruebas realizadas a las implementaciones del capítulo 4, los resultados obtenidos de la ejecución del mezclador, la comparación de la completitud y de otras métricas con otros mezcladores, entre otras cosas.

Finalmente, el capítulo 6 presenta las conclusiones y los trabajos futuros.

Capítulo 2

Marco Teórico

En este capítulo se presentan las bases teóricas para el entendimiento del trabajo realizado, comenzando por definir todo lo que comprende el área de Minería Semántica: Minería de datos semántica, Minería web semántica y Minería ontológica. Para el caso concreto de la minería ontológica, por ser el ámbito en el que se desarrolla el trabajo, presentamos en detalle las técnicas más comunes en esta área: los algoritmos de mezcla, enlazado y alineamiento de ontologías. También se definen qué es Middleware reflexivo y Computación Autonómica, así como qué son los Buses de Servicios. Finalmente, se presenta el algoritmo de colonias artificiales de abejas ABC. Todos estos aspectos teóricos serán usados en nuestro trabajo.

2.1 Minería Semántica (*Semantic Mining (SM)*)

La minería semántica se encarga de extraer conocimiento semántico desde diferentes fuentes semánticas, como lo son páginas web, contenido sin estructura en la web, contenido estructurado en la web, grafos anotados, ontologías, entre otros. La Minería Semántica se divide en tres grandes grupos, Minería de datos semántica, Minería web semántica y Minería ontológica, los cuales se describen detalladamente a continuación.

Uno de los desafíos de la Minería de Datos (DM por sus siglas en inglés *Data Mining*) ha sido incorporar conocimiento de un dominio desde los datos, esto lo expresó Fayyad, U. en su publicación "*From data mining to knowledge discovery in databases*" (1996). Un reto en particular es incorporar contenido semántico a los datos. El añadir contenido semántico a los

datos y usar técnicas de DM para la extracción de conocimiento (que en este caso vendrá con contenido semántico) se le conoce como *Minería de Datos Semántico* (Semantic Data Mining del inglés (SDM)).

El proceso de SDM se da en dos pasos, un primer paso de enriquecimiento semántico y un segundo de la aplicación de técnicas de DM como tal. *En el primer paso* se usan ontologías, o cualquier contenido semántico, y se realiza un mapeo con la data que se va a trabajar almacenada en bases de datos, que consiste en tomar los datos y buscar su equivalente en el contenido semántico. Dicho contenido semántico se agrega a los datos, para así extraer patrones con técnicas de DM que contenga el contenido semántico agregado. Otra manera de agregar el contenido semántico es, en vez de sólo agregarlo a los datos, sustituir los datos por sus equivalentes con contenido semántico. Después, en un *segundo paso* se aplican técnicas de DM para buscar patrones. Un ejemplo común es la extracción de reglas a través de técnicas de DM, en bases de datos que han sido enriquecidas Semánticamente. La DM detecta patrones de comportamiento de los históricos de datos, y al estar enriquecidos Semánticamente permite crear reglas más cercanas al lenguaje natural, dichas reglas dan más aporte a los expertos del área.

La integración de dos áreas de conocimiento, como lo son la *Web Semántica* y las técnicas de *Web Mining,* es conocida como *Semantic Web Mining* (SWM), según [16]. La *Semantic Web* es usada para darle significado a los datos que se encuentran en la Web, por otro lado, *Web Mining* son técnicas de DM (como por ejemplo las técnicas de *text mining*) usadas para extraer patrones de la Web, dichos patrones pueden ser de: uso, contenido y estructura. La web semántica es expresada en formatos como OWL, RDF, e incluso XML, y estos recursos son minados para extraer conocimiento. SWM y SDM tienden a confundirse, la diferencia primordial de SWM con SDM es el propósito y lo que se está minando, para SWM se minan datos de la *Semantic Web,* y el propósito es usar los patrones para mejorarla y

enriquecerla, por otro lado, SDM busca resultados cercanos al lenguaje natural y se minan datos almacenados en memoria como lo son las bases de datos.

Como ya se ha mencionado, para entrar en la categoría de SWM se deben minar datos de la Web, y los resultados serán usados por la Web Semántica. Para esta tarea existen varios tipos de *Minería Web* que se pueden aplicar dentro del SWM, enfocándose en lo que se puede minar: el contenido de la web, la estructura de la web y el uso que se hace de la web. El minado de contenido básicamente es una forma de *Minería de Texto*, que se aplica al contenido desplegado en la Web. Por ejemplo, se toma una página y se buscan términos concurrentes, o características del lenguaje en el que están escritos los textos, o se extraen entidades y relaciones entre esas entidades, todo lo anterior para construir reglas semánticas. El minado de la estructura estudia el esqueleto que forman los enlaces entre las páginas de la Web, es decir, se mina un conjunto de páginas y sus enlaces usando técnicas de análisis de grafos, para extraer conocimiento de qué páginas son más centrales, puntos débiles de conectividad, entre otros. Por otro lado, el minado del uso de la web se enfoca en minar el historial de uso de los usuarios, se busca descubrir los patrones de comportamiento en las consultas que hacen a una página, los movimientos que se hacen entre páginas, entre otros. Es usual que en el minado del uso se utilicen las otras dos técnicas (de estructura y de contenido), haciendo al minado de uso la más completa forma de *Minería Web* y la más usada [13].

2.1.1 Minería Ontológica (*Ontology Mining (OM)*)

La extracción de patrones de comportamiento, de conocimiento, entre otras características, usando las técnicas de DM, con la finalidad de construir o enriquecer ontologías, es conocida como *Minería Ontológica* (OM). Actualmente, con el gran crecimiento en las cantidades de ontologías disponibles es necesaria el área de OM, para explorar técnicas que puedan extraer conocimiento global de un conjunto de ontologías.

Algunas de las técnicas que se han venido desarrollando son de enlazado, mezcla y alineamiento entre varias ontologías.

2.1.1.1 Alinear Ontologías

La alineación consiste en el análisis de correspondencias entre dos o más ontologías (este proceso es la base para la mezcla y el enlazado, descritos en puntos más adelante). La alineación consiste en realizar la comparación (*matching)* entre los conceptos de las ontologías que se estén analizando, es decir, es el proceso de encontrar relaciones o correspondencias entre entidades de diferentes ontologías [28]. El proceso de alineación de ontologías está compuesto de forma general por los siguientes elementos: dos ontologías O1 y O2, un conjunto *p* de requisitos, un conjunto *r* de recursos para la alineación, y una función *f* de alineación que retorna un conjunto de correspondencias *A'*. Se detalla a continuación cada uno [21]:

- La función *f* es una función que evalúa y obtiene las correspondencias entre los conceptos, según los recursos *r*.
- En cada O1 y O2 se analizan los siguientes elementos: conceptos, propiedades de los conceptos y jerarquía de conceptos.
- El conjunto *p* representa los requisitos para realizar la alineación; *p* = {lenguaje de diseño (por ejemplo, OWL), número de elementos, vocabulario del idioma, etc.}.
- El conjunto de recursos se refiere a los aspectos matemáticos y computacionales empleados para realizar la correspondencia: *r* = {conjunto de medidas de similitud, de algoritmos y herramientas}.
- El conjunto *A'* simboliza todas las correspondencias semánticas encontradas.

En el estudio en [21] se encuentran varios recursos para realizar la alineación de ontologías, como por ejemplo Falcon-AO, que es una herramienta para alineación de ontologías basada en dos algoritmos: uno llamado LMO, basado en similitud lingüística (*linguistic matching*), y otro basado en similitud de grafos (*graph matching*) llamado GMO.

2.1.1.1.1 Medidas de Similitud Semántica

El objetivo de alinear ontologías es encontrar relaciones entre las entidades expresadas en diferentes ontologías. Estas relaciones a descubrir son de equivalencia, que se deben determinar a través de medidas de similitud entre estas entidades. Hay muchas formas de definir formalmente la similitud entre dos entidades, por ejemplo, en [28] dan las siguientes definiciones:

Una Similitud σ: o × o ⟶ R es una función de un par de entidades a un número real, que expresa la similitud entre dos objetos, tales que:

$\forall x, y \in O, \sigma(x, y) \geq 0$ (positividad)

$\forall x, y, z \in O, \sigma(x, x) \geq \sigma(y, z)$ (maximalidad)

$\forall x, y \in O, \sigma(x, y) = \sigma(y, x)$ (simetría)

Donde O es el conjunto de las ontologías O_i a comparar. La disimilitud es una operación dual. Dado un conjunto o de entidades, una disimilitud δ: o × o ⟶ R es una función de un par de entidades a un número real, tal que:

$\forall x, y \in O, \delta(x, y) \geq 0$ (positividad)

$\forall x \in O, \delta(x, x) = 0$ (minimalidad)

$\forall x, y \in O, \delta(x, y) = \delta(y, x)$ (simetría)

Una distancia (métrica) δ: o × o ⟶ R es una función de disimilitud que satisface la precisión y la desigualdad triangular:

$\forall x, y \in O, \delta(x, y) = 0$ si y sólo si x = y (definiteness)

$\forall x, y, z \in O, \delta(x, y) + \delta(y, z) \geq \delta(x, z)$ (desigualdad triangular)

Pasemos ahora a describir algunas de las medidas de similitud semánticas.

2.1.1.1.2 Técnicas de Similitud basadas en nombre

Algunos métodos terminológicos comparan cadenas [28], estos pueden ser aplicados a nombres, etiquetas o comentarios de entidades, con el fin de encontrar los que son similares.

Formalmente se pueden definir de la siguiente manera: El conjunto S representa al conjunto de cadenas, es decir, las secuencias de letras de cualquier longitud de un alfabeto L: S = L *. La cadena vacía se denota por ε, y $\forall$ s, t ∈ S, s + t es la concatenación de las cadenas de s y t. | s | denota la longitud de la cadena s, es decir, el número de caracteres que contiene. s[i] para i ∈ [1 …| s |] indica la letra en la posición i de s.

Un ejemplo es la cadena 'article', cadena que se hace de las letras a, r, t, i, c, l, e. Su longitud es de 7 caracteres. 'Peer-reviewed' y ' ' son otras dos cadenas (por lo que '-' y ' ' son letras del alfabeto), y su concatenación ‘peer-reviewed’ +‘ ’+ ‘article’ ofrece la cadena 'peer-reviewed article' cuya longitud es de 21.

Una cadena s es la subcadena de otra cadena t, si existen dos cadenas de s’ y s’’, tal que s' + s + s'' = t (denotado por s ∈ t). Dos cadenas son iguales (s = t) si y sólo si s ∈ t y t ∈ s. El número de apariciones de s en t (denotado por s # t) es el número de instancias de

pares de s', s'' (pueden ser distintos o no), tal que s' + s + s'' = t, donde s representa cualquier tipo de cadena perteneciente a t.

El principal problema en la comparación de las entidades de una ontología sobre la base de sus etiquetas se produce debido a la existencia de sinónimos y homónimos:

Los sinónimos son diferentes palabras utilizadas para designar la misma entidad; Los homónimos son palabras que se usan para nombrar diferentes entidades. En consecuencia, no es posible deducir con certeza que dos entidades son las mismas si tienen el mismo nombre, o que son diferentes porque tienen diferentes nombres. Pero estas no son las únicas razones que dificultan saber cuándo diferentes palabras son sinónimas u homónimas, hay otras razones, en particular:

- Pueden ser palabras diferentes en cada idioma (inglés, francés, italiano, español, alemán, griego), que se utilizan para nombrar las mismas entidades. Por ejemplo, la palabra Libro en Inglés es Book y Livre en francés.

- Pueden haber variaciones sintácticas de la misma palabra que a menudo se pueden usar, son diferentes ortografías aceptables: abreviaturas, uso de prefijos o sufijos opcionales, etc. Por ejemplo, discos compactos, CD, CD y CD-ROM puede considerarse equivalente en algunos contextos. Sin embargo, en otros contextos CD puede significar Corps Diplomatique y en otros cambiar de directorio.

Algunos métodos para hallar la similitud semántica basadas en nombres son los siguientes:

Igualdad de Cadena: devuelve 0 si las cadenas bajo consideración no son idénticas y 1 si son idénticas. Esto puede ser tomado como una medida de similitud.

σ: S × S ⟶ [0 1] tal que $\forall$ x, y $\in$ S, σ (x, y) = 1 si x = y, de lo contrario σ (x, y) = 0.

Se puede realizar después de cierta normalización sintáctica de la cadena, por ejemplo, la conversión a minúsculas, conversión de codificaciones, normalización diacrítica[1], entre otros.

Esta medida no explica cómo las cadenas son diferentes. Una forma más inmediata de la comparación de dos cadenas es la distancia de Hamming, creada en 1950, que cuenta el número de posiciones en las que las dos cadenas difieren, a continuación [28] presenta la versión normalizada por la longitud de la cadena más larga. La distancia de Hamming es una disimilitud δ: S × S ⟶ [0 1] tal que,

$$\delta(s,t) = \frac{\left(\sum_{i=1}^{\min(|s|,|t|)} s[i] \neq t[i]\right) + ||s| - |t||}{\max(|s|,|t|)} \quad (2.1)$$

Prueba de Subcadena: diferentes variaciones pueden ser obtenidas de la igualdad de cadenas, como la de considerar que las cadenas son muy similares cuando se trata de una subcadena de otra [28]. Esta prueba es una similitud σ: S × S ⟶ [0 1], tal que $\forall$ x, y $\in$ S, si existen p, s $\in$ S donde x = p + y + s, o y = p + x + s, entonces σ (x, y) = 1, de lo contrario σ (x, y) = 0. Es una similitud que mide la relación de la subparte común entre dos cadenas.

Similitud de Subcadena: es una similitud σ: S × S ⟶ [0 1], tal que $\forall$ x, y $\in$ S, y sea t la subcadena más larga común de x e y [28]:

$$\sigma(x,y) = \frac{2|t|}{|x|+|y|} \quad (2.2)$$

Esta definición se puede utilizar para las funciones de construcción basadas en el prefijo o sufijo común más largo.

1 Consiste en reemplazar los signos diacríticos de una palabra (ejemplos de los signos diacríticos son los acentos ortográficos ( ), la diéresis (¨), la virgulilla de la ñ, etc.) con sus sustitutos más frecuentes. Por ejemplo, la sustitución de Montréal por Montreal [28].

Así, por ejemplo, la similitud entre Article y Aricle sería 8/13 = 0,61, mientras que entre Article y Paper sería 1/12 = 0,08, por otro lado, entre Article y Particle sería 14/15 = 0,93.

La similitud n-gram: también se utiliza a menudo en la comparación de cadenas. Se calcula el número de n-grams comunes, es decir, cadenas de n caracteres. Por ejemplo, trigramas de la cadena Article son art, rti, tic, icl, cle. [28] define ngram (s, n) como el conjunto de subcadenas de s de longitud n. La similitud n-gram es una similitud σ: S × S → R, tal que:

$$\bar{\sigma}(s,t) = |ngram(s,n) \cap ngram(t,n)| \quad (2.3)$$

La versión normalizada de esta función es:

$$\bar{\sigma}(s,t) = \frac{|ngram(s,n) \cap ngram(t,n)|}{\min(|s|,|t|)-n+1} \quad (2.4)$$

Esta función es muy eficiente cuando sólo algunos caracteres faltan. Así por ejemplo, la similitud entre Article y Aricle sería 2/4 = 0.5, mientras que entre Article y Paper sería 0, entre Article y Particle sería 5/6 = 0,83.

Distancia de edición: Intuitivamente es una distancia de edición entre dos objetos, es el costo mínimo de operaciones que deben ser aplicadas a uno de los objetos con el fin de obtener el otro. Las distancias de edición fueron diseñadas para medir la similitud entre las cadenas que pueden contener errores de ortografía. En [28] la definen formalmente como: dado un conjunto op de operaciones sobre cadenas (op: S → S), y una función de costo w: op → R, tal que para cualquier par de cadenas existe una secuencia de operaciones que transforma la primera en la segunda (o viceversa), la distancia de edición es una disimilitud:

δ: S × S → [0 1], donde δ (s, t) es el costo de la secuencia de las operaciones menos costosas que transforman s en t.

$$\delta(s,t) = \min_{(op_i)_I; op_n(\ldots op_1(s))=t}(\sum_{i\in I}\omega_{op_i}) \quad (2.5)$$

Las operaciones consideradas habitualmente en la distancia de edición incluyen la inserción de un carácter ins(c, i), la sustitución de un carácter por otro sub(c, c', i), y la supresión de un carácter del(c, i). Se puede comprobar fácilmente las siguientes relaciones entre las operaciones: ins(c, i) = del(c, i)$^{-1}$, que sub(c, c', i) = sub (c', c, i)$^{-1}$ [28]. A cada operación se le asigna un costo, y la distancia entre dos cadenas es la suma de los costos de cada operación en el conjunto de operaciones menos costosas, para obtener una cadena desde la otra.

La distancia Levenshtein (Levenshtein 1965) es el número mínimo de inserciones, supresiones y sustituciones de caracteres requeridos para transformar una cadena a la otra. Es la distancia de edición cuando todos los costos son iguales a 1 [28]. La distancia Damerau-Levenshtein (Damerau 1964) utiliza además la operación de transposición, intercambiando dos letras adyacentes, con el mismo peso que las otras operaciones [28]. La distancia Needleman-Wunch (Needleman y Wunsch 1970) es la distancia de edición con unos costos más altos para ins y del [28].

2.1.1.1.3 Técnicas basadas en estructuras

Las estructuras de las ontologías pueden ser comparadas, en lugar de, o además de la comparación de sus nombres o identificadores. Esta comparación se puede subdividir en una comparación de la estructura interna de una entidad, es decir, además de su nombre y anotaciones, sus propiedades (o en el caso de ontologías OWL, las propiedades que tienen sus valores en un tipo de datos), o la comparación de la entidad con otras entidades a las que se relaciona [28]. A la primera se le llama estructura interna, y a la segunda se le llama estructura relacional. La estructura interna es la comparación de la definición de las

entidades sin referencia a otras entidades; la estructura relacional es el análisis del conjunto de relaciones que una entidad tiene con otras entidades. La estructura interna es explotada principalmente en esquemas de base de datos, mientras que la estructura relacional es más importante en la adecuación de ontologías formales y redes semánticas.

En la literatura, a los métodos basados en estructuras internas se les llama a veces enfoques basados en restricciones [34]. Estos métodos se basan en la estructura interna de las entidades, y se usan criterios tales como el conjunto de sus propiedades, la gama de sus propiedades (atributos y relaciones), su cardinalidad o multiplicidad, y la transitividad o la simetría de sus propiedades, para calcular la similitud entre ellos. Las entidades con estructuras internas comparables, o propiedades con dominios similares en dos ontologías, pueden ser numerosas. Por esa razón, este tipo de métodos se utilizan comúnmente para crear grupos de correspondencia, en lugar de descubrir las correspondencias exactas entre entidades.

Por lo general, las técnicas de estructuras se combinan con otras técnicas a nivel de los elementos, como los métodos anteriores de similitud léxica, y son responsables de reducir el número de correspondencias candidatas. Se pueden utilizar con otros enfoques, como una etapa de preprocesamiento. Veamos algunas de las técnicas a continuación:

Estructura taxonómica: la relación subClassOf es la columna vertebral de las ontologías. Por esta razón, se ha estudiado en detalle como una fuente de comparación para las clases [28]. Ha habido varias medidas propuestas para la comparación de clases basadas en esa estructura taxonómica. Las más comunes se basan en contar el número de arcos en la taxonomía entre dos clases. Otra forma es determinar la disimilitud topológica estructural en una jerarquía, definida en [28], como: δ: o × o → R es una disimilitud de una jerarquía H = (o, ≤), tal que:

$$\forall e, e' \in o, \qquad \delta(e, e') = \min_{c \in o}[\delta(e, c) + \delta(e', c)] \tag{2.6}$$

Donde δ (e, c) es el número de aristas intermedias entre un elemento e y otro elemento c. Esta función puede ser normalizada por la longitud máxima de un camino entre dos clases en la taxonomía:

$$\bar{\delta}(e, e') = \frac{\delta(e,e')}{max_{c,c' \in o}\delta(c,c')} \quad (2.7)$$

2.1.1.1.4 Criterios de Rendimiento para el Alineamiento de Ontologías

Los algoritmos y herramientas que se diseñan para alinear (tal como la adaptación del algoritmo ABC presentado al final de este capítulo), necesitan de criterios para evaluar que alineamiento es mejor que otro. La evaluación de alineamientos consiste en evaluar las propiedades de la alineación obtenida. Se puede realizar manual o automáticamente. De manera manual, se logra mediante la comparación del alineamiento que se obtuvo, con un alineamiento conocido deseado (para esta clase de pruebas, es necesario conocer de antemano el mejor alineamiento entre las dos ontologías de entrada), o preguntando a un experto para evaluar la calidad de la alineación [28]. Para ello, herramientas gráficas que permitan a los usuarios navegar de forma rápida tanto en la alineación y en las ontologías son invaluables. La evaluación automática es cuantitativa, se puede realizar mediante el uso de técnicas para la evaluación de alineaciones como la completitud (que tan completo es el alineamiento, en relación a los conceptos que incorpora de las ontologías de entrada), presentadas en [28].

Algunos criterios de rendimiento de completitud, que se usan para evaluar la alineación de ontologías son:

- #*Conceptos*: medida de rendimiento que indica cuantos conceptos logra alinear la técnica seleccionada en esa corrida. Es importante al momento de evaluar, ya que indica la completitud que abarca el alineamiento de los conceptos de las ontologías de entrada.
- #*Relaciones:* medida de rendimiento que indica cuantas relaciones logra alinear la técnica seleccionada en esa corrida. Es importante al momento de evaluar, ya que indica la cantidad de relaciones que son equivalentes de una ontología en otra, del set de ontologías a alinear.
- #*Propiedades:* medida de rendimiento que indica cuantas propiedades de los conceptos logra alinear la técnica seleccionada en esa corrida. Al momento de evaluar, indica que algunos conceptos son similares en sus propiedades.

2.1.1.2 Mezcla de Ontologías

Es el proceso donde varias ontologías dentro de un mismo dominio se unen para estandarizar el conocimiento, hacer crecer el conocimiento, o tener el conocimiento total de manera local, entre otras cosas. Esto último es útil en los sistemas distribuidos, ya que es costoso estar saliendo de una red para consultar una ontología de otro sistema. Los mezcladores unen ontologías que manejan el mismo conocimiento, pero con diferente representaciones, o que poseen representaciones parciales de dicho conocimiento, donde las ontologías pueden coincidir en ciertos conceptos y en otros no, lo que requiere la presencia de un experto del conocimiento, el cual debe estar presente al momento de realizar la mezcla para tomar decisiones [4].

Para realizar el proceso de Mezclas de Ontología se debe buscar cada concepto de una ontología A en otra ontología B. Si el concepto es encontrado en B, se completa su definición agregando todas las propiedades y relaciones que tenga este concepto [1]. La unión debe ser cuidadosa, ya que se pueden presentar repeticiones, expresiones disjuntas,

distinto nivel de detalle, y contradicciones. La salida del proceso de mezcla de ontologías puede ser vista como la unión parcial de dos conjuntos. A este proceso, para efectos de este estudio, lo conoceremos como *mezcla débil*, ya que puede dejar conocimiento por fuera sin ser mezclado.

De manera general para Mezclar dos ontologías A y B, y almacenar el resultado en C, el algoritmo clásico es el siguiente:

Algoritmo General de Mezcla (Fuente [1])

0. INICIO
1. Copia ontología de A a C.
2. A partir de la raíz de C:
3. Alinear B y C
4. Busca en B cada concepto de C el cual sea más similar (Concepto Mas Similar, CMS).
5. Si hay un CMS en B para un concepto dado Cc de C, se seleccionan las relaciones del CMS en B que se pueden añadir a Cc, así como los nuevos conceptos que se encuentran en estas relaciones, de la siguiente manera :
    - 5.1. Se añaden las relaciones nuevas del CMS en B a Cc.
        - 5.1.1. Se copian los conceptos que no tenga C que se encuentren en las nuevas relaciones.
    - 5.2. Se detectan inconsistencias con un razonador (relaciones o propiedades que contradicen a otras relaciones o propiedades) entre las relaciones de Cc y las de CMS en B
    - 5.3. Si no se logra resolver las inconsistencias, permanecen en C las relaciones originales de A.
    - 5.4. Se copian los conceptos hijos de CMS en B a C.

6. Si no hay un CMS, se toma el siguiente concepto de C (Nuevo Cc) siguiendo un recorrido del grafo que representa la ontología del tipo primero en profundidad, y se vuelve al paso 4.
7. FIN

Al final de este macro algoritmo se puede detectar que hay relaciones y conceptos en B que no están siendo copiados en el paso 5 (tal como se muestra en la ontología C de la Fig. 2.1 en la cual se dejaron de copiar los conceptos a, b y c).

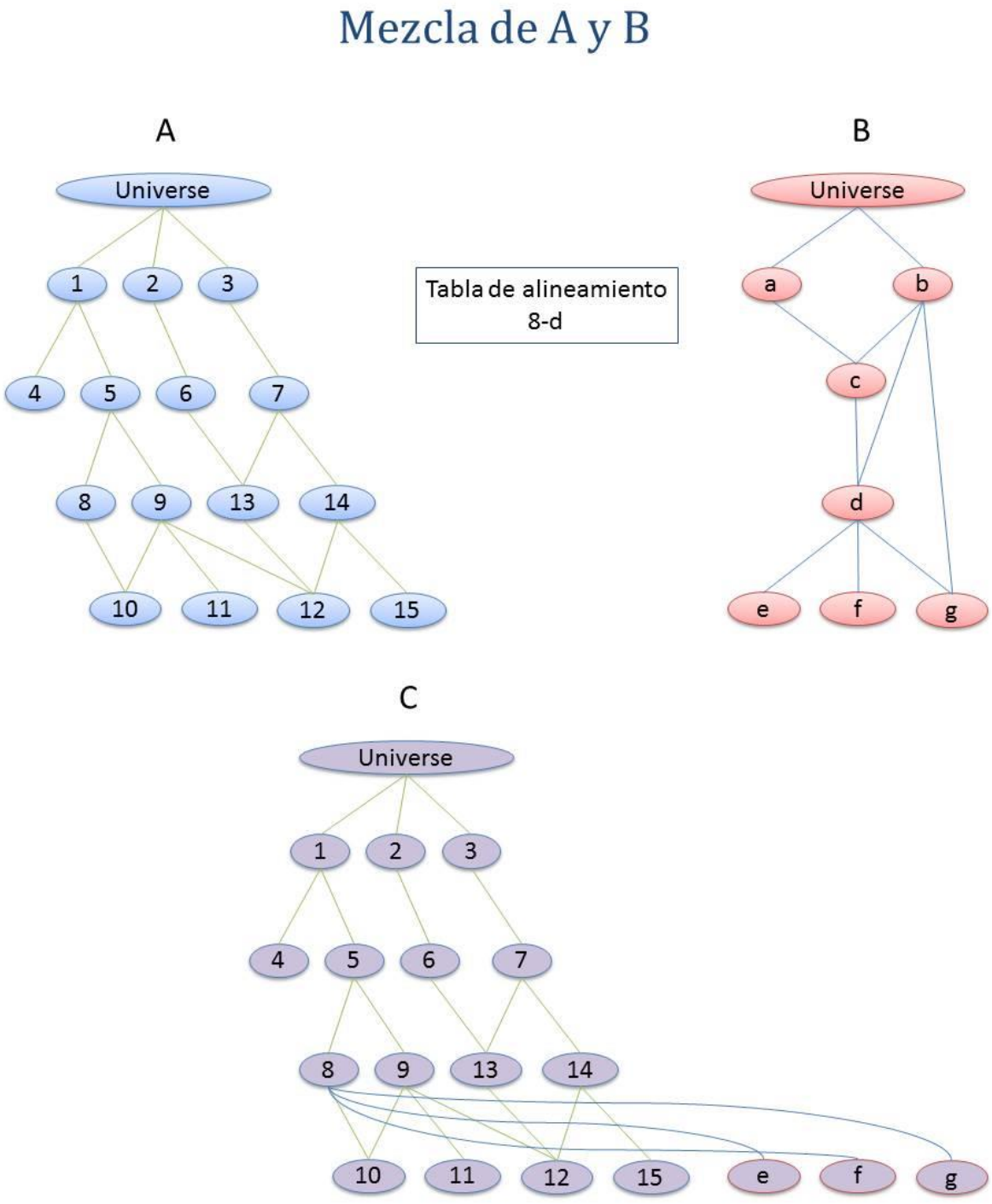

Fig. 2.1 Mezcla de A y B

De la Fig. 2.1 se puede observar, cómo este tipo de mezcla deja por fuera conceptos de B, tales como a, b y c, para efectos de este estudio, llamaremos esta mezcla, mezcla débil por esta razón antes mencionada.

Un algoritmo para hallar el CMS es el COM [1]. Dadas dos ontologías Oa y Ob, cuando se requiere encontrar el CMS de Ca de Oa en Ob se usan dos cadenas, el nombre de Ca y el nombre del predecesor de Ca. Esto conlleva a los siguientes casos, según [1]:

- Caso A: el concepto CA coincide con CB en B, y los predecesores PA y PB también.
- Caso B: PA coincide con PB, pero no hay coincidencia entre CA y CB.
- Caso C: CA coincide con CB, pero no hay ninguna coincidencia entre PA y PB.
- Caso D: CA no coincide con CB, y PA no coincide con PB.

2.1.1.2.1 Criterios de Rendimiento para la Mezcla de Ontologías

Algunos criterios de rendimiento que se usan para evaluar mezclas de ontologías son expuestos en [34]. Estas métricas permiten evaluar la calidad de la mezcla resultante. Ahora, es bueno resaltar que definir un conjunto de criterios de rendimiento para evaluarla la calidad de una mezcla de ontologías es una tarea difícil, ya que cada algoritmo de mezcla de ontologías que se ha propuesto busca solucionar problemas específicos (por ejemplo, para el problema de enriquecimiento y mezcla que estudian en [1], no importa dejar ciertos nodos por fuera de la mezcla), y no son desarrollados para resolver problemas de mezcla de manera genérica. Además, el proceso de mezcla tiene una fuerte dependencia con respecto a la calidad de las ontologías de entrada, si las mismas contienen errores de diseño o redundancia, esto se propagarán a la mezcla resultante.

Para evitar estos problemas, en [34] proponen el uso de los siguientes criterios para evaluar la calidad de la ontología resultante de un proceso de mezcla:

Cobertura: La cobertura de una mezcla es el grado de preservación de la información, y mide los conceptos de entrada (de las ontologías mezclándose) que se conservan en el resultado. Podemos diferenciar varios sub-casos, por ejemplo la cobertura con respecto a la ontología fuente o a la ontología objetivo (en [34] denominan a la ontología fuente a la ontología de la izquierda, y ontología objetivo a la ontología de la derecha). También podemos considerar la cobertura de las hojas, es decir, el grado en que se conservan los conceptos de las hojas de las ontologías de entradas. Ahora bien, la cobertura general se define como la media aritmética de la cobertura de las ontologías de origen y de destino [34]. Los valores de cobertura varían entre 0 y 1 (0-100%).

Compacidad: Esta medida comprueba el tamaño del resultado de la mezcla generada. El tamaño del resultado de la mezcla se puede aproximar como la suma del número de conceptos de ambas ontologías de entrada, menos el número de conceptos alineados (es decir, el número de correspondencias de igualdad). Por definición, el tamaño relativo del resultado de la mezcla completa debería ser máximo 1. Valores menores que 1 pueden indicar una compacidad mejorada, por ejemplo, si se dejan de copiar algunos conceptos de entrada para evitar la redundancia en el resultado de la fusión. Los valores mayores que 1 también son posibles, cuando la ontología resultante contiene conceptos recientemente introducidos, por ejemplo, agregados por un experto. Lo mejor es una compacidad mínima conjuntamente con una cobertura máxima. Para el caso en que los expertos añadan conceptos, esta medida relativa no aporta mucha información, pierde sentido, ya que lo ideal es calcularla con base a la mezcla completa que incluye todos los conceptos de ambas ontologías originales, para medir de alguna manera la redundancia. Los expertos siempre podrán introducir conceptos nuevos, sin redundar, esto aumentaría esta medida de compacidad, pero porque posee conceptos nuevos, no por ser redundante.

Redundancia: es deseable reducir el grado de redundancia o solapamiento semántico en una ontología integrada, para mejorar la comprensibilidad. En [34] usan dos medidas para

evaluar el grado de redundancia semántica, una vinculada al número absoluto de caminos a las hojas. La otra se basa en la siguiente ecuación: sea LP_S y LP_T el número de rutas a las hojas en las ontologías origen y destino de entrada, respectivamente, y M_L el número de conceptos hojas alineados. Entonces,

$$LP_B = LP_S + LP_T - M_L$$

LP_B es el menor número posible de caminos a las hojas que no introducen ninguna redundancia. En [34], usando ese valor de LP_B, definen la redundancia relativa de una ontología que resulta de una mezcla, como la relación entre el número de caminos a las hojas en la ontología resultante de la mezcla y LP_B. Los valores mayores que 1 indican la introducción de rutas redundantes para conceptos mezclados. Esto se considera nocivo para la comprensibilidad del resultado de la mezcla. Un valor de 1 es típicamente óptimo, ya que indica evitar con éxito rutas redundantes. Valores menores que 1 implican que algunos conceptos en las hojas no están cubiertos en el resultado de la fusión, lo que referencia a una pérdida de información y a la reducción de la cobertura (en este caso, de las hojas).

2.1.1.3 Enlazado de Ontologías

Es el proceso de encontrar relaciones entre entidades que pertenecen a diferentes ontologías, para crear una conexión entre estas ontologías sin necesidad de mezclarlas. Esto puede realizarse con la creación de una ontología intermedia que permite la navegación entre las ontologías que se están enlazando, o identificando las entidades comunes en ellas que sirvan de enlace. En general, la alineación de ontologías permite crear los enlaces directos usando las equivalencias entre los conceptos y propiedades de ontologías distintas encontradas, a este proceso lo conoceremos como *enlazado débil* dentro de esta investigación, ya que no crea conceptos nuevos. Los resultados pueden ser usados para visualizar correspondencias, transformar una fuente en otra, crear un conjunto de relaciones

o reglas entre las ontologías, o generar consultas para extraer información desde las dos ontologías [21].

Al igual que el proceso de mezcla, en este proceso se busca cada concepto de una ontología A en otra ontología B. Si el concepto es encontrado en B, se crea un enlace entre los dos conceptos, las propiedades y relaciones que tenga cada concepto permanecen intactas. De no encontrar conceptos semejantes no se crea nada. El proceso de enlazado de ontologías puede verse como un proceso de intersección de conjuntos. El macroalgoritmo de esta técnica es:

Algoritmo General de Enlazado (Fuente [5])

0. INICIO
1. Alinear A, B, y almacenar los resultados de similitud en una tabla
2. Definir el porcentaje de similitud mínimo.
3. Reducir la tabla del paso 2 a las relaciones que superen el mínimo
4. Si la tabla obtenida en 3 no es vacía:
    a. Crear una ontología C vacía.
    b. Crear en C todos los conceptos que se encuentren en la tabla del paso 3
    c. Agregar las relaciones de equivalencia "isEquivalent" de los pares de conceptos alineados de la tabla obtenida en 3.
5. FIN

El resultado del enlazado de ontologías es ilustrado en la Fig. 2.2, donde se enlaza A y B, y el resultado del enlazado débil es la ontología D, donde se puede observar una relación entre el concepto 8 y el concepto d, la cual es de equivalencia en OWL "isEquivalent". Por otro lado, todo el proceso de enlazado fuerte se puede realizar de manera manual, creando

la ontología de intersección entre las ontologías a enlazar con la ayuda de un experto del dominio bajo estudio, a este proceso lo conoceremos como *enlazado fuerte*, ya que se crean conceptos y relaciones nuevas. En este caso, un concepto nuevo propio de C es creado por un experto, tal como se muestra con los conceptos ′#′ y ′α′en Fig. 2.2 de la ontología C.

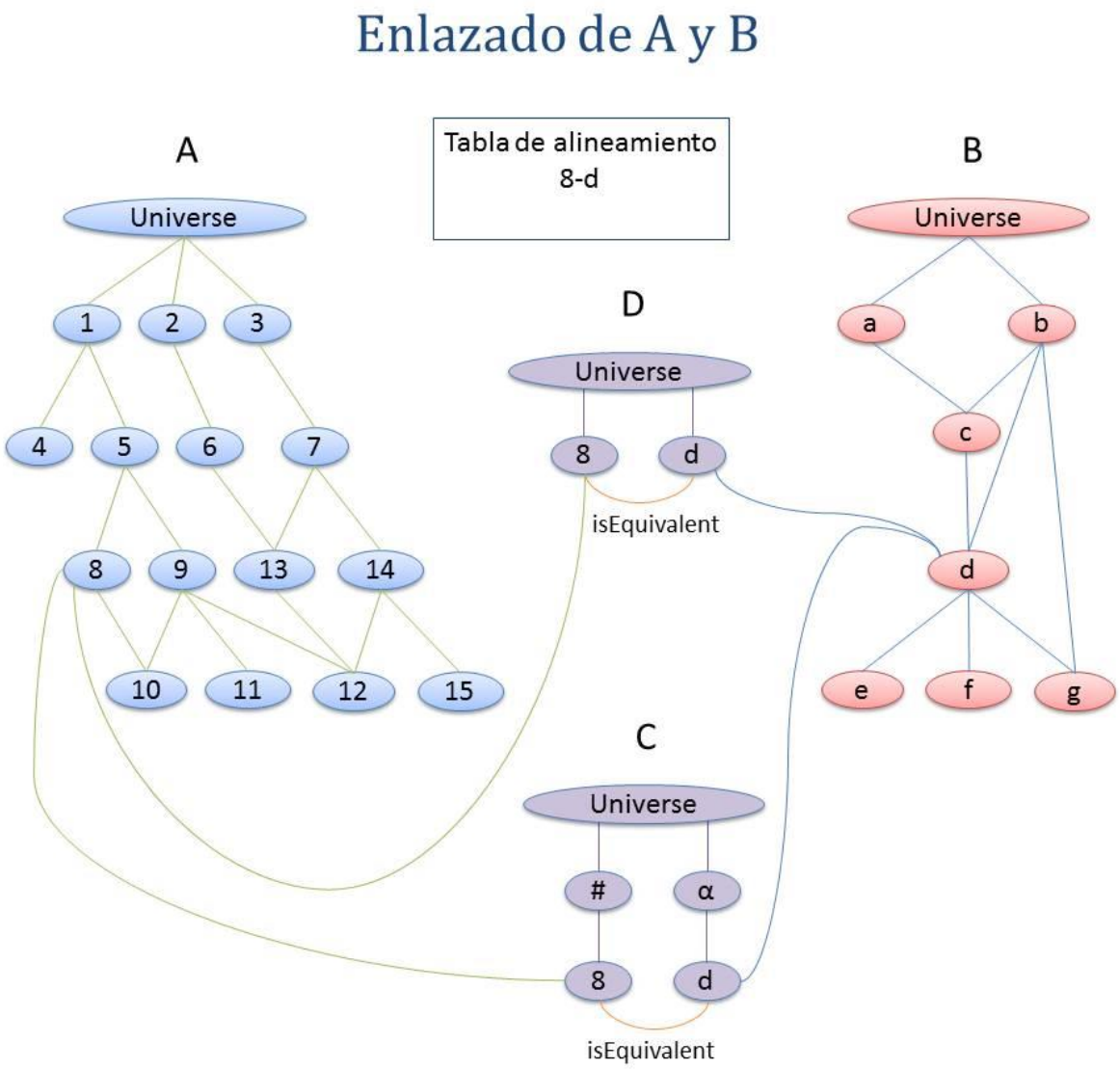

Fig. 2.2 Enlazado de A y B

2.1.1.3.1 Criterios de Rendimiento para el Enlazado de Ontologías

Los criterios para evaluar un enlace entre ontologías, consiste en evaluar la cobertura de las propiedades, conceptos y relaciones proveniente de las ontologías de entrada en la ontología enlace. Se puede realizar manual o automáticamente. Al igual que el alineamiento, se puede realizar de manera manual mediante un experto. Para ello, herramientas gráficas que permitan a los usuarios navegar de forma rápida en las ontologías son invaluables. La evaluación automática debe ser cuantitativa, se puede realizar mediante el uso de técnicas para la evaluación de alineaciones como la completitud, presentadas en [28], para la evaluación de alineamientos.

Algunos criterios de rendimiento de completitud, que se pueden usar son:

- *%Conceptos de cada ontología*: medida de rendimiento que indica el porcentaje de cuantos conceptos se incorporan desde la ontología enlace, para cada ontología; indica la completitud de conceptos que abarca el enlace de los conceptos de las ontologías de entrada.
- *%Relaciones*: medida de rendimiento que indica el porcentaje de cuantas relaciones se incorporan/cubren en la ontología enlace, de todas las ontologías de entrada.

2.2 Middleware Reflexivo y Computación Autonómica

El término *middleware* se refiere a una plataforma distribuida de interfaces y servicios que se encuentran entre la aplicación y el sistema operativo, tienen como objetivo facilitar el desarrollo, despliegue y gestión de aplicaciones distribuidas. La principal función de estas plataformas es enmascarar la heterogeneidad inherente en los sistemas distribuidos, además

de proporcionar un conjunto estándar de interfaces y servicios distribuidos entre aplicaciones.

La reflexión se refiere a la capacidad de un sistema para razonar y actuar sobre sí mismo. La reflexión computacional fue utilizado por primera vez por Pattie Maes, en su tesis de 1987 en la Vrije Universiteit Brussel, titulado: "Reflexión computacional"[26]. La reflexión permite un comportamiento dinámico y adaptativo, es decir, puede evolucionar. En general, los middleware reflexivos poseen dos grandes procesos:

- Introspección: capacidad de observar y razonar sobre su estado de ejecución.
- Intersección: capacidad de modificar su propio estado de ejecución, o alterar su propia interpretación o significado.

A demás, una arquitectura reflexiva posee los siguientes niveles:

- Nivel Base: en este nivel base se describen los cómputos o procesos que el sistema debe realizar, y contiene todo lo necesario para resolver un problema dado. En una arquitectura SOA se encuentra la composición de los servicios, las interacciones entre ellos, y el conjunto de reglas que rigen esas interacciones.
- Nivel Meta: reflexiona sobre cómo se realizan los cómputos o procedimientos en el nivel base, y verifica que su funcionamiento sea el esperado o requerido [29]. Este nivel está formado por objetos que desarrollan computación sobre la aplicación, para realizar las tareas de Introspección e Intercesión (parte reflexiva).

Por otro lado, la computación autonómica se refiere a un modelo de autogestión inspirado en el sistema nervioso del ser humano. El mismo incorpora sensores y actuadores que permiten observar el ambiente y actuar en consecuencia. La computación autonómica define una arquitectura compuesta de 6 niveles (ver Fig. 2.3):

Fig. 2.3 Arquitectura de la computación autonómica

- Recurso Gestionado: puede ser cualquier tipo de recurso (hardware o software) que puede ser gestionado. El recurso gestionado se controla a través de sus sensores y actuadores.
- Punto de Enlace: Enlaza a los sensores y/o actuadores requeridos para la gestión de los recursos.
- Gestor Autonómico: Implementa los lazos de control inteligentes que automatizan las tareas de autorregulación/autogestión de las aplicaciones.
- Orquestador de Gestores Autonómicos: Debido a que un sistema autonómico, puede contar con varios gestores autonómicos que necesitan trabajar en conjunto, para garantizar el funcionamiento correcto de los recursos. Este nivel proporciona el canal de comunicación para la coordinación entre ellos.
- Manejador Manual: Permite a los humanos configurar los gestores autonómicos para realizar su tarea de autogestión, proveyendo para esto una interfaz hombre-máquina que permite conectar al hombre con el gestor autonómico.
- Fuentes de Conocimiento: Proporciona acceso a los conocimientos requeridos para la gestión autonómica del sistema.

El Gestor Autonómico es el elemento encargado de realizar las tareas de auto-regulación de la aplicación por medido de un ciclo inteligente conocido como ciclo MAPE [29]. Como se observa en la Fig. 2.4, el ciclo MAPE de un Gestor autonómico está formado por cuatro fases o componentes:

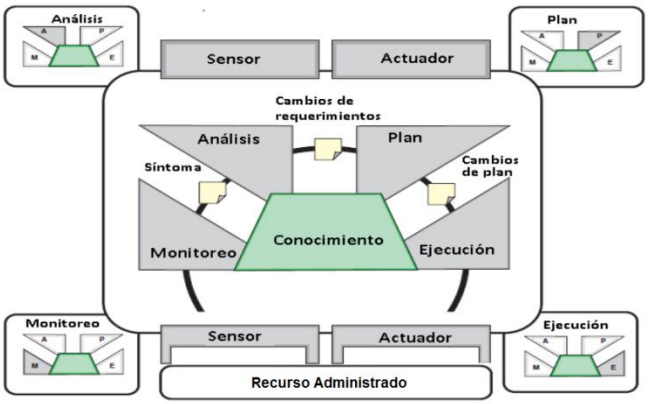

Fig. 2.4 Arquitectura MAPE.

Monitoreo: es el componente de supervisión del ciclo MAPE. La supervisión implica capturar propiedades del entorno (ya sea físico o virtual) que son de importancia para las propiedades autonómicas del sistema. Los componentes de software o hardware que se utilizan para realizar el monitoreo se llaman sensores [30]. Por ejemplo, la latencia de la red y el ancho de banda pueden medir el desempeño de los servidores web, mientras que la indexación de bases de datos y la optimización de consultas afectan el tiempo de respuesta de un DB, que pueden ser monitoreadas. El agente autonómico para monitoreo requiere datos para reconocer una degradación en el rendimiento. Los tipos de propiedades a supervisar y los sensores utilizados suelen ser específicos a cada aplicación.

Analizador: hace el diagnóstico de los estados de los componentes monitoreados. Dado una secuencia de eventos recibidos, el analizador detecta un estado operacional de interés (por ejemplo, una posible falla en el funcionamiento), y al ser detectado, envía al agente planificador un mensaje con el evento detectado.

Planificación: toma en cuenta los eventos detectados por el analizador para planificar una serie de cambios a efectuar en el elemento gestionado. En el caso más simple, se puede definir como un sistema de reglas evento-condición-acción (ECA), reglas que producen directamente los planes de adaptación según los eventos específicos recibidos del analizador

[30]. El componente planificador autonómico no almacena ninguna información sobre el estado del elemento gestionado, y se basa únicamente en los datos que son recibidos del analizador.

Ejecutor: es el componente encargado de realizar el plan. El ejecutor sólo recibe un mensaje del componente planificador, dicho mensaje contiene todas las acciones a realizar, en una secuencia específica, por ejemplo en arquitecturas SOA, estas acciones vienen dadas por una coreografía u orquestación.

2.2.1. Arquitectura Orientada en Servicios (SOA)

SOA (*Service Oriented Architecture*) es un modelo de desarrollo de software en el que una aplicación se divide en pequeñas unidades, llamadas servicios lógicas o funcionales. Permite el despliegue de aplicaciones distribuidas muy flexibles, con bajo acoplamiento entre componentes de software, los cuales operan en entornos distribuidos heterogéneos. Ejemplos de componentes de software que pueden ser integrados son los servicios web. En general, los servicios web son entidades computacionales que son autónomas e independientes de la plataforma, y que pueden estar compuestas por otras [31].

Servicios: cualquier función discreta que se puede ofrecer a un consumidor externo. Esta función puede ser una función de negocio individual, o un conjunto de funciones que forman un proceso [33]. Hay muchos aspectos adicionales a un servicio que también hay que tener en cuenta en su definición en SOA, tales como:

- Encapsular las funciones de negocio reutilizables
- Definir interfaces independientes de la implementación
- Usar protocolos de comunicación que hacen transparente la ubicación

Un sistema SOA proporciona una manera de integrar los servicios heterogéneos para una aplicación [31]. Para ello, en SOA se proponen los siguientes componentes para toda plataforma que soporte estos tipos de aplicaciones (los dos primeros están presentes solo en las aplicaciones SOA basadas en servicios web):

Lenguaje de Descripción de Coreografía de Servicios Web (WS-CDL): define el modelo para la descripción de la Coreografía de Servicios Web. Se centra en la colaboración de igual a igual entre servicios.

Marcado semántico para Servicios Web (OWL-S): define una ontología para describir servicios Web semántico [32], lo que permite automatizar las tareas de descubrimiento, invocación, composición, y seguimiento de los servicios web.

XML: Extensible Markup Language (XML) es el lenguaje de marcado que subyace en la mayoría de las especificaciones utilizadas para los servicios Web. XML es un lenguaje genérico que puede describir cualquier tipo de contenido de forma estructurada.

SOAP: Simple Object Access Protocol (SOAP), es una red con su lenguaje de comunicación, independiente de la plataforma, que permite a un cliente llamar a un servicio remoto.

WSDL: Web Services Description Language (WSDL) es una interfaz. El proveedor de servicios utiliza un documento WSDL con el fin de especificar las operaciones que ofrece su servicio Web, y los parámetros y tipos de datos de esas operaciones. Un documento WSDL también contiene la información de acceso al servicio.

WSIL: Servicios Web de idiomas de Inspección (WSIL) es una especificación basada en XML que localiza servicios Web sin utilizar UDDI. Sin embargo, WSIL también se puede utilizar con UDDI.

UDDI: del inglés *Universal Description, Discovery and Integration*, es a la vez un API de cliente y una aplicación de servidor basado en SOAP, que se puede utilizar para almacenar y recuperar información acerca de los proveedores de servicios Web.

En general, los servicios son inherentemente dinámicos y no pueden ser asumidos como siempre estables, debido a que durante la evolución natural de un servicio puede sufrir alteraciones (cambios en sus interfaces, un mal funcionamiento, entre otros) [22].

2.2.1.1 Buses de servicio

El bus de servicios empresariales (ESB, por sus siglas en inglés) es una infraestructura de middleware que apoya la implementación de SOA dentro de una empresa [33]. ESB es compatible con los conceptos de implementación de SOA a través de:

- Desacoplamiento de la vista del consumidor de un servicio, de la implementación de este servicio.
- Desacoplamiento de aspectos técnicos de las interacciones entre servicios.
- Integración y gestión de los servicios.

Los ESB se pueden utilizar para realizar algunas de las siguientes funciones de middleware:

- Mapear solicitudes de servicio y direccionarlas
- Transformar formatos de datos
- Apoyar los modelos de seguridad y de transacciones entre consumidores y proveedores de servicios
- Proveer protocolos de comunicación entre múltiples plataformas
- Proporcionar funciones de mensajería de correlación, publicación, eventos asíncronos y de petición/respuesta.

Varios conceptos son usados en los EBS:

BPEL: (Business Process Execution Language) es un lenguaje basado en XML que permite a los servicios Web en una arquitectura orientada a servicios (SOA), interconectarse y compartir datos. Al respecto, se ha creado el estándar BPEL4WS (Lenguaje de Ejecución de Procesos de Negocio para Servicios Web), que consisten en funciones que se definen a través de interfaces WSDL. En la Fig. 2.5 se puede observar una interacción de varios procesos, y ESB encapsulando la comunicación entre ellos.

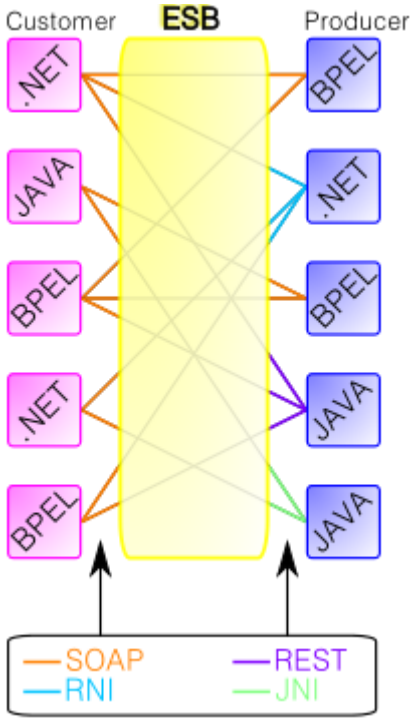

Figura 2.5 ESB. Imagen extraída de http://en.wikipedia.org/wiki/Enterprise_service_bus

2.3 Colonias de Abejas Artificiales (ABC)

La Colonia Artificial de Abejas (ABC) es uno de los algoritmos de optimización más recientemente definidos [27], motivado por el comportamiento inteligente de las abejas. Es

tan simple como la Optimización de partículas de Enjambre (PSO) y los algoritmos de Evolución Diferencial (DE [27].

El algoritmo ABC puede ser usado para resolver problemas de optimización multidimensionales y multimodales (un modelo multimodal es aquel que posee múltiples puntos críticos, los cuales pueden ser máximos o mínimos). En el modelo, la colonia de abejas artificiales consiste en tres grupos de abejas: abejas recolectoras, en espera y exploradoras. La primera mitad de la colonia se compone de las abejas artificiales recolectoras y la segunda mitad incluye las en espera y exploradoras. Para cada fuente de alimento, sólo hay una abeja empleada. En otras palabras, el número de abejas recolectoras es igual al número de fuentes de alimentos alrededor de la colmena a explotar. La abeja empleada cuya fuente de alimento se haya agotado se convierte en exploradora [27].

ABC tiene como parámetros de control el tamaño de la colonia y el número máximo de ciclo. Como herramienta de optimización, proporciona un procedimiento de búsqueda basado en una población, en la que las posibles soluciones representan potenciales fuentes de alimentos para las abejas artificiales. El objetivo de la colonia de abeja es descubrir los lugares de fuentes de alimentos con alta cantidad de néctar. En el sistema ABC las abejas artificiales vuelan alrededor en un espacio de búsqueda multidimensional. Las abejas recolectoras y en espera eligen las fuentes de alimentos en función de la experiencia de ellas y de sus compañeras de nido. Si la cantidad de néctar de una nueva fuente es mayor o mejor que el de la anterior en sus memorias, se memorizan la nueva posición y se olvidan de la anterior. Las abejas exploradoras vuelan y eligen las fuentes de alimentos de forma aleatoria, sin necesidad de utilizar la experiencia. Así, el sistema ABC combina métodos de búsqueda local, llevadas a cabo por las abejas recolectoras y en espera a través de la comunicación de las abejas recolectoras dando su experiencia a las en espera, con métodos de búsqueda globales gestionadas por las en espera y exploradoras en la aleatoriedad que se les da para

cambiar de fuente de alimento, esto intenta equilibrar el proceso de exploración y explotación [27].

A continuación los principales pasos del algoritmo:

Algoritmo General ABC (Fuente [27])

0. INICIO
1. Enviar las exploradoras a las fuentes de alimentos iniciales.
2. REPETIR
    - 2.1. Enviar las abejas recolectoras a las fuentes de alimentos encontradas por las exploradoras, según una probabilidad de conservar su opinión actual (en un principio no poseen fuente actual, así que van donde les indiquen las exploradoras que es mejor), basada en las cantidades de néctar encontrada y en la posición actual.
    - 2.2. Las recolectoras comparten el valor de la calidad del néctar a las abejas en espera.
    - 2.3. Enviar las abejas en espera a las fuentes de alimentos según sus calidades del néctar.
    - 2.4. Detener el proceso de explotación de las fuentes agotados por las abejas
    - 2.5. Enviar a los exploradores al área de búsqueda al azar para descubrir nuevas fuentes de alimentos.
    - 2.6. Memorizar la mejor fuente de alimento encontrado hasta ahora
3. HASTA (se cumplen los requisitos de parada)
4. FIN

Capítulo 3.

Algoritmos de Minería Ontológica

Este capítulo está dedicado a presentar los dos algoritmos de minería ontológica desarrollados en este trabajo. Al principio se presenta un algoritmo de recomendaciones de técnicas de alineación de ontologías bio-inspirado, y después se presenta un algoritmo de mezcla fuerte.

3.1. Sistema de recomendación de Alineamiento de Ontologías usando ABC

Como ya se ha explicado en el marco teórico, a raíz del auge de las ontologías, surge la necesidad de la integración de las mismas, para lo cual se han desarrollado varias técnicas de alineamiento de ontologías, entre las que se pueden mencionar: similitud estructural, distancia de nombres, distancia de nombre y propiedades, nombres idénticos.

En esta sección se propone un algoritmo basado en ABC (del inglés: *Artifitial Bee Colony*) que selecciona automáticamente la técnica de alineamiento adecuada en un momento dado, basado en las características de las ontologías que se quieren integrar. Este algoritmo es un primer paso hacia la automatización del proceso de integración de ontologías.

Al tener varias técnicas de alineamiento de ontologías, el problema consiste en decidir cuál de las técnicas usar. En particular, se definen los siguientes parámetros para el algoritmo ABC:

- S_i: Servicio que se puede realizar para solventar una actividad solicitada, en nuestro caso corresponde a una técnica de alineamiento, y es equivalente a una fuente de néctar en el algoritmo ABC.
- $G(S_i)$: Ganancia obtenida por el servicio S_i (técnica de alineamiento *i*, ver fórmula 3.1), y es equivalente a la calidad del néctar en el caso del algoritmo ABC.
- $Sa(S_i)$: Número de nodos alineados por la técnica de alineación S_i. Es usado para calcular $G(S_i)$ que es la calidad del néctar.
- $CA(S_i)$: Tiempo que tarda una abeja en ir al Servicio S_i y regresar con resultados (en nuestro caso, este es el tiempo de cálculo empleado por la técnica de alineamiento *i*, que repercute también con la calidad del néctar $G(S_i)$).
- Pc: Probabilidad de conservar la opinión. Valor pseudo-aleatorio, con una distribución normal. Este valor oscila dentro del rango de 0 y 1, el cual se multiplica por el valor de $S_a(S_i)$, para así incluir la aleatoriedad en $G(S_i)$.

La ganancia $G(S_i)$ es calculada de la siguiente manera:

$$G(S_i) = \frac{S_a(S_i)}{CA(S_i)} \times P_c \qquad (3.1)$$

La Fig. 3.1 describe el procedimiento general que se sigue. A groso modo, el mismo consiste en que los agentes abejas tienen la tarea de escoger una técnica de alineamiento (estas técnicas pueden variar, estar no disponibles o aparecer nuevas técnicas en los proveedores de alineamiento), las abejas recolectoras se dirigen cada una a una fuente de alimento, cada fuente de alimento es una técnica de alineamiento (hay tantas recolectoras como fuente de alimento o técnica de alineamiento), y cada abeja recolectora regresa con la información de la calidad del alineamiento, basado en el tiempo en realizarlo y la cantidad de nodos alineados. Esta calidad del alineamiento es la calidad del néctar, la cual es transmitida a las abejas en espera, estas abejas deciden a que técnica de alineamiento ir basándose en la

calidad que le transmiten las otras abejas y una probabilidad que hace el estocástico. En este caso, las abejas exploradoras solo se usan para determinar inicialmente donde están las fuentes de néctar que representan las técnicas de alineamiento (eventualmente se podrían volver a usar para notificar la agregación de una nueva técnica de alineamiento). Los resultados de la calidad de los alineamientos se van guardando en un vector, este vector es la memoria global y compartida de las abejas. Todo este proceso se repite hasta que se alcancen los objetivos o se llegue a una condición de parada, como por ejemplo el número máximo de iteraciones.

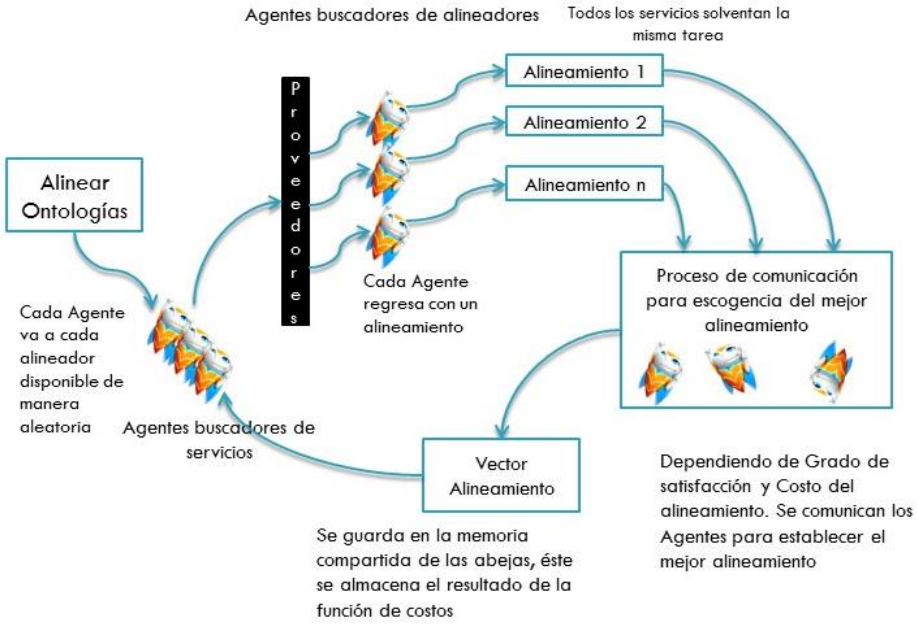

Fig. 3.1 Ciclo de las abejas para encontrar un buen alineamiento

A continuación se detallan los cambios hechos al algoritmo ABC:

Adaptación del algoritmo ABC

0. INICIO
1. Enviar las exploradoras para determinar dónde están las técnicas de alineamiento (fuentes de alimentos).
2. REPETIR
    2.7. Enviar las abejas recolectoras a las fuentes de alimentos encontradas por las exploradoras, y determinar la calidad del alineamiento de cada una de ellas usando la fórmula (3.1)
        a. Si las abejas recolectoras ya poseen una técnica de alineamiento de un paso anterior, pueden cambiar de opinión y no conservar la actual, para ello comparan su ganancia con la de una fuente de alimento vecina de la siguiente manera:

    Para una abeja x que sigue la fuente de alimentos i ($\boldsymbol{x = S_i}$),

    Si ($\boldsymbol{G(S_i) < G(S_j), tal\ que\ j\ es\ vecina\ a\ i}$ entonces:

    $$\boldsymbol{x = S_j}$$

    2.8. Las recolectoras comparten el valor de la calidad del alineamiento a las abejas en espera, y estas preferirán ir a las mejores técnicas de alineamiento divulgadas por las recolectoras.
    2.9. Detener el proceso de explotación de las fuentes abandonas por las abejas
    2.10. Enviar a los exploradores al área de búsqueda al azar para descubrir nuevas fuentes de alimentos,
    2.11. Memorizar la mejor fuente de alimento encontrada hasta ahora
3. HASTA (se cumplen los requisitos de parada)

4. FIN

Si la satisfacción $Sa(S_i)$ fuese la misma para todas las abejas, quedaría como parámetro de control el tiempo de ejecución de las técnicas, $CA(S_i)$.

El algoritmo es iterativo, lo que permite hacer varias sugerencias. Es decir, no es necesario que todas las abejas lleguen a un mismo servicio (fuente de néctar). Si es necesaria la selección de un solo servicio (técnica de alineamiento), y al final de las iteraciones no se llega a sugerir un solo proveedor del servicio (una sola técnica), se toma el servicio que tenga más abejas sugiriéndolo, cabe destacar que las fuentes de alimentos (técnicas de alineamiento), pueden pasar a no estar disponibles en cierto momento, o aparecer nuevas (técnicas de alineamiento que se vayan incorporando) a medida que pasa el tiempo.

3.2. Algoritmos de Mezcla de Ontologías

El problema de la mezcla tradicional de ontologías (que nosotros hemos llamado mezcla débil), es que deja conocimiento sin ser incorporado en la ontología resultante (tal como se mostró en la Fig. 2.1 del capítulo 2). A continuación se presenta un algoritmo de mezcla fuerte, que es un híbrido de una mezcla débil, y una extensión que busca mezclar el conocimiento que queda por fuera de la mezcla débil.

Como ya se ha venido describiendo, la mezcla débil es un enriquecimiento de una de las dos ontologías de entrada, con el conocimiento existente en la otra ontología de entrada, para mayor entendimiento ver el Capítulo 2. En la segunda ontología (a la cual se le está extrayendo el conocimiento para ser agregado a la primera), puede quedar conocimiento sin

ser agregado. Solo son copiados los conceptos alineados de la segunda ontología (ver figura 3.2), en el mejor de los casos con un buen alineamiento, todos los conceptos de la segunda ontología serán copiados.

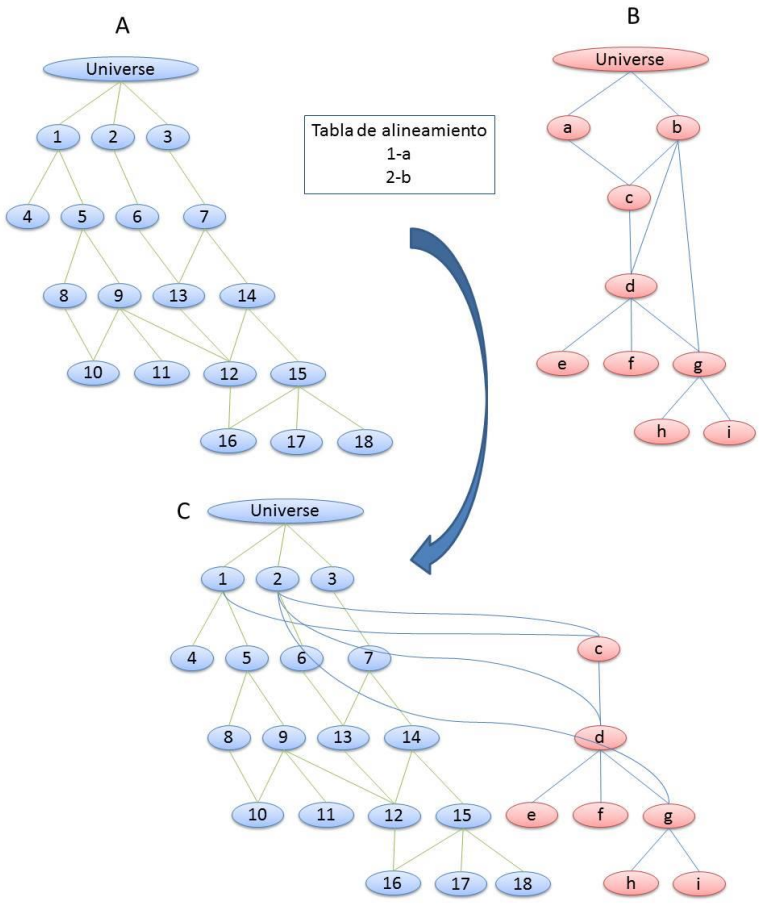

Fig. 3.2 Mezcla débil donde no es dejado conocimiento por fuera

Nuestra propuesta de Mezcla Fuerte se hace en dos partes, una primera parte donde se realiza la mezcla débil, y una segunda parte donde se incorporan los conceptos y relaciones dejadas por fuera.

Primera Parte: Nuestro sistema para realizar la mezcla débil de dos ontologías consistentes A, B en una ontología C es:

Requerimientos: una tabla de alineamiento T, y que A, B sean consistentes (que ninguna relación o propiedad contradiga a otra relación o propiedad).

Algoritmo Mezcla débil

0. INICIO
1. Copiar A en C
2. Para cada Concepto Más Similar (CMS) en B de la tabla de alineamiento hacer
    2.1. Recorrer en orden los hijos del CMS
    2.2. SI el hijo no está en la tabla de alineamiento
        2.2.1. Agregar el hijo al espacio
        2.2.2. Agregar la relación con su padre alineado
    2.3. SI NO 2.2 (Es decir que sí está en la tabla de alineamiento, y padre e hijo fueron alineados con sus equivalentes),
        2.3.1. Si la relación es diferente a las existentes
            2.3.1.1. Copiar la relación
    2.4. FIN SI 2.2.
3. FIN

Como ya se ha señalado, este algoritmo tiene una desventaja que se puede observar en las Fig. 3.3, donde un concepto queda por fuera ya que no todos los nodos de B fueron alineados.

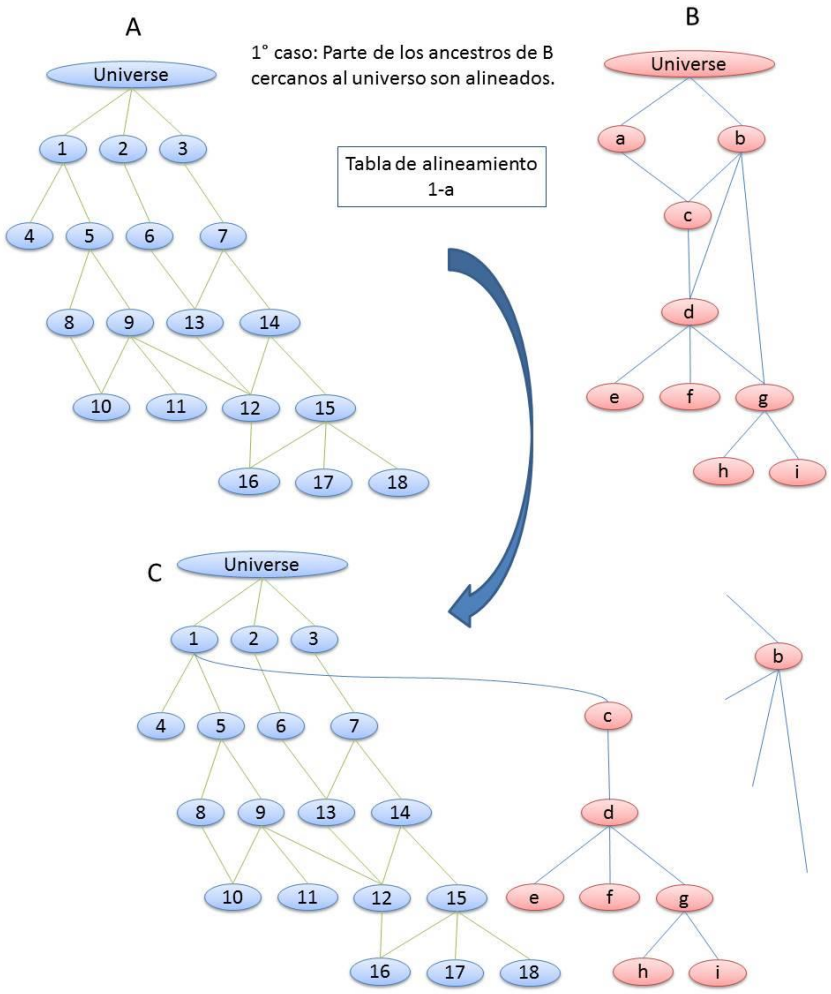

Fig. 3.3 Mezcla débil donde un concepto de B no es agregado a C

Segunda Parte: si de la ontología a la cual se le está extrayendo el conocimiento para ser agregado a la primera queda aún conocimiento sin ser agregado, se analizan los siguientes casos:

Caso 1: Se alinearon parcialmente los conceptos de B (ver Fig. 3.4), que hace que los nodos no alineados no se copien en la ontología resultado. En este caso, solo bastaría con agregar los nodos no alineados a C, y copiar las relaciones que no fueron copiadas o alineadas (ver Fig. 3.4).

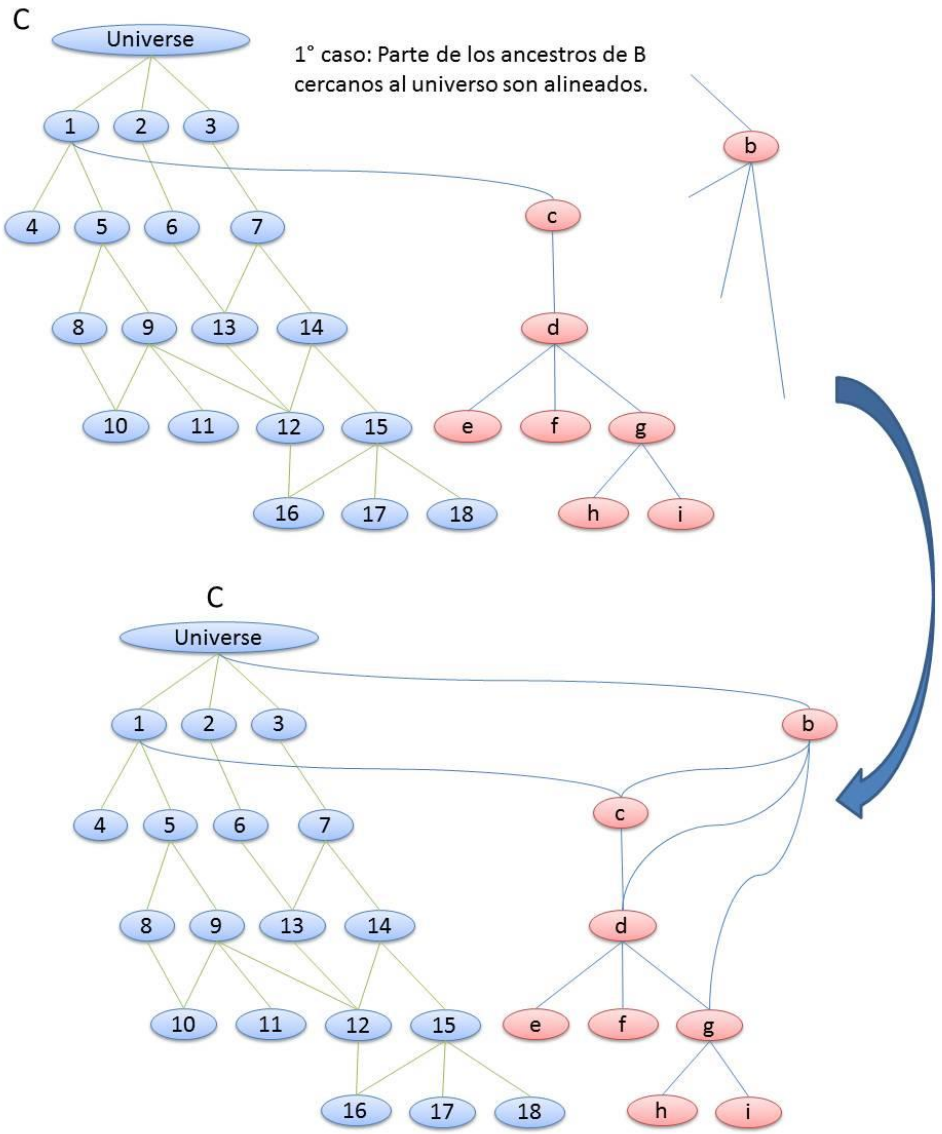

Fig. 3.4 Mezcla fuerte donde se incorpora un concepto de B que no fue agregado a C

Caso 2: Se alinearon ciertos nodos pero no se alinean sus hijos, esto deja a los ancestros sin ser copiados en la fase de mezcla débil (ver Fig. 3.5). La mezcla fuerte debe agregar estos conceptos al universo de C, y las relaciones donde ellos participan, como ocurre con b-g ,c-8 y b-8, tal como se observa en la Fig. 3.6.

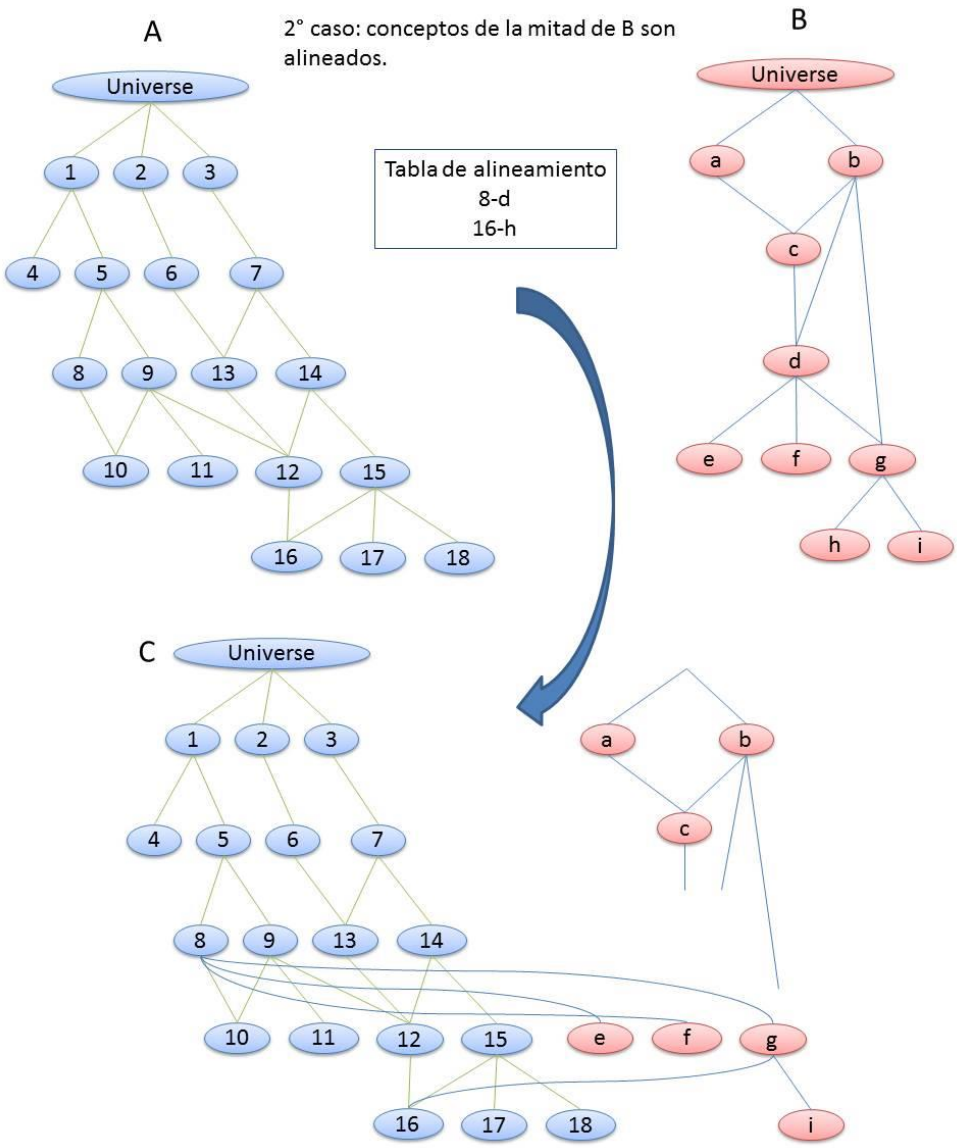

Fig. 3.5 Mezcla débil donde un concepto de B ancestro no es agregado a C

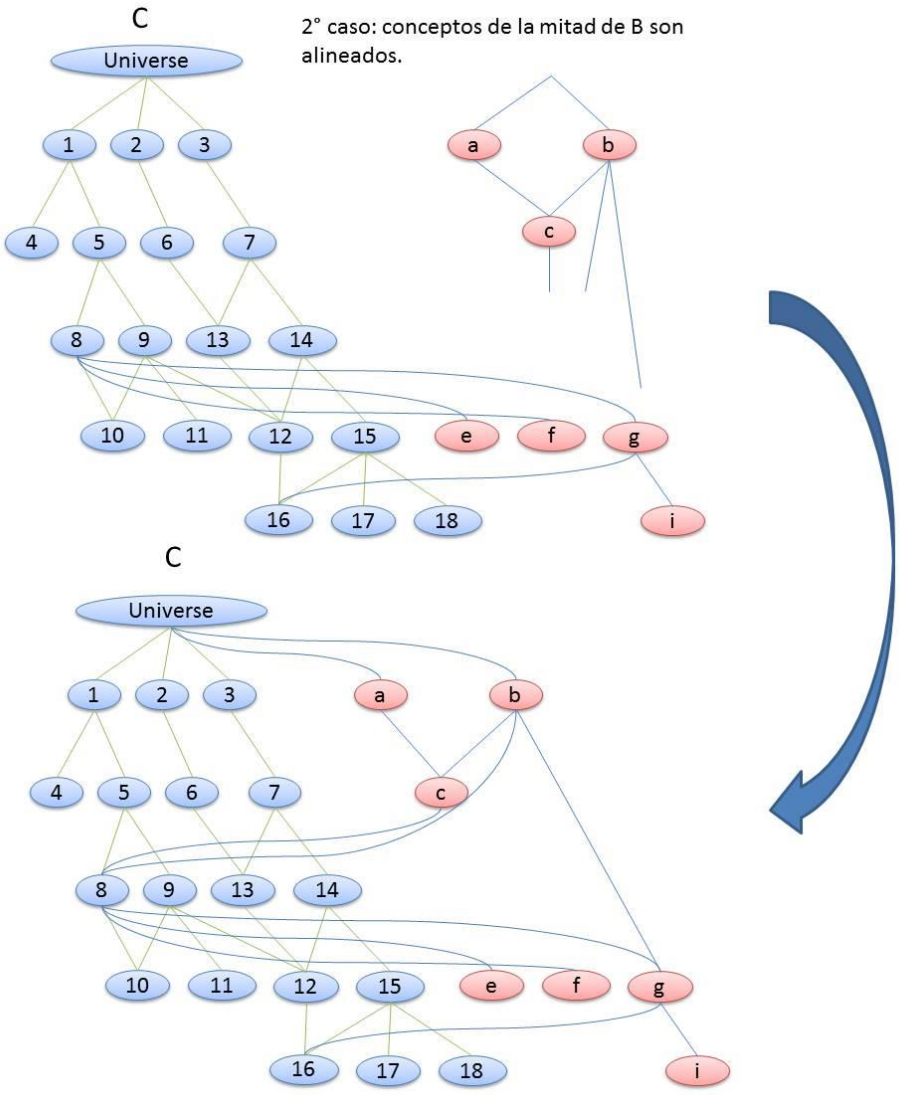

Fig. 3.6 Mezcla fuerte donde el concepto ancestro de B es agregado a C

Caso 3: Solo los nodos hojas se alinearon, copiándose sus propiedades, dejando por fuera de C un conjunto de conocimiento grande de B (ver Fig. 3.7). La mezcla fuerte debe agregar todos los conceptos no copiados a C, buscar las relaciones que estos conceptos ya tenían con otros en B, y copiar también esas relaciones con los conceptos que fueron copiados o con los que fueron alineados, tal como pasa con g-16 (ver Fig. 3.8).

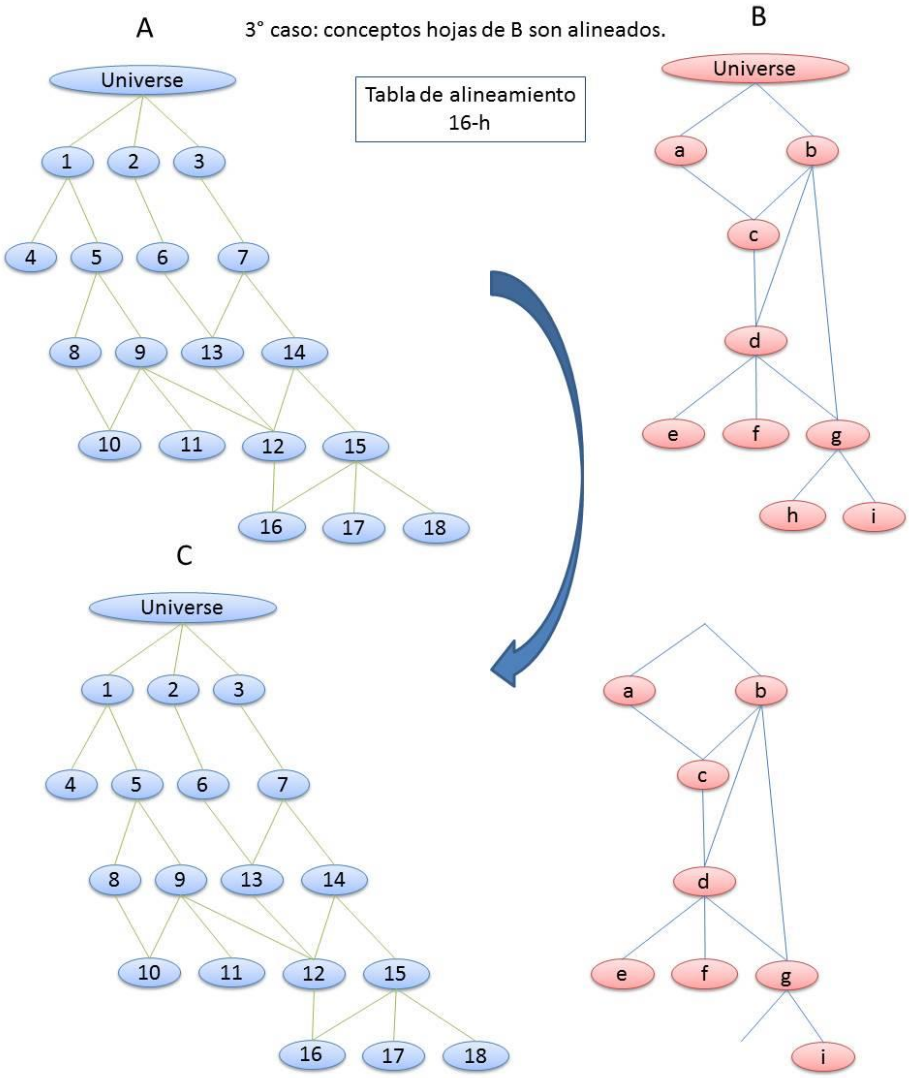

Fig. 3.7 Mezcla débil donde solo los nodos hojas son alineados

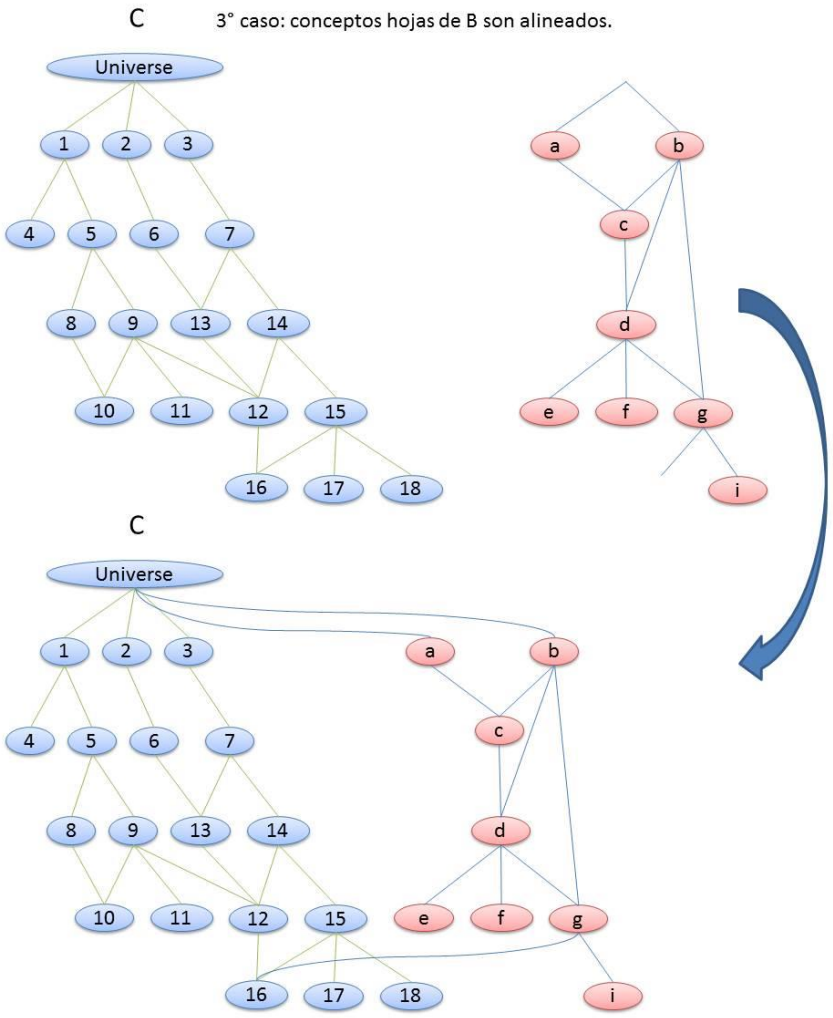

Fig. 3.8 Mezcla fuerte donde solo los nodos hojas son alineados

Estos son los tres casos básicos, ya que se toma en cuenta la alineación parcial de: los nodos ancestros al primer nivel en el caso 1, nodos en la mitad en el caso 2, y por último, nodos hojas en el caso 3. Si se realiza la alineación total de los ancestros del primer nivel, la mezcla débil cubre todo el conocimiento, en los otros dos casos si ocurre que se dejara algún conocimiento sin copiar. Como se puede notar la solución a los tres casos es la misma, ya que estos son muy similares y se logró con el algoritmo de mezcla fuerte, solucionar los tres casos.

Para realizar la mezcla fuerte de dos ontologías consistentes A, B en C

Algoritmo Mezcla Fuerte

0. INICIO
1. Realizar una mezcla débil
2. Detectar conceptos de B no copiados ni alineados
3. SI existe al menos un subárbol no copiado ni alineado

3.1. PARA CADA subárbol no alineado hacer (caso 1, 2, 3):

3.1.1. Buscar el ancestro (más cercano al universo) del sub árbol copiado

3.1.1.1. Copiar todo el subárbol al espacio

3.1.1.2. Crear la relación con el ancestro y el universo

3.1.1.3. Copiar las relaciones entre los conceptos del subárbol no copiada

3.1.1.4. PARA CADA nodo del sub árbol no copiado (Recorrer en orden) (caso 2, 3)

3.1.1.4.1. PARA CADA hijo del nodo en B

3.1.1.4.1.1. SI está en la tabla de Alineamiento

3.1.1.4.1.1.1. Crear la relación del padre del subárbol no copiado al hijo CMS en la Ontología Resultado C

3.1.1.4.1.2. SI NO 3.1.1.4.1.1. Está en la tabla de

Alineamiento (no hacer nada, ya que la relación se copió en 3.1.1.3)

3.1.1.4.1.3. FIN SI 3.1.1.4.1.1.

4. FIN SI 3.
5. FIN

Como ya se ha mencionado, nuestro algoritmo de mezcla fuerte parte de una mezcla débil (paso 1), después verifica si hay nodos dejados por fuera de la mezcla, en caso de que esto se cumpla (paso 3 del algoritmo) empieza a recorrer estos nodos como sub árboles (paso 3.1). Cada ancestro lo agrega directamente al universo, y se trae todas sus relaciones. Después recorre de nuevo estos subárboles que ya ha incorporado al universo de C, y busca agregar las relaciones con los nodos que fueron alineados y copiados en la mezcla débil (esto se realiza en el paso 3.1.1.4.), copia todas las relaciones pertinentes.

Una mejora que se agregó al algoritmo anterior (dando como la versión final del servicio que se diseñó, al algoritmo presentado con esta modificación), consiste en un intento para acercar la ontología C al resultado de los expertos, esto es realizado en esta nueva versión, donde se toman los nodos dejados por fuera en la mezcla débil, a los que llamaremos *leftovers*, y se busca si hijos de estos nodos dejados fuera de la mezcla débil fueron copiados o existe su CMS en la otra ontología. La modificación del algoritmo se encuentra en los pasos 3.1.1.4.1.1.2. y 3.1.1.4.1.1.3., que se agregan:

Modificación algoritmo Mezcla Fuerte

3.1.1.4.1. PARA CADA hijo del nodo en B

3.1.1.4.1.1. SI está en la tabla de Alineamiento

3.1.1.4.1.1.1. Crear la relación del padre del subárbol no copiado de B al hijo alineado CMS en la Ontología Resultado C

3.1.1.4.1.1.2. Tomar el padre actual (originario de A) del hijo alineado CMS (originario de A) y ponerlo como padre nuevo del subárbol no copiado de B.

3.1.1.4.1.1.3. Quitar la relación del padre actual (originario de A) del hijo alineado CMS (originario de A) para evitar redundancia.

3.1.1.4.1.2. SI NO 3.1.1.4.1.1. Está en la tabla de Alineamiento (no hacer nada, ya que la relación se copió en 3.1.1.3)

3.1.1.4.1.3. FIN SI 3.1.1.4.1.1.

Esto es, al detectar estos nodos en la ontología C, se inserta como su padre el nodo dejado en el grupo de los *leftovers*, copiando todas las relaciones de este nuevo nodo en la ontología C, trayéndose todas las relaciones como en las otras versiones de la mezcla fuerte. Ahora, para no copiarlo directamente al universo, este se copia como hijo del padre actual

de su hijo recientemente copiado de la ontología C, y se elimina la relación padre-hijo de los conceptos que estaban en la ontología C.

Este proceso se describe de manera gráfica en las Figs. 3.9 y 3.10. En la Fig. 3.9 se realiza el proceso de mezcla débil donde solo se copian las relaciones de los nodos alineados. En la Fig. 3.10 se muestra el nuevo proceso de inserción de los *leftovers*.

Como es de esperar, mezclar A sobre B, o B sobre A, genera resultados diferentes, esta propuesta genera ambas mezclas, y da como resultado la ontología mezcla que posea mejores criterios de rendimiento (los cuales fueron expuestos en el capítulo 2). La mejor mezcla es aquella que no posea inconsistencias, tenga una completitud de todos los conceptos de A y B (objetivo de este estudio), sea menos redundante y más compacta.

La técnica de mezcla fuerte es interesante, ya que incorpora el conocimiento que no se estaba tomando en cuenta en otros mezcladores. Por otro lado, la nueva versión con la mejora, al introducir conceptos en el grafo de la ontología ya creado, y modificando su estructura, acerca los resultados al de los expertos, ya que no simplemente copia el conocimiento al universo de la mezcla, sin introducirlo en un lugar en específico, la ontología resultante C termina con una estructura muy semejante a la propuesta por los expertos, tal como se demostrará en el Capítulo 5 de resultados.

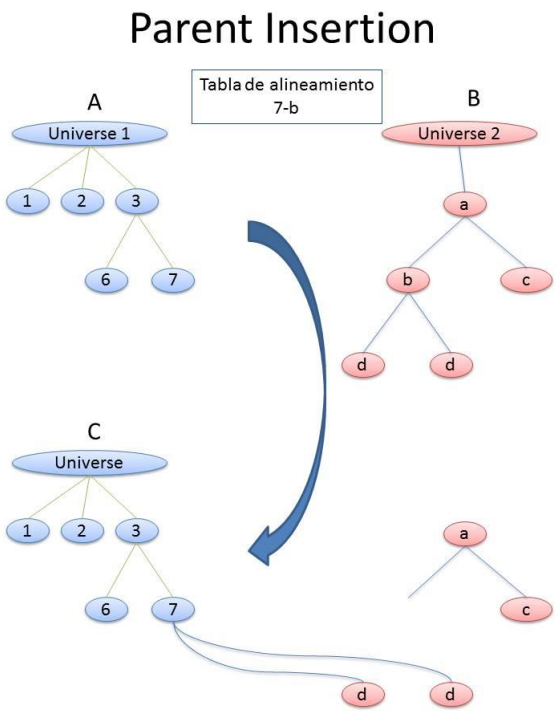

Fig. 3.9 Proceso mezcla débil tradicional, ejemplo mezcla fuerte

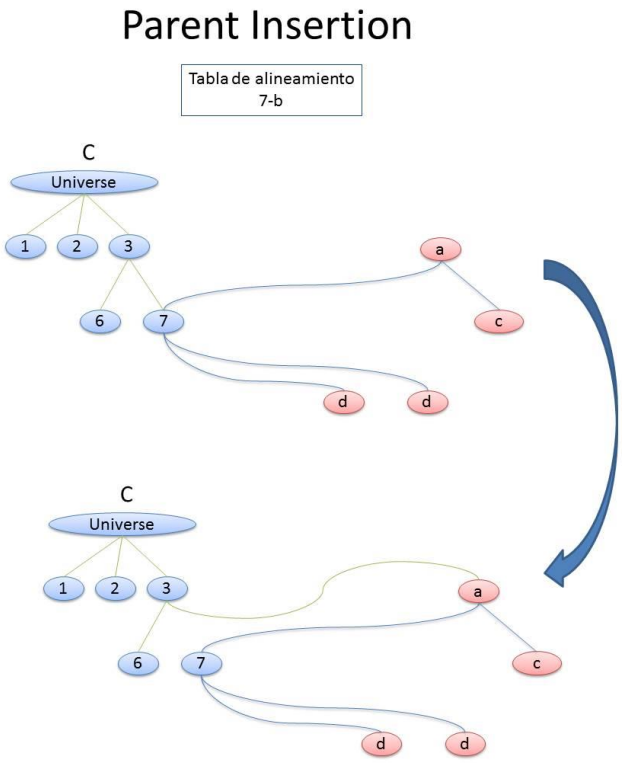

Fig. 3.10 Nuevo proceso mezcla fuerte

Capítulo 4

Diseño del Middleware orientado a la integración autonómica de ontologías

En este capítulo se presenta el diseño de un Middleware Reflexivo para la integración autonómica de ontologías. Esto es, un middleware que permite invocar una serie de servicios de minería ontológica, para realizar, entre otras cosas, alineaciones, mezcla débil o fuerte, enlazado fuerte o débil, verificación de la consistencia de ontologías, selección automática de técnica de alineación, entre otros.

4.1. Caracterización del Middleware

El objetivo del middleware es gestionar un conjunto de servicios para la integración de ontologías. Algunos de los posibles servicios son:

- Alinear ontologías
- Integrar ontologías, usando alguna de las siguientes técnicas:
    - Mezcla débil
    - Mezcla fuerte
    - Enlazado débil
    - Enlazado fuerte
- Escoger técnica de Alineamiento
- Validar la coherencias de las ontologías

- Extraer/Actualizar ontologías desde archivos no estructurados [42, 43]
- Extraer/Actualizar ontologías según el contexto [40, 41]

A continuación se describen los servicios vinculados a este trabajo, que prestará el middleware (de los cuales fueron implementados: Escoger técnica de Alineamiento y Mezcla fuerte):

Nombre	Alinear
Parámetros de entrada	Dos ontologías A y B, una técnica de similitud semántica.
Parámetros de salida	Una tabla de correspondencias
Descripción	Utiliza la técnica de similitud semántica para ir creando las correspondencias de equivalencias.

Nombre	Escoger técnica de Alineamiento
Parámetros de entrada	Dos ontologías A y B
Parámetros de salida	Una lista de las mejores técnicas para alinear A y B.
Descripción	Utiliza técnicas de optimización (como el algoritmo ABC) para la escogencia de las mejores técnicas para realizar el alineamiento de A y B.

Nombre	Mezcla débil
Parámetros de entrada	Dos ontologías A y B del mismo dominio y un alineamiento relativamente grande en relación a las ontologías de entrada
Parámetros de salida	Una ontología C con el conocimiento de una de las dos ontologías, enriquecido con el conocimiento de la otra
Descripción	Basado en [1], se toma una ontología A, se copia como resultado en C, y se va enriqueciendo con B, comparando todos los conceptos de la ontología C (que son los mismos de A en este momento) con los de la ontología B, enriqueciendo los conceptos de C con sus conceptos semejantes de B. Esto dejar por fuera parte del conocimiento de B.

Nombre	Mezcla fuerte
Parámetros de entrada	Dos ontologías del mismo dominio y un alineamiento relativamente pequeño en relación a las ontologías de entrada
Parámetros de salida	Una ontología C con el conocimiento completo de las dos ontologías
Descripción	Copia el conocimiento de la ontología de entrada que sea más completa en la ontología de salida C, se realiza una mezcla débil, y al final, se agrega a C el conocimiento que quedo por fuera de la mezcla débil.

Nombre	Enlazado débil
Parámetros de entrada	Dos ontologías A y B con dominios que no son idénticos y un alineamiento.
Parámetros de salida	Una ontología intermedia con conceptos intermedios de A y B
Descripción	Crea una ontología intermedia de A y B partiendo del alineamiento. Para eso, se sigue lo expuesto en [21], básicamente lo que se realiza es la intersección de las ontologías, a partir de la cual se podrían hacer inferencias específicas en cada ontología.

Nombre	Enlazado fuerte
Parámetros de entrada	Dos ontologías A y B con dominios que no son idénticos.
Parámetros de salida	Una ontología intermedia con conceptos intermedios de A y B
Descripción	Se crea una ontología intermedia de A y B. Como no existe alineamiento, la ayuda de un experto es necesaria para este servicio. Es realizado de manera semiautomático, con la ayuda de un experto del conocimiento global que se está enlazando, el cual puede definir nuevos conceptos, así como enlaces que relacionan conceptos de las ontologías distintas, creando así una Macro-Ontología (que posea conceptos globales para las dos ontologías de entrada, esta macro-ontología permite entre las ontologías enlazadas: la traducción, navegación y razonamiento) con partes de conocimiento de las ontologías enlazadas.

4.2. Arquitectura del Middleware

El diseño general del middleware para los servicios más pertinentes para este estudio se observa en la Fig. 4.1, el cual se puede extender con más servicios, como los de la lista presentada en la sección 4.1. En específico, todo Middleware Reflexivo consta al menos de dos niveles:

Nivel base: en nuestro caso, en este nivel se encuentran las ontologías a ser integradas, así como la ontología con el conocimiento integrado, y todas las aplicaciones que estén usando estas ontologías.

Nivel meta: en este caso, se encuentran los servicios que se van a ejecutar para integrar y mantener las ontologías. Como ya se ha mencionado, algunos de estos servicios son: Mezclar Ontologías (débil y fuerte), Enlazar Ontologías (débil y fuerte) y Alinear Ontologías (tradicional y emergente).

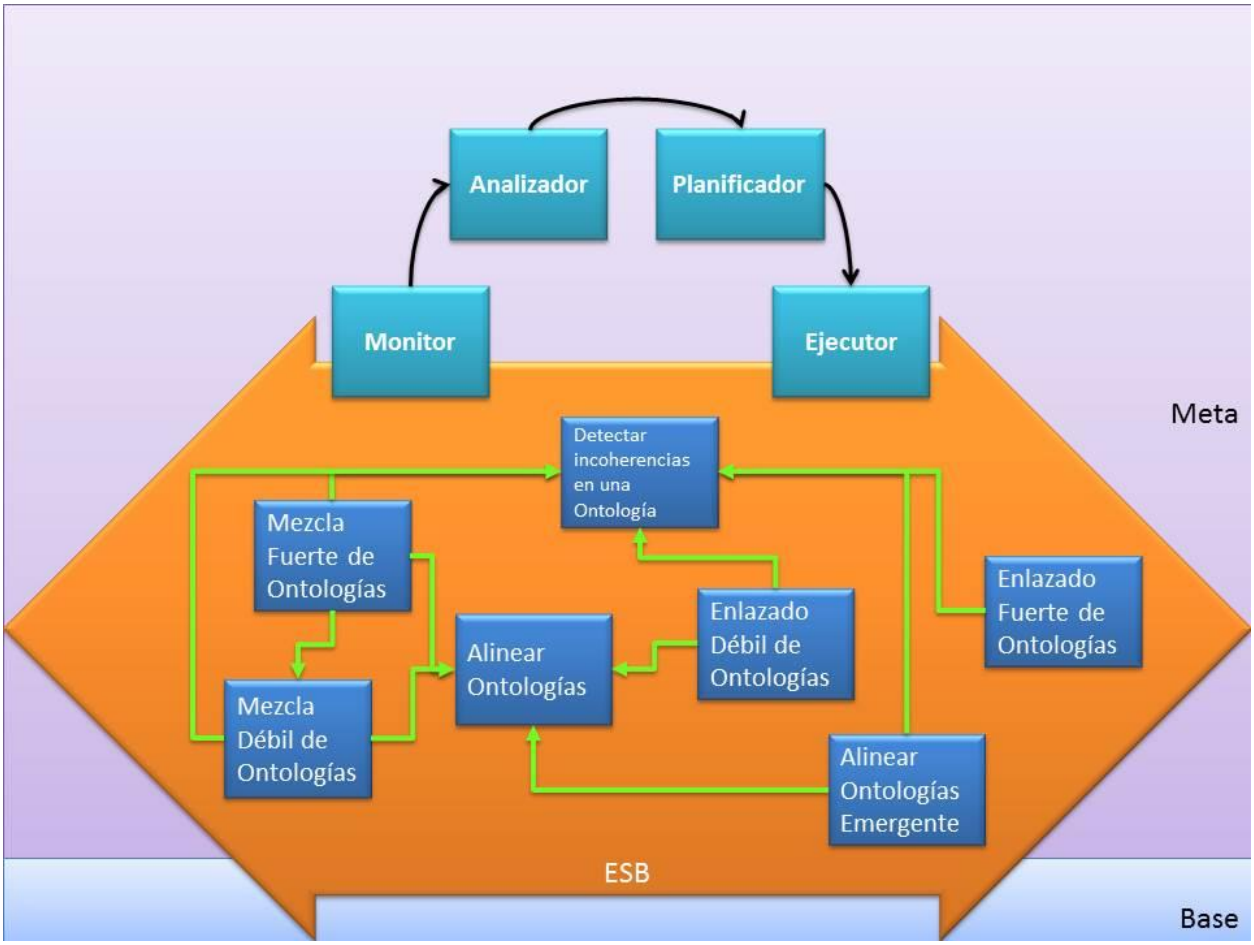

Fig. 4.1 Middleware reflexivo para la integración de ontologías

Ahora, nuestro Middleware, por ser autonómico se caracteriza por tener un ciclo MAPE (Monitoreo, Análisis, Planificación y Ejecución) (ver figura 4.1). En general, en la *fase de monitoreo* se observa el comportamiento de las ontologías que están en el nivel base, estas observaciones pasan al *analizador,* el cual realiza la reflexión y detecta qué realizar para mantener o realizar la integración. La operación detectada a realizar se envía al *planificador*, el cual es el encargado de especificar y ordenar, mediante una orquestación o coreografía, todos los servicios requeridos para mantener/integrar las ontologías, según lo detectado. Este plan es enviado al ejecutor, quien realiza el llamado a todos los servicios que se

encuentran en la coreografía u orquestación. En la figura 4.2 se observa en detalle el ciclo MAPE, que es descrito a continuación.

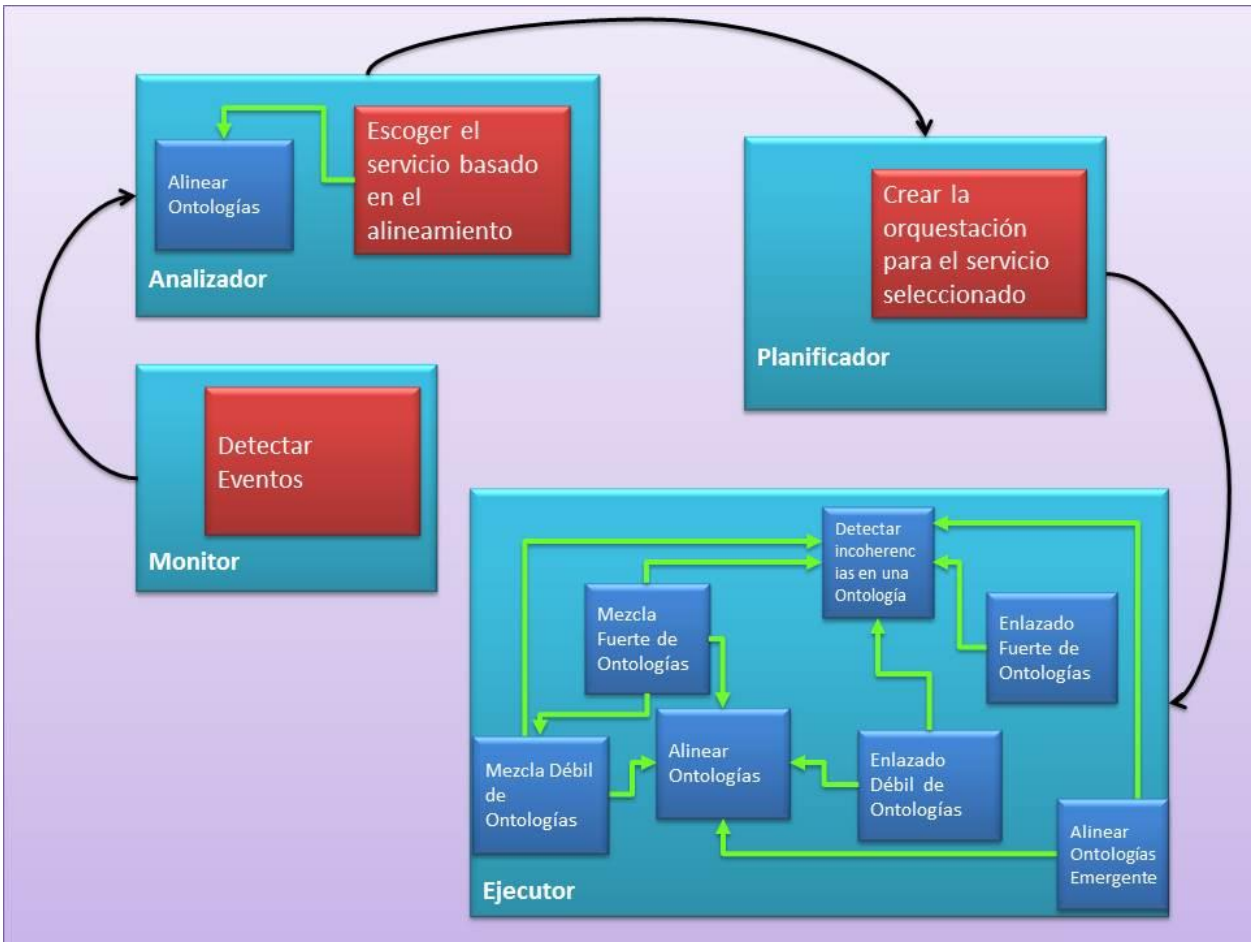

Fig. 4.2 Ciclo MAPE del Middleware reflexivo para la integración de ontologías

Monitor

En esta fase se monitorean las ontologías que están en el nivel base. Estas observaciones pueden generar principalmente dos casos, uno *basado en peticiones del usuario*, y el otro *basado en cambios que se detectan en las ontologías*. Los eventos a escuchar *en el caso basado en usuario* son las peticiones realizadas por este, dichas peticiones son invocaciones directas a alguno de los servicios para integrar ontologías. Por otro lado, los eventos *en el caso basado en*

el dominio ontológico son los cambios detectados en las ontologías en el nivel bases que ya fueron integradas (si no han sido integradas es el usuario quien decide cuando hacerlo). Es decir, los eventos son las modificaciones realizadas a alguna de las ontologías ya integrada, ya que eso implica que la ontología que maneja el conocimiento integrado debe ser actualizada. Esos eventos son detectados al momento de cambiar la fecha de la última modificación a alguna de las ontologías base que fueron mezcladas. Al momento que esta fecha cambia, se genera un evento que se envía al analizador.

Analizador

En esta fase es donde se realiza el proceso de análisis de qué se requiere hacer para el proceso de integración de las ontologías, considerando los dos posibles casos: uno basado en peticiones del usuario y el otro en el dominio ontológico. Para ello, esta fase requiere invocar al proceso de alineación y de análisis de consistencia de las ontologías, para determinar que se puede hacer. Cada caso posee un conjunto de supuestos que permiten realizar el análisis de qué está sucediendo, y qué se debe hacer para realizar o mantener la integración de las ontologías. El analizador parte del hecho de que los siguientes Supuestos se cumplen:

Supuesto1: *si existe un alineamiento entre dos ontologías A y B, y estas son parte de un mismo dominio (dominios equivalentes), implica que estas se pueden mezclar.*

Argumentos: para realizar una mezcla de ontologías es necesario encontrar correspondencias de equivalencia entre las ontologías a mezclar, para encontrar correspondencias, estas deben compartir un mismo dominio. Por otro lado, un alineamiento es una serie de correspondencias semánticas entre ontologías, por ende, si existe un alineamiento estas ontologías se pueden mezclar.

Supuesto2: *si existe un alineamiento grande entre dos ontologías A y B, se debe realizar una mezcla débil entre las dos ontologías*

Argumentos: esto es, porque al existir gran cantidad de conceptos alineados, al copiar A en la ontología resultado C, la mayor parte o la totalidad del conocimiento de B será copiado también.

Supuesto3: *si existe un alineamiento pequeño entre dos ontologías A y B, se debe realizar una mezcla fuerte entre las dos ontologías.*

Argumentos: porque al quedar alineados pocos conceptos, al copiar A en la ontología resultado C, la mezcla débil deja gran parte del conocimiento de B no copiado en este enriquecimiento. Así, se necesita una mezcla fuerte para copiar los conceptos dejados por fuera.

Supuesto4: *si dos ontologías A y B, no son parte de un mismo dominio (dominios equivalentes), implica que estas se deben enlazar.*

Argumentos: al intentar integrar dos ontologías que están en dominios diferentes, es decir, que el conocimiento que cada una maneja no se parece, su alineación será casi nula. En ese sentido, una mejor solución para integrar, es una ontología intermedia que maneje conceptos comunes entre las ontologías. De esta manera, cada ontología se mantiene intacta, pero hay una meta-ontología que los comunica/enlaza, pudiéndose razonar de un lado al otro sin problemas

Supuesto5*: si existe un alineamiento entre dos ontologías A y B de diferentes dominios, se debe realizar un enlazado débil entre las dos ontologías*.

Argumentos: al ser de diferentes dominios se debe enlazar, pero además, por el hecho de existir un alineamiento débil entre ellas, la técnica a usar es enlazado débil.

Supuesto6: *si no existe un alineamiento entre dos ontologías A y B, y estas no son parte de un mismo dominio, se debe enlazar las ontologías con ayuda de un experto del dominio*.

Argumentos: la única manera de crear un enlace sin alineamiento es generando conceptos intermedios que enlacen las ontologías, para ello es necesario la intervención de un experto que maneje los dos dominios.

Supuesto7: *si alguna de las dos ontologías A y B a ser integradas son inconsistentes, entonces no se realiza ningún tipo de integración*.

Argumentos: esto es porque si se integra conocimiento inconsistente desde la base, este acarreará la inconsistencia de la integración. Esta verificación se hace a nivel de un razonador.

Basados en esas hipótesis, se presenta a continuación el macro algoritmo para ambos casos:

Algoritmo del Analizador

0. INICIO
1. SI las ontologías de entrada son consistentes (Supuesto7)
    1.1 SI el usuario realiza una petición (caso *basado en el usuario*), se ejecutan las

siguientes reglas para la escogencia del servicio:

1.1.1 SI la petición fue mezclar entonces

1.1.1.1 Determinar cuáles de los Supuesto2, y 3 se cumplen

1.1.1.2 Dependiendo si se cumplen y cuales se cumplen

1.1.1.2.1 Invocar al servicio de mezcla fuerte o débil

1.1.1.2.2 Indicar si se pudo o no mezclar

1.1.2 FIN SI 1.1.1

1.1.3 SI la petición fue enlazar

1.1.3.1 Determinar cuáles de los Supuesto5 o 6 se cumplen

1.1.3.2 Dependiendo si se cumplen y cuales se cumplen

1.1.3.2.1 Invocar al servicio de enlazado fuerte o débil

1.1.3.2.2 Indicar si se pudo o no enlazar

1.1.4 FIN SI 1.1.3

1.2 FIN SI 1.1

1.3 SI ocurrió un cambio en las ontologías ya integradas en el nivel base (caso *del dominio ontológico*)

1.3.1 Determinar cuáles de los Supuesto1 o 3 se cumplen

1.3.2 SI Supuesto1 se cumple, determinar cuáles de las Supuesto2 o 3 se cumplen

1.3.2.1 Dependiendo si se cumplen y cuales se cumplen

1.3.2.1.1 Invocar al servicio de mezcla fuerte o débil

1.3.2.1.2 Indicar si se pudo o no mezclar

1.3.3 FIN SI 1.3.2

1.3.4 SI Supuesto4 se cumple, determinar cuáles de los Supuesto5 o 6 se cumplen

1.3.4.1 Dependiendo si se cumplen y cuales se cumplen

1.3.4.1.1 Invocar al servicio de enlace fuerte o débil

1.3.4.1.2 Indicar si se pudo o no enlazar

1.3.5 FIN SI 1.3.4

1.4 FIN SI 1.3

2. FIN SI 1.
3. FIN

En este algoritmo, lo primero que se verifica es el Supuesto7 para determinar si las ontologías son inconsistentes (paso 1). Seguidamente se determina en cual caso se está (pasos 1.1. o 1.3), si fue requerida por el usuario o basado en los cambios en las ontologías. En ambos casos, igualmente se verifican las Supuesto1 o 4 (si se desea mezclar (pasos 1.1.1 y 1.3.2) o enlazar (pasos 1.1.3 y 1.3.4)). Después, en ambos casos se pasa a invocar la técnica respectiva (pasos 1.1.1.2.1, 1.1.3.2.1, etc.). Las Supuesto1, 2, 3 están diseñados para determinar si hay que realizar la mezcla, por otro lado, el Supuesto 4, 5 y 6 sirven para decidir si hay que realizar un enlazado débil o fuerte.

Planificador

Es el encargado de realizar la orquestación o coreografía de los servicios a ejecutar. El resultado es un archivo con la descripción del orden de ejecución. Los pasos genéricos de orquestación para los servicios especificados en este trabajo, basado en la petición del usuario o basado en el dominio ontológico, son los siguientes:

Mezcla débil:

1. Chequear que las ontologías de entrada sean consistentes.
2. Realizar la alineación
3. Realizar la mezcla.
4. Retornar la mezcla

Mezcla fuerte:

1. Chequear que las ontologías de entrada sean consistentes.
2. Realizar la alineación
3. Realizar una mezcla débil.
4. Completar el enriquecimiento hasta que todo el conocimiento sea incluido.
5. Retornar la mezcla

Enlazado débil:

1. Chequear que las ontologías de entrada sean consistentes.
2. Realizar la alineación
3. Realizar el enlazado.
4. Retornar la ontología de enlace.

Enlazado fuerte:

1. Chequear que las ontologías de entrada sean consistentes.
2. Realizar la alineación
3. Realizar el enlazado con la ayuda de un experto.
4. Retornar la ontología de enlace.

Alineamiento Emergente de Ontologías:

1. Chequear que las ontologías de entrada sean consistentes.
2. Realizar la selección de la técnica de alineamiento.
3. Retornar el alineamiento.

Ahora bien, el planificador podría realizar orquestaciones de servicios más complejos, según las necesidades de minería ontológicas. Debería poder ensamblar dinámicamente los servicios requeridos en un momento dado.

Ejecutor

En esta fase se ejecutan todos los servicios que se definen en la orquestación del planificador, ya sea basado en peticiones del usuario o basado en cambios que se detectan en las ontologías. En esta fase se sigue el flujo de ejecución de servicios definido por la orquestación, sobre las ontologías a integrar (en el siguiente capítulo se dan ejemplos de su funcionamiento).

Capítulo 5

En este capítulo se describen los experimentos realizados con las diferentes propuestas realizadas en la tesis: con los algoritmos de la escogencia emergente de la técnica de alineamiento (selecciona una o varias técnicas de alineamiento, como las mejores para alinear las ontologías de entrada) y de mezcla fuerte, así como también con el middleware reflexivo para la integración de ontologías.

5.1. Protocolo experimental para el sistema de recomendación de técnicas de Alineamiento de Ontologías usando ABC

A continuación se presentan los experimentos realizados con el algoritmo ABC, para la selección emergente de técnicas de alineamiento.

Diseño del Experimento

El objetivo principal de estos experimentos es verificar que el algoritmo ABC propuesto este realizando la selección de las mejores técnicas de alineamiento. En nuestro caso, una técnica será buena en función del número de nodos que alinea y del tiempo de ejecución que toma para realizarlo.

5.1.1. Datos de Prueba

Para realizar los experimentos con el algoritmo de selección de técnicas de alineamiento, fueron usadas las ontologías propuestas en [34], mostradas en las Figs. 5.1, 5.2 y 5.3. La Fig. 5.1 muestra dos ontologías de automóviles a ser alineadas, donde la de la izquierda muestra de manera específica automóviles alemanes e italianos, y la de la derecha muestra de manera general todos los automóviles europeos. En la Fig. 5.2 se describe una

parte de la anatomía del cuerpo humano, en específico, la vinculada a los ojos, la de la izquierda muestra de manera general todo como tejidos, y la de la derecha subdivide los conceptos como partes de subsistemas.

Fig. 5.1. Ontologías de Automóviles

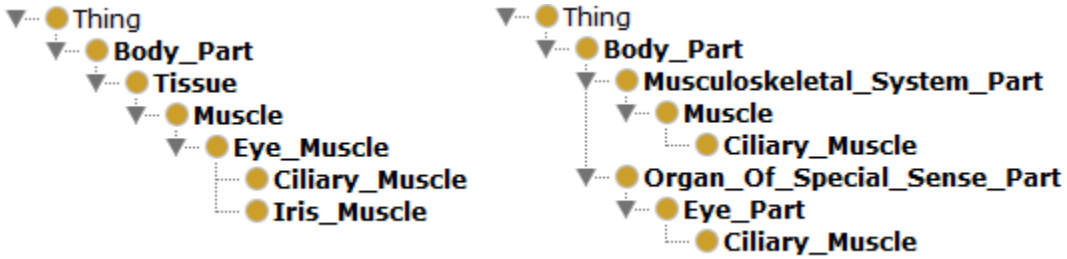

Fig. 5.2 Ontologías sub anatomía del cuerpo humano

Otro par de ontologías usadas describen el universo del hardware de los computadores (ver Fig. 5.3). La de la izquierda divide el hardware en accesorios, computadores de escritorio y portátiles; y la de la derecha tiene marcas que venden/fabrican los productos de la ontología de la izquierda.

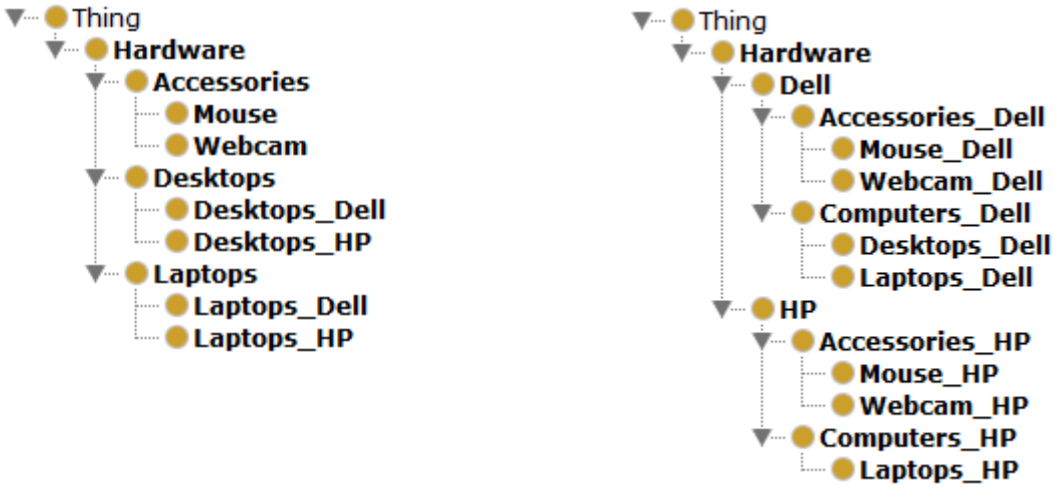

Fig. 5.3 Ontologías de computadores

5.1.2. Parámetros, Medidas de Rendimiento y Técnicas a Evaluar

Para realizar las pruebas al sistema de recomendación de técnicas de alineamiento de ontologías, se define el formato presentado en la tabla 5.1 para cada caso de estudio, en la que se especifican cada uno de los campos a analizar a continuación.

Ciclo Máximo: Parámetro que define la Condición de parada de nuestro algoritmo ABC, en su proceso de selección de la mejor técnica para alinear dos ontologías dadas. Otro criterio de parada posible es que las abejas decidan todas, una técnica como la mejor, si aún no se ha llegado al ciclo máximo para realizar la parada.

Tiempo: Medida de rendimiento que especifica cuánto dura la técnica seleccionada en promedio por corrida, para alinear las ontologías bajo estudio.

#nodos: Medida de rendimiento explicada en el capítulo 2, que indica cuantos nodos (conceptos) logra alinear la técnica seleccionada en esa corrida. Se usa esa medida en particular, porque nuestro algoritmo se basa en que las técnicas para encontrar las similitudes semánticas son buenas y confiables; el algoritmo se centra en escoger la técnica que encontró más similitudes semánticas.

#escogencia de Técnica: Número de veces que se escogió cada técnica en el total de corridas realizadas

#*corridas*: parámetro que especifica el número de veces que se ejecutó nuestro algoritmo para probar su robustez. Los resultados estadísticos de estas corridas son determinados a partir de dichas corridas (promedios, varianzas, entre otros). Todos los experimentos se realizaron con el #corridas en 30.

Tabla 5.1 Parámetros a considerar en cada uno de los experimentos con el algoritmo de recomendación de técnicas de alineamiento de ontologías.

Ciclo Máximo	Tiempo	#nodos	#escogencia de Técnica

Para cada experimento se completa una tabla similar. Se realizaron diferentes experimentos variando otro parámetro de los algoritmos ABC: *el tamaño de la colonia*, que especifica el número de abejas (empleadas, curiosas, exploradoras) usadas para resolver el problema de selección de la mejor técnica de alineamiento para dos ontologías dadas.

Las técnicas de similitud semantica (algoritmos de alineamiento) usadas por el algoritmo ABC, se encuentran implementadas en el API [44]. Estas técnicas determinan el numero de fuentes de alimento a analizar, que son en nuestro caso 8 (corresponden al mismo número de técnicas a estudiar). Cada técnica retorna un valor entre 0%-100%, para indicar el grado de similitud entre cada par de conceptos (clases) evaluados. Estas técnicas son descritas brevemente a continuación (una descripción detalladas de las mismas se encuentra en [28]):

a) Class Structure: esta técnica se basa en encontrar similitud tomando en cuenta la estructura del grafo de las clases.

b) Distance edited name: es la distancia semántica que se requiere para que se parezcan los nombres de las clases.
c) Distance edited subclass name: es la distancia semántica que se requiere para que se parezcan los nombres de las subclases, de las clases que se estan comparando.
d) Name and properties: compara la similitud de los nombres de las clases, y las propiedades que estas poseen.
e) Same names: encuentra la similitud semántica identica de los nombres de las clases en comparación.
f) Distance SMOA name (A String Metric for Ontology Alignment) [43]: calcula la similitud entre dos entidades como la función de sus partes en común menos sus diferencias. La función de coincidencia está basada en una métrica de subcadena, en la que se calcula la subcadena común más grande entre dos cadenas. En cuanto a la función de diferencia, se basa en la longitud de las cadenas no coincidentes que han resultado de la etapa de coincidencia inicial. A la diferencia se le da un peso menos importante en el cálculo de la similitud global.
g) String Distance: procesa clases, propiedades e individuos, todos como cadenas de caracteres, para encontrar similitudes entre ellos.
h) Sub structures distance: esta técnica se basa en encontrar la similitud, tomando en cuenta la estructura del grafo de las subclases de las clases en comparación.

5.1.3. Resultados Obtenidos

A continuación pasamos a mostrar las tablas que describen los resultados de los diferentes experimentos realizados. Todas las pruebas se corrieron 30 veces, y se usó como umbral de similitud en las técnicas usadas de alineamiento (ver seccion anterior) el valor de

80% (medida mínima de parecido entre dos nodos, para considerar que los mismos quedan alineados (son similares) por el método que se esté usando).

El primer grupo de tablas (tablas 5.2, 5.3 y 5.4) muestran los resultados del algoritmo para seleccionar la técnica de alineamiento adecuada para alinear las ontologías de automóviles, para diferentes valores de los parámetros de nuestro algoritmo. En la siguiente sección se detallan los resultados, pero se puede constatar inicialmente que el algoritmo escoge como mejores técnicas normalmente a las técnicas b, d, e y f, y nunca escoge, o escoge muy poco, a las técnicas a y h.

Tabla 5.2 Ontologías de Automóviles. Tamaño de la Colonia al doble de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	5:12	0	a) 0
	1:00	3	**b) 6**
	2:01	3	c) 3
	2:01	3	d) 5
	1:33	3	**e) 6**
	2:33	3	f) 5
	2:00	3	g) 5
	2:55	3	h) 0
100	3:12	0	a) 0
	1:00	3	b) 4
	1:01	3	c) 4
	1:01	3	d) 3
	0:50	3	**e) 6**
	1:33	3	f) 5
	1:00	3	g) 4
	1:42	3	h) 4
25	1:00	0	a) 0
	0:50	3	b) 5
	0:51	3	c) 2
	0:49	3	**d) 6**
	0:41	3	**e) 6**
	1:00	3	f) 5
	0:59	3	g) 4
	1:12	3	h) 2

Tabla 5.3 Ontologías de Automóviles. Tamaño de la Colonia igual al de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	5:12	0	a) 0
	1:00	3	b) 5
	2:01	3	c) 5
	2:01	3	d) 4
	1:33	3	**e) 9**
	2:33	3	f) 5
	2:00	3	g) 0
	2:55	3	h) 2
100	3:12	0	a) 0
	1:00	3	b) 5
	1:01	3	c) 4
	1:01	3	d) 9
	0:50	3	**e) 9**
	1:33	3	f) 0
	1:00	3	g) 3
	1:42	3	h) 0
25	1:00	0	a) 0
	0:50	3	b) 6
	0:51	3	c) 3
	0:49	3	d) 2
	0:41	3	**e) 7**
	1:00	3	**f) 7**
	0:59	3	g) 3
	1:12	3	h) 2

Tabla 5.4 Ontologías de Automóviles. Tamaño de la Colonia a la mitad de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	5:12	0	a) 0
	1:00	3	**b) 7**
	2:01	3	c) 5
	2:01	3	d) 4
	1:33	3	e) 4
	2:33	3	f) 5
	2:00	3	g) 3
	2:55	3	h) 2

100	3:12	0	a) 0
	1:00	3	b) 3
	1:01	3	**c) 6**
	1:01	3	d) 5
	0:50	3	e) 5
	1:33	3	f) 4
	1:00	3	g) 4
	1:42	3	h) 4
25	1:00	0	a) 0
	0:50	3	b) 5
	0:51	3	c) 4
	0:49	3	d) 4
	0:41	3	e) 4
	1:00	3	**f) 8**
	0:59	3	g) 2
	1:12	3	h) 3

En este segundo grupo de tablas (tablas 5.5, 5.6, 5.7) se muestran los resultados para las ontologías que describen la anatomía del ojo, para diferentes valores de los parámetros de nuestro algoritmo (tamaño de la colonia y ciclos máximos). Rápidamente se puede ver que el algoritmo escoge como mejores técnicas normalmente a las técnicas d y f, y nunca escoge a las técnicas a, b, c, e y g. En la siguiente sección se analizan en detalle los resultados.

Tabla 5.5 Ontologías de la Anatomía del ojo. Tamaño de la Colonia al doble de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	4:49	0	a) 0
	1:14	3	b) 0
	2:10	3	c) 0
	2:12	5	d) 11
	1:42	3	e) 0
	2:54	5	**f) 12**
	2:15	3	g) 0
	3:10	5	h) 7

100	4:01	0	a) 0
	1:01	3	b) 0
	0:59	3	c) 0
	1:10	5	**d) 14**
	0:49	3	e) 0
	1:20	5	f) 12
	1:00	3	g) 0
	1:23	5	h) 4
25	0:50	0	a) 0
	0:59	3	b) 0
	0:53	3	c) 0
	0:47	5	d) 11
	0:45	3	e) 0
	0:35	5	**f) 14**
	0:59	3	g) 0
	0:53	5	h) 5

Tabla 5.6 Ontologías de la Anatomía del Ojo. Tamaño de la Colonia igual al de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	4:49	0	a) 0
	1:14	3	b) 0
	2:10	3	c) 0
	2:12	5	d) 12
	1:42	3	e) 0
	2:54	5	**f) 15**
	2:15	3	g) 0
	3:10	5	h) 3
100	4:01	0	a) 0
	1:01	3	b) 0
	0:59	3	c) 0
	1:10	5	d) 7
	0:49	3	e) 0
	1:20	5	**f) 18**
	1:00	3	g) 0
	1:23	5	h) 5
25	0:50	0	a) 0
	0:59	3	b) 0
	0:53	3	c) 0
	0:47	5	**d) 12**
	0:45	3	e) 0
	0:35	5	f) 8
	0:59	3	g) 0

	0:53	5	h) 10

Tabla 5.7 Ontologías de la Anatomía del Ojo. Tamaño de la Colonia a la mitad de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	4:49	0	a) 0
	1:14	3	b) 0
	2:10	3	c) 0
	2:12	5	**d) 12**
	1:42	3	e) 0
	2:54	5	**f) 12**
	2:15	3	g) 0
	3:10	5	h) 6
100	4:01	0	a) 0
	1:01	3	b) 0
	0:59	3	c) 0
	1:10	5	d) 7
	0:49	3	e) 0
	1:20	5	**f) 14**
	1:00	3	g) 0
	1:23	5	h) 9
25	0:50	0	a) 0
	0:59	3	b) 0
	0:53	3	c) 0
	0:47	5	**d) 12**
	0:45	3	e) 0
	0:35	5	f) 9
	0:59	3	g) 0
	0:53	5	h) 9

Finalmente, el último grupo de experimentos se realizó para alinear a las ontologías que describen el hardware del computador (ver tablas 5.8, 5.9 y 5.10).El resultado es muy similar al anterior, el algoritmo solo escoge como mejores técnicas a las técnicas d, f, y h. En la siguiente sección se analizan estos resultados.

Tabla 5.8 Ontologías del Hardware de un Computador. Tamaño de la Colonia al doble de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	4:32	0	a) 0
	1:23	4	b) 0
	2:15	7	c) 0
	3:17	9	d) 7
	1:56	4	e) 0
	3:12	9	**f) 14**
	3:01	4	g) 0
	3:23	9	h) 9
100	4:14	0	a) 0
	1:16	4	b) 0
	1:21	7	c) 0
	1:15	9	**d) 11**
	1:01	4	e) 0
	1:19	9	**f) 11**
	1:23	4	g) 0
	1:31	9	h) 8
25	1:02	0	a) 0
	1:06	4	b) 0
	0:56	7	c) 0
	0:49	9	d) 8
	0:49	4	e) 0
	0:46	9	**f) 17**
	1:06	4	g) 0
	0:59	9	h) 5

Tabla 5.9 Ontologías del Hardware de un Computador. Tamaño de la Colonia igual al de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	4:32	0	a) 0
	1:23	4	b) 0
	2:15	7	c) 0
	3:17	9	**d) 11**
	1:56	4	e) 0
	3:12	9	**f) 11**
	3:01	4	g) 0
	3:23	9	h) 8
100	4:14	0	a) 0
	1:16	4	b) 0
	1:21	7	c) 0
	1:15	9	**d) 16**
	1:01	4	e) 0
	1:19	9	f) 10
	1:23	4	g) 0
	1:31	9	h) 4
25	1:02	0	a) 0
	1:06	4	b) 0
	0:56	7	c) 0
	0:49	9	d) 12
	0:49	4	e) 0
	0:46	9	**f) 14**
	1:06	4	g) 0
	0:59	9	h) 4

Tabla 5.10 Ontologías del Hardware de un Computador. Tamaño de la Colonia a la mitad de la cantidad de técnicas de alineamientos.

Ciclos Máximo	Tiempo (s)	#nodos	#escogencia de Técnica
250	4:32	0	a) 0
	1:23	4	b) 0
	2:15	7	c) 0
	3:17	9	**d) 15**
	1:56	4	e) 0
	3:12	9	f) 11
	3:01	4	g) 0
	3:23	9	h) 4

100	4:14 1:16 1:21 1:15 1:01 1:19 1:23 1:31	0 4 7 9 4 9 4 9	a) 0 b) 0 c) 0 **d) 13** e) 0 f) 9 g) 0 h) 8
25	1:02 1:06 0:56 0:49 0:49 0:46 1:06 0:59	0 4 7 9 4 9 4 9	a) 0 b) 1 c) 4 **d) 9** e) 0 **f) 9** g) 0 h) 7

5.1.4. Análisis de resultados

En las tablas 5.2, 5.3 y 5.4 se puede observar que para las ontologías del universo de los automóviles, las diferentes técnicas logran siempre alinear 3 nodos, sin importar la técnica de similitud semántica usada. Eso implica que nuestro algoritmo realiza la escogencia basada en el tiempo que dura la técnica para realizar el alineamiento. Escoge indistintamente a b, c, d, e, f, g y h, aunque en ciertos casos despunta b, d, e, o f, esto se debe a que por complejidad computacional, son las técnicas que tardan menos en su ejecución. Por otro lado, en cuanto a los parámetros del algoritmo, aumentar el # de Ciclos Máximo no ayuda, ya que los tiempos de ejecución no difieren en grandes cantidades. Con respecto al Tamaño de la Colonia, vemos que la misma influye porque al reducirlo mucho, se pueden dejar fuentes de alimentos sin explorar.

Con respecto a los resultados de la alineación de las ontologías de la anatomía del ojo, los alineamientos varían entre 3 y 5 nodos. Vemos así que el número de nodos alineados pasa a tener un peso fundamental en la decisión de las abejas. Ahora bien, las únicas técnicas que logran alinear 5 nodos son d, f y h, por eso son las únicas que se seleccionan (indistintamente

entre ellas), siendo más eficientes en tiempo de ejecución d y f (por eso se escogen más). Las técnicas d, f y h logran alinear más porque son técnicas que no solo se basan en el nombre de la clase, sino que incluyen otros aspectos, como las propiedades y la estructura del grafo de clases; y d y f duran menos porque, como ya se mencionó, no tienen alta complejidad computacional. En cuanto al # de Ciclos Máximo, este parámetro en este caso se comporta diferente al anterior (es muy importante), un valor menor a 30 puede hacer que no se escoja una buena técnica, ya que se podría llegar al caso donde las abejas no tengan las suficientes iteraciones para llegar a una de las técnicas (fuentes de alimento) que alinean los 9 nodos, y darla así como una de las mejores. Con respecto al Tamaño de la Colonia, su influencia es, al igual que antes, importante. Al reducirlo mucho se pueden dejar fuentes de alimentos sin explorar.

Finalmente, con respecto a los resultados del alineamiento de las ontologías del hardware de las computadoras, son más variados los alineamientos de las técnicas: 4, 7 y 10 nodos. Eso hace que nuestro algoritmo privilegié las que más alinean nodos (d, f y h), pero escogiendo solo en unos pocos casos a técnicas que alinea menos (b y c). Así, sigue siendo el tiempo de ejecución el segundo criterio para escoger entre las mejores. Para esos casos excepcionales en que se escogen técnicas que alinean menos nodos (escogencia técnicas b y c), la explicación viene dada por los parámetros. Esos resultados corresponden cuando se usa la mitad de la población con respecto a las técnicas, y un número de ciclos muy reducidos. Esto nos indica que el algoritmo no tiene el suficiente tiempo, ni el número de abejas, para explorar todas las posibles alternativas, quedándose, por consiguiente, en algunos casos, con la primera técnica escogida, sin poder moverse a mejores zonas del espacio de soluciones. Vemos así en estos experimentos la importancia de esos dos parámetros, pareciendo que un número ideal sería que la población sea del tamaño al menos igual al número de técnicas de alineamiento a evaluar, y que el número de ciclos sea al menos 100. Aquí también vemos que las técnicas que alinean más nodos de nuevo son d, f y h, porque son técnicas que logran

encontrar buena cantidad de similitudes (sus criterios de similaridad son los mejores, ya que se basan en comparar mucho más que solo el nombre de los conceptos, como lo realizan otras técnicas) en un tiempo prudencial, en relación al resto de las técnicas, para estos casos de estudios.

La literatura indica que un buen parámetro para el tamaño de la colonia de abejas es el mismo al de las fuentes de alimentos [27], para garantizar la exploración de todas las fuentes. Vemos en las últimas pruebas que eso se constata. Para el resto de estudios, logramos obtener las mejores técnicas usando como tamaño de la colonia la mitad del número de las fuentes de alimentos, que recordemos que en nuestro caso era las 8 técnicas de similitud semántica. La razón de que no influye en el primer experimento es que el número de nodos alineados por todas las técnicas es idéntico. Si no se exploran todas las fuentes de alimentos no importa, al menos una logra encontrar un buen alineamiento.

Un último comentario tiene que ver con los trabajos parecidos a nuestra propuesta. En la literatura no encontramos algoritmos que automaticen el proceso de selección de técnicas de alineamiento, dada dos ontologías, como nuestra propuesta. Por otro lado, el único trabajo basado en algoritmos bio-inspirados en el área de alineamientos es [46], pero esté no realiza una selección de las mejores técnicas de alineamiento, en ese trabajo se propone otro mecanismo de alineamiento de ontologías usando la técnica PSO.

5.2. Protocolo experimental para el Sistema de Mezcla Fuerte de Ontologías

En esta sección se presentan los experimentos para probar el algoritmo de mezcla fuerte.

Diseño del Experimento

El objetivo principal de estos experimentos es verificar si la técnica de mezcla propuesta en este trabajo en la sección anterior, denominada mezcla fuerte, es capaz de integrar todo el conocimiento de las ontologías de entrada, sin dejar conocimiento por fuera. Para eso definiremos varias métricas que nos permitan hacer esa comprobación.

5.2.1 Datos de Prueba

Para la mezcla fuerte se usaron las mismas ontologías de [34], descritas en la sección anterior para probar al algoritmo ABC (ver las Figs. 5.1, 5.2 y 5.3). Esto nos permitió comparar los resultados con [34]. En específico, el umbral de similitud usado por el algoritmo de alineamiento, que usa nuestro algoritmo de mezcla fuerte fue de 95% (es el porcentaje mínimo de parecido entre dos nodos, para indicar que son alineados).

5.2.2. Medidas de Rendimiento

Los criterios de rendimiento que se usan en este trabajo (Cobertura, Compacidad y Redundancia) fueron descritos en el capítulo 2, los cuales permiten comparar los resultados obtenidos con [34]. Estas métricas permiten evaluar la calidad de la mezcla resultante.

5.2.3. Resultados Obtenidos

Recordemos que queremos evaluar la capacidad de nuestro algoritmo de mezcla fuerte debe incluir todo el conocimiento de las ontologías de entrada, sin dejar conocimiento por fuera, tal como se mostró en el capítulo 3, en la versión final de nuestro algoritmo, donde se incorporan los *leftovers*.

Para el primer par de ontologías sobre automóviles europeos se usó como técnica de similitud semántica (de alineamiento) una de las ganadoras de la tabla 5.4 (ver sección anterior), en específico la técnica f, basada en la distancia de nombres (denominada SMOA) [43]. La mezcla de las ontologías de los automóviles se muestra en la Fig. 5.4, en la cual se puede observar en Fig. 5.4.a) se muestran las ontologías a mezclar, en Fig. 5.4.b) se observa la ontología mezcla resultado, la cual es la mezcla usando como base la ontología de la izquierda, esta mezcla resultó ganadora debido a que presenta los mejores criterios de rendimientos, y se observa que los carros alemanes y los carros italianos pasan a ser parte de los carros europeos. En la Fig. 5.4.c) se muestra la ontología mezcla desechada, por tener peores criterios de rendimiento. La tabla 5.11 muestra las comparaciones con los resultados obtenidos en [34].

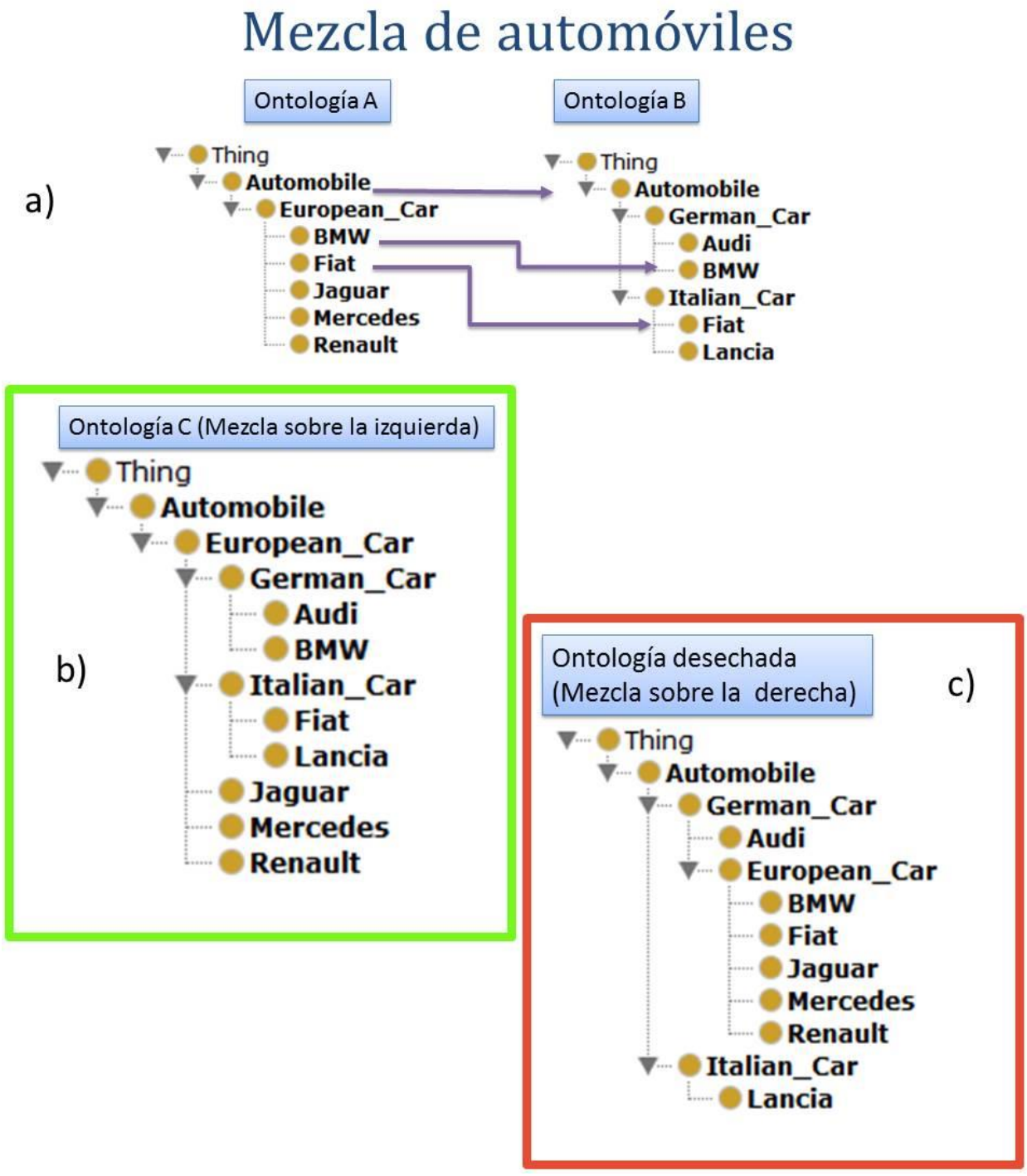

Fig. 5.4 Mezcla de las ontologías sobre automóviles europeos.

La mezcla de las ontologías de la anatomía del ojo es mostrada en la Fig. 5.5. En la Fig. 5.5.a) se observan las ontologías base, en la Fig. 5.5.b) la ontología mezcla desechada, debido a que los criterios de rendimiento son peores que la otra ontología mezcla que es la Fig. 5.5.c) (ver tabla 5.11), esta ontología resultante inserta conceptos y crea subdivisiones, sin crear redundancia como en la Fig. 5.5.b), acercándose a los resultados de los expertos referenciados en [34]. Las diferencias entre los resultados del experto, y el resultado de nuestro algoritmo, es que el experto siempre podrá agregar conceptos nuevos, que no existen en las ontologías de origen.

La última prueba realizada al algoritmo de mezcla fuerte se ve en la Fig. 5.6, donde se mezclan las ontologías de hardware de computadores observadas en Fig. 5.6.a).Se desecha el resultado de Fig. 5.6.b), debido a que crea inconsistencia (ver tabla 5.11). En particular, en estos resultados se muestra como el algoritmo introduce todos los conceptos de todas las ontologías, sin crear redundancia, tal como se observa en Fig. 5.6.c).

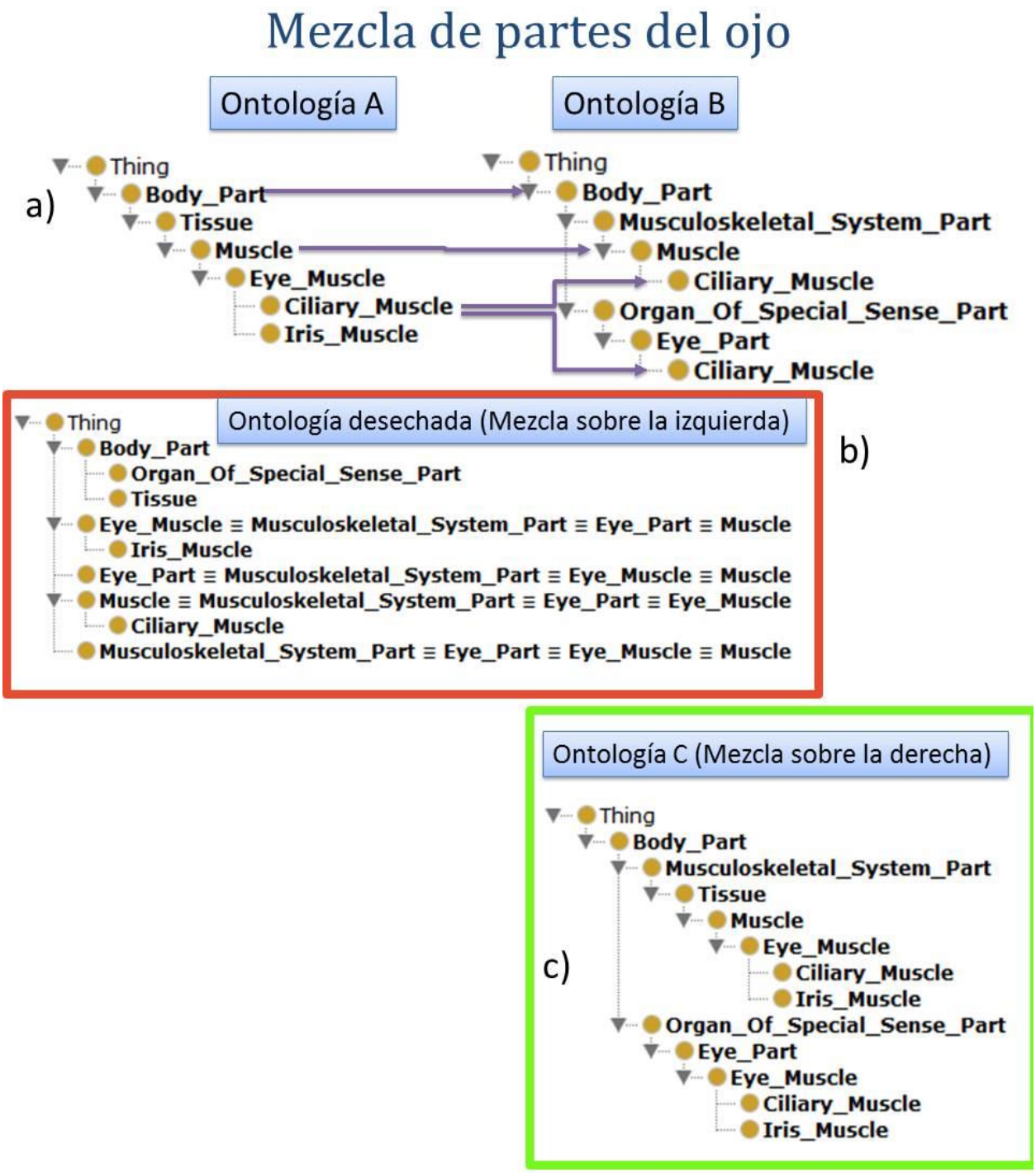

Fig. 5.5 Mezcla de las ontologías de la anatomía del ojo.

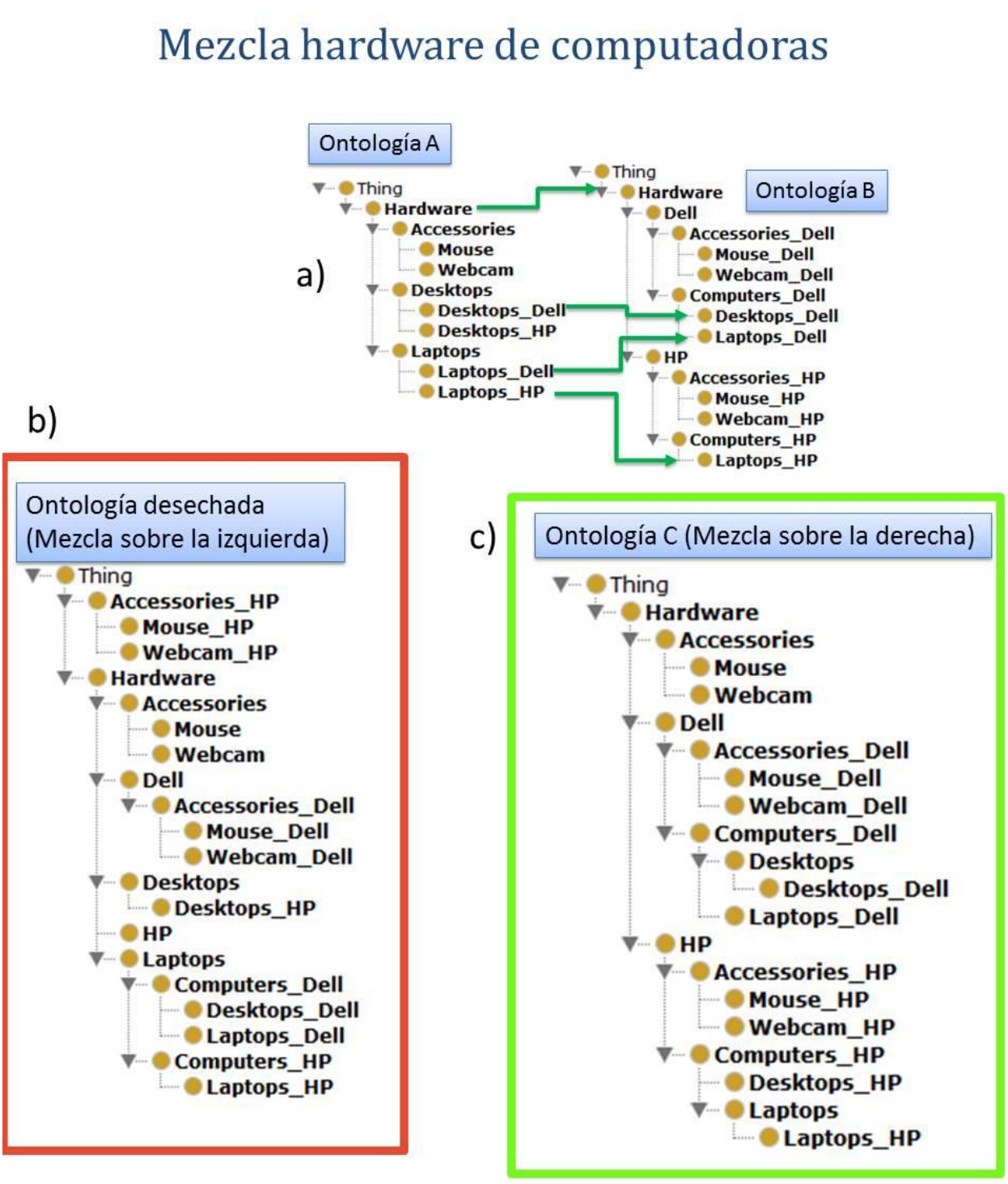

Fig. 5.6 Mezcla de las ontologías sobre hardware de computadores.

5.2.4. Análisis de resultados

Pasemos a analizar la tabla 5.11. Podemos observar que las ontologías que resultan de la mezcla fuerte que proponemos poseen una mayor cobertura, que las que resultan de las técnicas presentadas en [34], excepto para el caso de *Full Merge*. En general, en [34] las soluciones asimétricas dan peores resultados, cuando son orientadas a la ontología objetivo (*target-driven*) logran una cobertura promedio menor a 0,90 de la ontología fuente; y para el caso que son orientadas a la ontología fuente (*source-driven*) es aún peor, logran una cobertura promedio menor a 0,80 de la ontología objetivo. Ahora bien, todas las soluciones tienen una cobertura de hojas de 1, ya que no se están perdiendo ningún concepto hoja con ninguno de los algoritmos. Eso mismo se replica para el caso de compacidad.

Nuestro algoritmo ingresa los conceptos que antes quedaban por fuera en los algoritmos presentados en [34]. A esas técnicas de mezcla las hemos llamado débiles en este trabajo. Por otro lado, al comparar nuestra técnica con *Full Merge,* este último algoritmo crea redundancia en algunos casos (ver última columna de la tabla 5.11). La razón que nuestro algoritmo no tenga ese problema, es porque en esa mezcla se agregan los conceptos de ambas ontologías, sin diferenciar si los conceptos padres fueron alineados o no.

Por otra parte, nuestro algoritmo genera ontologías con métricas muy parecidas a las que definiría un experto, a excepción del primer caso, donde la compacidad relativa del experto es mayor, debido a que para efectos del estudio [34], agregaron conceptos nuevos, para crear diferenciación en algunos automóviles, y que no quedaran por fuera (los conceptos nuevo que introdujo el experto fueron: British_Car con Jaguar de hijo, y Frennch_Car con Renault de hijo). Para el resto de los casos de estudios, los expertos referenciados en [34] no agregaron conceptos nuevos al resultado de las mezclas.

En cuanto a la diferencia que se da en nuestro caso, al escoger el lado izquierdo o derecho, se debe a:

En el primer caso, se escoge la mezcla de la izquierda, porque aunque las medidas de comparación expresadas en la tabla 5.11 son idénticas para ambos casos (izquierda y derecha), al momento de evaluar la consistencia, la de la derecha es inconsistente, ya que existe la propiedad de entrada donde German_Car es un European_Car, y la mezcla de la ontología de la derecha rompe esta propiedad; a diferencia de la mezcla sobre la ontología de la izquierda que mantiene la propiedad German_Car es un European_Car (ver figura 5.4).

En el segundo caso se escoge la mezcla de la derecha, porque la mezcla sobre la izquierda es mucho más redundante que la de la izquierda, tal como se observa en la tabla 5.11.

En el tercer caso se escoge la mezcla de la derecha, porque al igual que el primer caso de estudio, aunque las medidas expresadas en la tabla 5.11, son idénticas para ambos casos (izquierda y derecha), al momento de evaluar la consistencia la de la izquierda es inconsistente, ya que existe la propiedad de entrada donde el concepto Desktops no puede ser un Laptops, y viceversa, lo que la mezcla de la ontología de la izquierda rompe diciendo que Desktops_Dell con la propiedad de entrada que es un Desktops, lo mezcla diciendo que es un Laptops (ver figura 5.6).

Estas diferencias, que suceden al escoger una ontología dada (izquierda o derecha) como ontología base, se deben a que nuestro algoritmo va recorriendo la ontología base para ir agregando los conceptos en la ontología resultante. Eso incide claramente en la ontología resultante (la ontóloga base va guiando y determinando la manera como se va integrando los conceptos de la otra ontología en la ontología resultante, lo que hace muy sensitivo al algoritmo a la ontología base).

Tabla 5.11 Comparación de resultado con [34]

		Cobertura				Compacidad		Redundancia	
Ontologías	Técnica de Mezcla	Fuente	Objetivo	Hojas	Overall	Tam. Abs	Tam. Rel	Camino a las hojas	Relativo
Automóviles	Full Merge	1	1	1	1	11	1	8	1.14
	Source-Driven	1	0.67	1	0.83	9	0.82	7	1
	Target-Driven	0.86	1	1	0.93	8	0.73	7	1
	Mezcla Fuerte sobre la izquierda	**1**	**1**	**1**	**1**	**11**	**1**	**7**	**1**
	Mezcla Fuerte sobre la derecha	**1**	**1**	**1**	**1**	**11**	**1**	**7**	**1**
	Resultado de los expertos	1	1	1	1	12	1.09	7	1
Anatomía del ojo	Full Merge	1	1	1	1	9	1	7	1.75
	Source-Driven	1	0.83	1	0.91	8	0.89	4	1
	Target-Driven	0.86	1	1	0.93	8	0.8	4	1
	Mezcla Fuerte sobre la izquierda	**1**	**1**	**1**	**1**	**9**	**1**	**8**	**2**
	Mezcla Fuerte sobre la derecha	**1**	**1**	**1**	**1**	**9**	**1**	**4**	**1**
	Resultado de los expertos	1	1	1	1	9	1	3	1
Hardware	Full Merge	1	1	1	1	20	1	13	1.3
	Source-Driven	1	0.86	1	0.93	18	0.9	10	1
	Target-Driven	0.9	1	1	0.95	19	0.95	10	1
	Mezcla Fuerte sobre la izquierda	**1**	**1**	**1**	**1**	**20**	**1**	**10**	**1**
	Mezcla Fuerte sobre la derecha	**1**	**1**	**1**	**1**	**20**	**1**	**10**	**1**
	Resultado de los expertos	1	1	1	1	20	1	10	1

5.3. Protocolo experimental para el Middleware reflexivo de integración de ontologías.

Para probar el diseño del Middleware se realizan dos corridas en frio, en las cuales se presentan dos casos, que implican que el Middleware deba funcionar en las dos modalidades para las cuales fue diseñado. La primera modalidad es basada en peticiones del usuario, y la segunda basada en eventos (los posibles eventos se describieron en el capítulo 4).

5.3.1. Caso basado en usuario

En este caso, es el usuario el que hace la petición de un servicio de manera explícita. Supongamos que dicha petición es la de realizar una mezcla fuerte. El comportamiento del Middleware es el que se observa en la Fig. 5.7, en el que se estarían ejecutando los siguientes pasos:

1. El consumidor (usuario) realiza la petición de mezcla fuerte (y envía las dos ontologías a mezclar).
2. El monitor del Middleware detecta la petición de mezcla.
3. El monitor envía las ontologías y la solicitud de mezcla al analizador.
4. El analizador utiliza el servicio de alinear (para solicitar el servicio de alineamiento, primero verifica su disponibilidad), tal que el supuesto 1 se cumple (pertenecen al mismo dominio).
5. El analizador basado en el supuesto 1, en el resultado del alineamiento, y en la petición de mezcla, verifica si se cumple el supuesto 3 (si el alineamiento es grande). Como es el caso, envía al planificador la solicitud de mezcla fuerte.
6. El planificador crea la orquestación usando WS-CDL, creando el archivo XML que contiene lo siguiente:
    - Chequear que las ontologías de entrada sean consistentes, se verifica el supuesto 7.
    - Solicitar la alineación.
    - Realizar una mezcla débil.
    - Completar el enriquecimiento de la mezcla débil, hasta que todo el conocimiento sea incluido (mezcla fuerte).
    - Retornar la mezcla

7. El planificador envía la orquestación en XML al ejecutor.
8. El ejecutor verifica la disponibilidad de los proveedores de los servicios pertinentes para realizar la mezcla fuerte, utilizando WSDL/OWL-S.
9. Si los servicios se encuentran disponibles, el ejecutor los va ejecutando según indique la orquestación.
10. La mezcla es dada como resultado al consumidor.

Vemos así como todo el ciclo MAPE es realizado de manera autónoma por nuestro middleware.

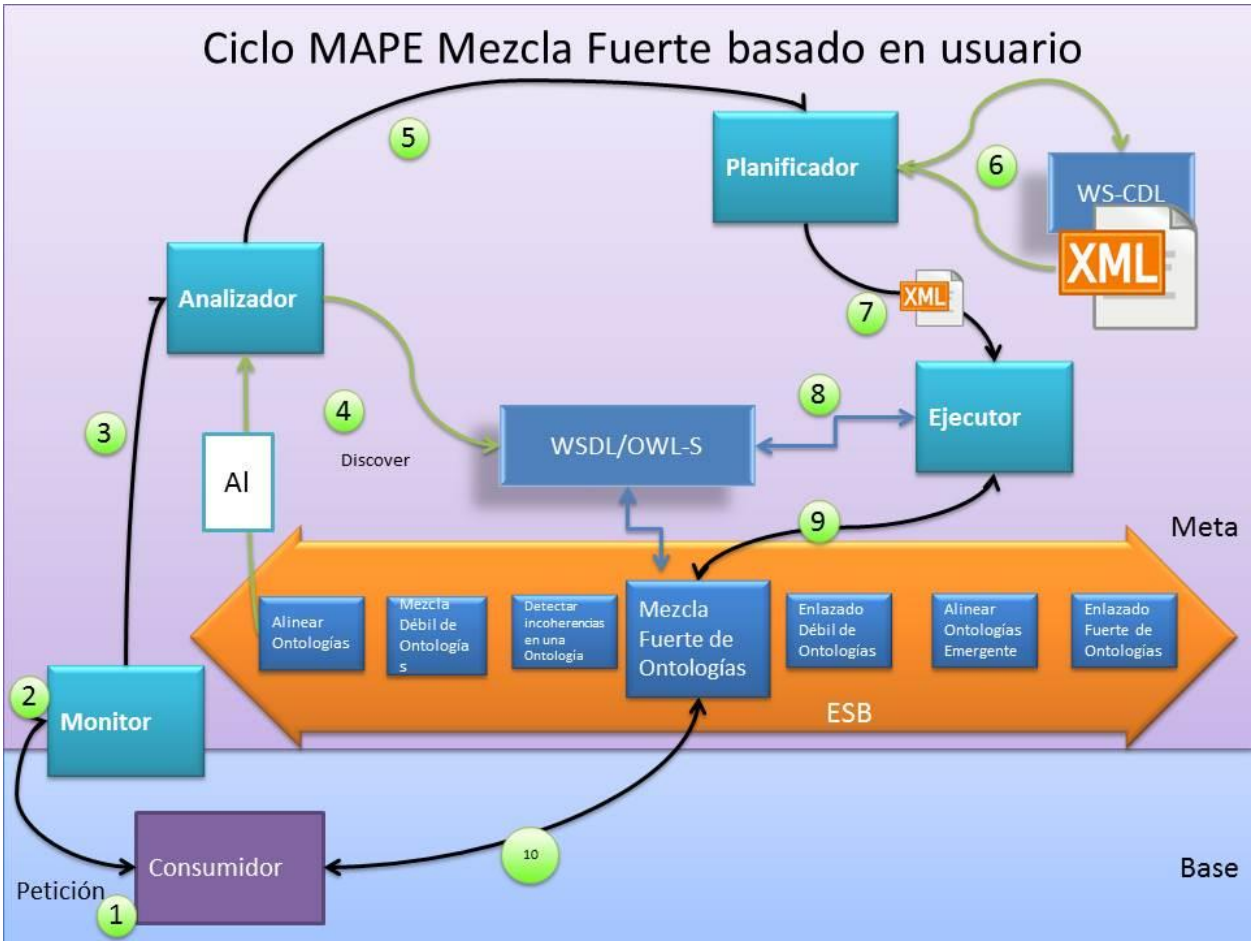

Fig. 5.7 Comportamiento del Middleware para el caso de mezcla fuerte por petición del usuario.

5.3.2. Caso basado en eventos

En este otro caso, las modificaciones hechas a las ontologías que se encuentran integradas (es decir, fueron integradas con anterioridad por petición del usuario) son detectadas por el Middleware. Supongamos que dichas ontologías están enlazadas. El comportamiento del Middleware descrito en la Fig. 5.8 consiste en los siguientes pasos:

1. Las ontologías base cambian.
2. El monitor detecta el cambio en las ontologías base (la fecha de la última modificación de alguna de ellas cambió).
3. El monitor envía un mensaje al analizador que incluye el evento de cambio de las ontologías, con las mismas.
4. El analizador utiliza el servicio de alinear (verifica su disponibilidad antes de solicitarlo) para tomar la decisión, constatando que el supuesto 4 se cumple (no pertenecen al mismo dominio).
5. El analizador basado en el supuesto 4, en el alineamiento realizado, determina que el supuesto 5 se cumple (el alineamiento existe), y le envía al planificador la solicitud de Enlazado Débil.
6. El planificador crea la orquestación usando WS-CDL, en un archivo XML que contiene:
    - Chequear que las ontologías de entrada sean consistentes.
    - Realizar la alineación
    - Realizar el enlazado.
    - Retornar la ontología de enlace.

7. El planificador envía la orquestación en XML al ejecutor.
8. El ejecutor verifica la disponibilidad de los proveedores de los servicios pertinentes para realizar el enlazado débil, utilizando WSDL/OWL-S.
9. Si los servicios se encuentran disponibles, el ejecutor los va ejecutando según indique la orquestación.
10. La ontología enlace es actualizada como resultado.

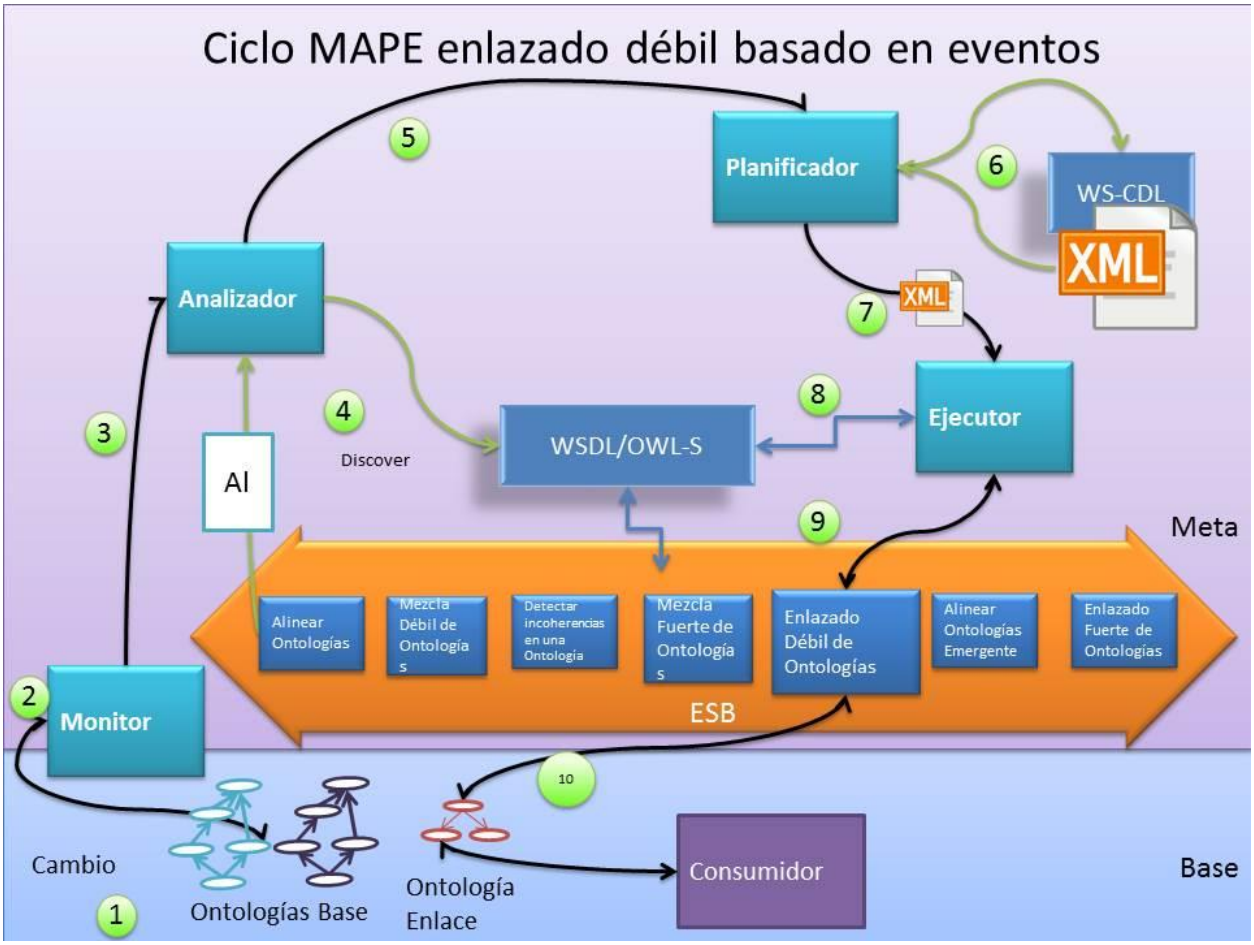

Fig. 5.8 Ciclo MAPE para el caso de enlazado débil basada en eventos.

5.3.3. Análisis

El Middleware propuesto permite trabajar bajo dos enfoques, uno en el que puede recibir peticiones de usuarios para cualquier tipo de servicio de integración de ontologías:

Alinear ontologías, Mezclar ontologías (débil o fuerte), Enlazar ontologías (débil o fuerte), Detectar incoherencias, entre otros. Otro que permite mantener la integración de ontologías que previamente fueron mezcladas, enlazadas o alineadas, si estas ontologías bases que fueron integradas cambian. Para ello, el middleware es capaz de detectar este evento, y dependiendo del análisis que el middleware hace sobre el alineamiento que puede realizar, y los dominios de las ontologías base, es capaz de actualizar la integración de las ontologías base, sin que el usuario se percate de que esto sucedió.

Capítulo 6

Conclusiones y Recomendaciones

6.1. Conclusiones

En general, en la literatura no se encontró, una arquitectura autonómica de integración de ontologías basada en servicios. El trabajo realizado del Middleware Reflexivo para la integración de Ontologías, es una arquitectura completa y extensible, donde de manera autonómica mantiene las integraciones previamente realizadas, o crea nuevas integraciones. Es interesante este tipo de arquitectura, ya que permite la incorporación de nuevos servicios semánticos (por ejemplos, servicios de aprendizaje de ontologías, de minería ontológica, etc.) de manera modular. Eso facilita la incorporación de servicios ya existentes dentro del Middleware, realizado por diferentes proveedores de servicios. Brindar una plataforma para solucionar el problema de integración de ontologías, es un aporte importante en el área de ingeniería ontológica.

En general, los trabajos que realizan la integración de ontologías, no toman en consideración mezcla y enlazado al mismo tiempo, si no que realizan uno de los dos procesos. Para ello, diseñan soluciones a problemas muy concretos, como lo es el de traducción entre conceptos de ontologías (usando técnicas de enlazado), o fusión de conocimiento distribuido (usando técnicas de mezcla). Al introducir la integración de ontologías desde un punto de vista de Minería Ontológica y orientada a servicios, permite realizar una propuesta de diseño arquitectónico donde se puede tomar en consideración todos los servicios de Minería Ontológica pertinentes para la integración ontologías, algunos de ellos los cuales son: mezcla (fuerte y débil), enlazado (fuerte y débil), alineamiento,

extracción de ontologías desde texto no estructurado, ontologías emergentes dependiendo del contexto, entre otros.

Un proceso transversal para las tareas de integración de ontologías es el de alineamiento. Debido a su importancia, en la literatura hay una gran diversidad de técnicas de alineamiento. Esas técnicas permiten comparar conceptos y relaciones entre las ontologías, y son basadas en mecanismos de análisis de similitud semántica. Así, al momento de alinear hay que escoger una técnica a priori que logre encontrar las relaciones de equivalencia para un problema dado, esto implica seleccionar la técnica adecuada para realizar la integración. En este trabajo se propuso un algoritmo de selección automática de la técnica de alineamiento para dos ontologías dadas, basado en el algoritmo de colonias de abejas ABC. Esto permite desligar al usuario del proceso de integración, dejando este proceso completamente automático. Un estudio como este no se encuentra en la literatura, ya que se enfocan en crear nuevas técnicas de similitud semántica, como es el caso de [45].

Por otro lado, los algoritmos de mezcla no toman en consideración si dejan o no conocimiento por fuera de la mezcla. El diseño e implementación del algoritmo propuesto, que llamamos mezcla fuerte, además de realizar las tareas tradicionales de enriquecimiento (mezcla débil), logra integrar los conceptos dejados por fuera de la mezcla por otros algoritmos, sin introducir redundancia a la ontología resultante, dando resultados muy parecidos al de los expertos. La comparación de este algoritmo con otros trabajos muestra como de manera exitosa incluye el conocimiento que estos otros algoritmos dejan por fuera.

En general, se alcanzaron los objetivos planteados en la propuesta, como fueron: Diseñar una middleware reflexivo basado en servicios para la Integración Automática de Ontologías; Especificar algunos de los servicios requeridos basados en *Minería Ontológica;* Analizar el estado del arte en algoritmos de mezcla y enlazado, y derivado del comportamiento de ellos, propusimos una nueva clasificación de dichos algoritmos,

llamados mezcla fuerte y débil, y enlazado fuerte y débil; Diseñar un algoritmo de mezcla fuerte; Diseñar un algoritmo para la selección automática de alineamiento, en nuestro caso basado en el algoritmo ABC; y Especificar casos de estudio para validar el proceso de integración semántica automática usando el middleware, y los algoritmos de mezcla fuerte y selección automática de alineamiento propuestos.

6.2. Recomendaciones

Para continuar con la investigación, se propone la extensión del Middleware con todos otros posibles servicios de minería semántica, como los de Generar ontologías de manera emergente dependiendo del contexto, o los de extracción de ontologías desde archivos de textos no estructurados. Además, se debe pasar a la implementación completa del Middleware, para después realizar pruebas del funcionamiento del mismo.

Un análisis que se deja como investigación abierta, es el servicio que en este trabajo denominamos como enlazado fuerte, en el cual, en un principio, necesita de la ayuda de un experto para generar nuevos conceptos y nuevas relaciones que no se tienen en las ontologías a enlazar. La investigación consiste en descubrir una técnica/manera que genere estos conceptos y relaciones nuevas, sin la necesidad del experto.

Por otro lado, en este trabajo se tomaron criterios de calidad para no dejar conocimiento por fuera, y se seleccionó un algoritmo bio-inspirado para seleccionar técnicas de similitud semántica al momento de alinear. Dichos algoritmos tienen parámetros, que según sus valores pueden acarrear o no altos costos computacionales dependiendo del tamaño de las entradas. Una mejora al trabajo sería realizar una evaluación de los tiempos de ejecución de nuestros algoritmos, y en base a ellos (y la trazabilidad obtenida) realizar tareas de paralelización en los algoritmos, para así acelerar sus tiempos de ejecución.

Para concluir esta sección, otro trabajo futuro es el diseño e implementación de una interfaz amigable para el usuario, con todos los servicios desarrollados en este trabajo, y con

los que se mencionan en el diseño del middleware (seria la interfaz del middleware, para trabajar según los dos enfoques posibles).

Referencias Bibliográficas

[1] Cuevas, A., y Guzman, A. (2010) Automatic Fusion of knowledge stored in Ontologies. Journal Intelligent Decision Technologies - Engineering and management of IDTs for knowledge management systems archive Volume 4 Issue 1, January 2010 Pages 5-19.

[2] Davidson, G. Ward, R., et all (2010). Data Mining for Ontology Development. SANDIA REPORT. SAND2010-3708. Unlimited Release. Printed June 2010

[3] d'Aquin, M., Kronbergerb, G., and Suárez-Figueroa, M. (2012). Combining Data Mining and Ontology Engineering to enrich Ontologies and Linked Data. KMi, The Open University, Walton Hall, Milton Keynes, UK

[4] Niang, C., Bouchou, B., y Lo, M. (2010). Towards Tailored Domain Ontologies. Ontology Matching. OM-2010. Proceedings of the ISWC Workshop

[5] Jin, R., y Agrawal, G. (2001). A Middleware for Developing Parallel Data Mining Applications. Proceedings of the first SIAM conference on Data Mining

[6] Fayyad, U., Piatetsky-shapiro, G., y Smyth, P. (1996). From data mining to knowledge discovery in databases. AI Magazine, 17:37-54.

[7] Grcar, M., Trdin, N., y Lavrac, N. (2012). A Methodology for Mining Document-Enriched Heterogeneous Information Networks. Oxford University Press on behalf of The British Computer Society. The Computer Journal, 2012.

[8] Hilario, M., y Kalousis, A. (2011). Semantic meta-mining. Tutorial at ECML/PKDD 2011, Athens, September 9, 2011.

[9] Lavrac, N., Kavšek, B., Flach, P. y Todorovski, L. (2004). *Subgroup discovery with CN2-SD*. J. Mach. Learn. Res., 5, 153–188.

[10] Lavrac, N., y Vavpetic A. (2011). Introduction to Semantic Data Mining (SDM). Tutorial at ECML/PKDD 2011, Athens, September 9, 2011.

[11] Lawrynowicz, A. y Potoniec, J. (2011). Learning from Description Logics (DL-learning). Tutorial at ECML/PKDD 2011, Athens, September 9, 2011.

[12] Lui, H (s.f.). Towards Semantic Data Mining. Department of Computer and Information Science, University of Oregon, Eugene, OR, 97401, USA

[13] Madkour, A. (s.f.). Semantic Web Mining: Using Association Rules for Learning an Ontology.

[14] Manuja, M. y Garg D. (2011). Semantic Web Mining of Un-structured Data: Challenges and Opportunities. International Journal of Engineering (IJE), Volume (5): Issue (3): 2011.

[15] Mechouche, A. Abadie, N. y Mustière, S. (2010) Alignment-Based Measure of the Distance between Potentially Common Parts of Lightweight Ontologies. In proceeding of: Proceedings of the 5th International Workshop on Ontology Matching (OM-2010), Shanghai, China, November 7, 2010

[16] Quboa, Q. y Saraee, M. (2013). A State-of-the-Art Survey on Semantic Web Mining. Intelligent Information Management, 2013, 5, 10-17

[17] Russell, S. y Norvig, P. (2003). Artificial Intelligence a modern approach. Prentice Hall, Upper Saddle River, N.J., 2nd international edition edition, 2003.

[18] Sun, H.; Fan, W.; Shen, W.; Xiao, T. (2010). Ontology fusion in HLA-based collaborative product development. 2010 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2010), Istanbul, Turkey, October 10-13, 2010, pp. 1-7

[19] Trajkovski I1, Lavrac N, Tolar J. (2008) SEGS: *search for enriched gene sets in microarray data*. J Biomed Inform. 2008 Aug 41 (4):588-601

[20] Vavpetic A., Lavrac N., (2012) Semantic *Subgroup Discovery Systems and Workflows in the SDM-Toolkit*. The Computer Journal Advance Access June 4, 2012.

[21] Zagal R. (2008), ALINEACIÓN DE ONTOLOGÍAS USANDO EL MÉTODO BOOSTING. Tesis de grado de Maestro en ciencias de la computación. Instituto Politécnico Nacional Centro de Investigación en Computación.

[22] Vizcarrondo, J., Aguilar, J., Exposito, E. y Subias, A. (2012). ARMISCOM: Autonomic Reflective MIddleware for management Service COMposition. GIIS'12 1569677215

[23] Nickul, D., Reitman, L., Ward, J. y Wilber, J. (2007). Service Oriented Architecture (SOA) and Specialized Messaging Patterns. Adobe Technical White Paper

[24] Endrei, M., Ang, J., et all. (2004). Patterns: Service-Oriented Architecture and Web Services. Redbooks, IBM. ISBN 073845317X

[25] Keen, M., Acharya, A., et all. (2004). Patterns: Implementing an SOA Using an Enterprise Service Bus. Redbooks, IBM. ISBN 0738490008

[26] Coulson, G. (s.f.). What is Reflective Middleware? School of Computing & Communications, Lancaster University.

[27] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report-TR06

[28] Euzenat, J. y Shvaiko, P. (2013). Ontology Matching, second edition. Springer Heidelberg New York Dordrecht London.

[29] Abraham, B. Aguilar, J. (2010). Arquitectura Inteligente de Desarrollo de Software. Universidad de Los Andes. Trabajo presentado como requisito final para optar al título de Doctor en Ciencias Aplicadas.,

[30] Huebscher, M. y Mccann, J. (2008). A survey of Autonomic Computing — degrees, models and applications. ACM Journal Name, Vol. V, No. N, Month 20YY,

[30] Lin, A. (2007). "Conceptual Model for Business-Oriented Management of Web Services", Proceedings of the 6th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems, pp. 80-85, 2007

[31] Ismaili, F. y Sisediev, B. (2008). "Web services research challenges, limitations and opportunities", WSEAS Transactions on Information Science and Applications archive Vol. 5, Issue 10, pp. 1460-1469, 2008.

[32] Chiribuca, D., Hunyadi, D., Popa, E. (2008). "The Educational Semantic Web", 8th WSEAS International Conference on Applied Informatics and Communications, pp. 314-319, 2008.

[33] Keen, M., Acharya, A., et all. (2006). Getting Started with WebSphere Enterprise Service Bus V6. Redbooks, IBM. ISBN 073849710X

[34] Raunich S., Rahm E., (2013) Target-driven merging of taxonomies with ATOM. Information Systems 42 (2014) 1–14

[35] Parundekar, R., Knoblock, C., y Ambite J. (s.f.). Linking and Building Ontologies of Linked Data. University of Southern California, Information Sciences Institute and Department of Computer Science 4676 Admiralty Way, Marina del Rey, CA 90292

[36] Yoshida, J., Kawamura, N., Tamura, K., y Watanabe, K. (2012). Hitachi Open Middleware for Big Data Processing. Hitachi Review Volume 61 Number 2 March 2012

[37] Pottinger, R. y Bernstein P. (2003). Merging Models Based on Given Correspondences. Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

[38] Dou, D., McDermott D., Qi P. (s.f.). Ontology Translation by Ontology Merging and Automated Reasoning. DARPA/DAML program. Yale University, Computer Science Department, New Haven, CT 06520

[39] Jain, P., Hitzler1, P., Sheth, A., Verma, K., y Yeh, P. (s.f.). Ontology Alignment for Linked Open Data. Kno.e.sis Center, Wright State University, Dayton, OH and Accenture Technology Labs, San Jose, CA

[40] Mendonca, M., Aguilar, J., y Perozo. N. (2014). An Emergent Ontology for Ambient Intelligence based on an Ant Colony Optimization Algorithm. Proceedings of the XL Conferencia Latinoamericana en Informática (CLEI 2014), IEEE Xplore, Montevideo, Uruguay, September 2014.

[41] Mendonca, M., Aguilar, J., y Perozo. N. (2014). Middleware Reflexivo Semántico para Ambientes Inteligentes. Proceedings of the Conferencia Nacional de Computación, Informática y Sistemas (CoNCISa 2014), pp. 24-32, October 2014.

[42] Puerto, E., Aguilar, J,, y Rodriguez., T. (2012). Automatic Learning of Ontologies for the Semantic Web: A case study on lexical learning. Proceedings of the Multiagent System Based Learning Environments (MASLE), Springer-Verlag, Cartagena, Colombia, November 2012.

[43] Rodríguez, T., Aguilar, J., y Ríos-Bolívar, A. (2013). Ontología de Tareas del Marco Ontológico Dinámico Semántico para la Web Semántica. Proceedings of the Conferencia Nacional de Computación, pp. 8-15, Naiguatá, Venezuela, October 2013.

[44] Stoilos, G., Stamou G., y Kollias, S. (2005). A String Metric for Ontology Alignment. Lecture Notes in Computer Science, 3729: 624-637, 2005.

[45] David, J., Euzenat, J., Scharffe, F., y Trojahn dos Santos, C. (2011). The Alignment API 4.0, Semantic web journal 2(1):3-10, 2011

[46] Bock, J., Danschel, C., y Stumpp, M. (2011). MapPSO and MapEVO. Results for OAEI 2011.