BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"DESARROLLO DE AGENTES MÓVILES QUE RECUPERAN INFORMACIÓN"

TESIS PROFESIONAL
QUE PARA OBTENER EL GRADO DE
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

ES PRESENTADA POR:

LORENA LEAL BANDO

ASESORA:
DRA. DARNES VILARIÑO AYALA

COASESORA:
DRA. FABIOLA LÓPEZ Y LÓPEZ

PUEBLA, PUE. Otoño de 2007

Agradecimientos

A mis padres y a mi hermano por el apoyo, paciencia y cariño que me han brindado a lo largo de mi desarrollo personal y profesional.

A mis abuelitos y tías por estar siempre presente en sus oraciones.

Agradecimientos

Un agradecimiento especial a la Dra. Darnes Vilariño Ayala que con sus consejos y paciencia alentaron la culminación de mis estudios de maestría y por esta convivencia de más de 4 años.

A mi jurado la Dra. Fabiola López y López y al Dr. Miguel Ángel León Chávez por su amabilidad y cooperación en el desarrollo de esta tesis.

Gracias a la Vicerrectoría de Estudios de Postgrado por el apoyo para la presentación de este trabajo a nivel internacional.

Al Consejo Nacional de Ciencia y Tecnología
por la beca otorgada para la culminación de
mis estudios de Postgrado.

Dedicatoria

A Daniel por su cariño, apoyo y
comprensión en las buenas y en las
malas.

A Rogelio y Alicia por su
paciencia.

A mis compañeros de maestría por
esta convivencia de 2 años.

Índice General

Índice de Figuras

Índice de Tablas

Introducción

Las computadoras no son lo suficientemente buenas para saber qué tienen que hacer, es decir, cada actividad debe ser anticipada, planeada y codificada. Sin embargo los sistemas actuales requieren ser inteligentes y autónomos para tomar decisiones por sí solos: estas entidades de software son conocidas como *agentes* (Wooldrige, 2002).

El origen de este paradigma computacional se remonta al comienzo de los años 80 cuando fueron introducidas técnicas de Inteligencia Artificial en la industria. Se planteó entonces la necesidad de añadir a los sistemas inteligentes, la capacidad de comunicación para compartir objetivos, tareas y conocimiento. Es hasta finales de la década de los 90 cuando la expansión de Internet le dio un impulso definitivo a la investigación y desarrollo basado en agentes.

"Un agente es una entidad con objetivos, los cuales alcanza al ejecutar una serie de acciones, posee un dominio de conocimiento y está situado en un entorno concreto." Con dichas características diversas áreas de investigación se han dado a la tarea de profundizar en el tema de los agentes, una de ellas mencionada anteriormente es la Inteligencia Artificial dicha disciplina estudia a los agentes como sistemas inteligentes aislados, para que operen en una gran variedad de ambientes y circunstancias (Weiss, 2000).

Este trabajo pone principal interés en los agentes móviles, los cuales son entidades de software que pueden moverse de un nodo a otro en una red. La historia de este tipo de sistemas se remonta desde la necesidad de intercambiar información entre nodos computacionales o entre nodos que comparten recursos como servidores o dispositivos de Entrada/Salida. Además, actualmente la información se encuentra en sistemas distribuidos, de gran magnitud, abiertos y heterogéneos (Kotz, et al, 1999). Es así como la comunicación entre entidades en un sistema de cómputo distribuido se vio apoyado por diferentes tipos de paradigmas, como el paso de mensajes y los llamados a procedimientos remotos (RPC).

La percepción de agente móvil cambió el concepto de movilidad de procesos ya que reducen el tráfico en la red y tienen la capacidad para manejar gran cantidad de datos (Horvat, 2000), (Kotz, et al, 1999). Las aplicaciones de agentes móviles más comunes que se han puesto

en marcha han sido en el comercio electrónico, voto electrónico, gestión de red, procesamiento paralelo, aplicaciones industriales y recuperación de información (Wooldridge, 2000), (Weis, 2000). De esta última disciplina se liga el objetivo de este trabajo de tesis.

El área de Recuperación de Información (RI) ha tomado gran auge entre los investigadores y no es para menos, cuando se tienen que diseñar nuevas técnicas que auxilien a los usuarios en los procesos de búsqueda. Actualmente uno de los contenedores más ricos en información es la Web, sin embargo su inminente crecimiento ha generado problemas al momento de recuperar información, debido principalmente a la temporalidad de los sitios, información repetitiva, diversidad de datos, falta de estandarización para representar dichos datos, entre otros (Castells, et al, 2001).

Las técnicas de RI y específicamente de generación automática de resúmenes presentan varios retos, uno de ellos es el idioma. Para nuestro caso el Español es una lengua rica en sinónimos, homónimos, etc., y existen muy pocas herramientas para nuestro idioma que realicen un análisis para la obtención de resumen, las pocas que se encuentran disponibles desgraciadamente no son de distribución libre, además que las versiones de prueba que ofrecen no realizan un correcto procesamiento del vocabulario, por lo que los resúmenes pudieran omitir ciertos términos importantes; un ejemplo de estos programas es la herramienta Copernic Summarizer [22].

La investigación para la generación automática de resúmenes centra su atención en la selección de oraciones relevantes de un documento, de esta manera el usuario adquiere mayor cantidad de elementos informativos en menor tiempo. Es así como el resumen se vuelve una representación formal del documento original, mediante la consulta a este compendio de oraciones el usuario toma la decisión de leer o no todo el documento.

Sin embargo, los usuarios cada vez requieren que las máquinas de búsqueda proporcionen resultados lo más apegado posible a su consulta. En ese aspecto han surgido las ontologías, las cuales han dado una nueva faceta al área de RI. Las aplicaciones de las ontologías desarrolladas van desde sistemas que recomiendan a los usuarios qué sitios acceder para obtener información (Middleton, et al, 2001), a sistemas que emplean a las ontologías para ampliar los términos de búsqueda (Alani, et al, 2003).

La propuesta de este trabajo de investigación utiliza agentes móviles que apliquen técnicas para obtener resúmenes de páginas web y que éstos estén apegados a las preferencias de los

usuarios. La necesidad de este tipo de aplicaciones se ve apoyada por tres principales razones: la primera es la cantidad de textos repetitivos que se encuentran en Internet, la segunda es que la información está ubicada en diversos puntos geográficos y su acceso no debe ser limitante para obtener dicho recurso, y por último los usuarios requieren de sistemas personalizados de acuerdo a sus preferencias.

Ante esta problemática el presente trabajo propone los siguientes objetivos:

Objetivo general:

Desarrollar un sistema de software basado en agentes móviles que recupere información apegada a los intereses de un usuario.

Objetivos particulares:

1. Generación mediante la metodología del Proceso Unificado de Desarrollo de Software de los modelos de análisis, diseño e implementación para reconocer los módulos que forman parte del sistema.
2. Diseño e implementación de una ontología de domino que permita recuperar datos importantes de acuerdo al nivel de un usuario.
3. Diseño e implementación de un algoritmo que recupere de extractos de documentos mediante la técnica del párrafo virtual y de la ontología.
4. Desarrollo de una estrategia de movilidad en los agentes.

Es decir, el agente móvil estará dotado de mecanismos que le permitan migrar por diferentes nodos para recuperar información. Para ello será necesario que cuente con técnicas de recuperación de información que le ayuden a analizar los documentos, construir resúmenes de dichos documentos de acuerdo al nivel de conocimiento de un usuario (básico, intermedio o avanzado), y una vez finalizada la clonación por diversos nodos, el agente móvil regresará con aquellos resúmenes relevantes.

Además este trabajo permitirá que a partir de la información extraída el usuario pueda ahorrar una considerable cantidad de tiempo y saber en que ubicación está realmente la información que él busca.

La teoría de agentes es una línea de investigación que se puede considerar una de las más actuales, y el hecho de vincularla con el área de Recuperación de Información la hace aun más atractiva, pues se han desarrollado muchas aplicaciones que recuperan información relevante en el idioma Inglés, pero no así en Español.

A través del trabajo previo a esta Tesis se ha tenido contacto con varias plataformas para el de desarrollo de agentes móviles, sin embargo muchas de ellas no son de distribución libre. Se trabajó con la herramienta ProActive, la cual es libre y que permite la implementación de Objetos Activos que son las unidades básicas de ejecución dentro de este middleware. La experiencia obtenida mediante el uso de ProActive (Leal, 2005) condujo a seleccionar JADE como herramienta para la implementación de los agentes. Pues los objetos activos presentan limitantes al momento de la migración. Por ello la plataforma que se empleará en este trabajo es JADE, es una herramienta desarrollada en Java y que ofrece diversas facilidades para la implementación de sistemas distribuidos.

La metodología para el seguimiento del desarrollo de los agentes móviles es el Proceso Unificado de Desarrollo de Software, que nos permitirá capturar los requerimientos del sistema, y modelar las vistas de diseño e implementación del mismo mediante diagramas del Lenguaje Unificado de Modelado (UML).

A continuación se describe de manera general el contenido final de este trabajo:

- El capítulo 1 dará una introducción al lector acerca de los aspectos teóricos referentes a la recuperación de información ontologías y agentes móviles.
- El capítulo 2 se enfoca al diseño; se describe mediante la metodología del proceso unificado de desarrollo de software los modelos de análisis y diseño. Se detallan la estrategia a seguir por el agente, la construcción de un extracto y de la ontología.
- El capítulo 3 muestra las herramientas empleadas, el diagrama de componentes y la implementación del sistema.
- En el capítulo 4 se describen los experimentos diseñados y las pruebas que se realizaron.
- En el último capítulo se hace un recuento de las aportaciones logradas con este trabajo, además de proponer algunas sugerencias de trabajos posteriores a éste.

Cabe mencionar que los resultados obtenidos en este trabajo han sido presentados en los siguientes eventos nacionales e internacionales:

1. Lorena Leal Bando, Darnes Vilariño Ayala, Fabiola López y López, *Diseño de Agentes Móviles para Generar Extractos de Documentos en ProActive*, Memorias del 2°

Congreso Nacional de Ciencias de la Computación, Noviembre 2004, pp. 287-292, ISBN 968 863 798 X.

2. Lorena Leal Bando, Darnes Vilariño Ayala, Fabiola López y López, *Agentes Móviles para Extractos de Documentos*, 6to. Congreso Estudiantil de Computación, Instituto Politécnico Nacional, Research on Computing Science 13, Mayo 2005, pp. 63-74, ISSN 1665-9899.

3. Lorena Leal Bando, Darnes Vilariño Ayala, Fabiola López y López, *Mobile Agents with ProActive for Document Extracts*, International Conference on Computational Intelligence for Modelling, Control & Automation, 28 - 30 Noviembre 2005, Viena Austria, ISBN 1740 88247 4, Publicación por IEEE.

4. Lorena Leal Bando, Darnes Vilariño Ayala, Fabiola López y López, *Mobile Agents with Different Informatin Retrieval Techniques*, Iberagents 2006 International Joint Conference Iberamia, Riberao Preto, Brasil, Octubre 27. 2006.

5. Lorena Leal Bando, Darnes Vilariño Ayala, Fabiola López y López, Héctor Jiménez Salazar, Josefa Somodevilla García, *Agente Móvil Dotado de Técnicas de Recuperación de Información que emplean Perfiles de Usuario*, Memorias del 4° Congreso Nacional de Ciencias de la Computación, Noviembre 2006, pp. 38- 43, ISBN 968 9182 30-7.

6. Leal Bando Lorena, Darnes Vilariño Ayala, Fabiola López y López, Héctor Jiménez Salazar, *The Virtual Paragraph as a Retrieval Information Technique Implanted in Mobile Agents*, CIMCA 2006, Sydney Australia, 28 Nov. – 1 Dic. 2006. ISBN: 0-7695-2731-0 Publicación por IEEE.

Capítulo 1

Marco teórico

1.1 Recuperación de Información

Uno de los tesoros más valiosos del hombre es la información, pero el potencial de este tesoro radica en la habilidad para realizar ciertas operaciones con ella como:

- Buscar la información necesaria.
- Comparar las diferentes fuentes, hacer inferencias lógicas y conclusiones.
- Manejar los textos, por ejemplo, traducirlos a otros idiomas.

En la actualidad para efectuar este tipo de tareas se ha hecho uso de la computadora por el simple hecho de que son capaces de procesar grandes volúmenes de información, sin embargo es omitido un pequeño detalle, la computadora no entiende los textos para su procesamiento, solamente cadenas de caracteres sin ningún sentido y no usa información útil para el juicio lógico. Ahora identificado el problema, el reto de los investigadores es lograr que las computadoras sean una herramienta eficiente en el proceso de recuperación de información.

El área de recuperación de información (RI a partir de este momento), se encarga principalmente del estudio de sistemas y técnicas para asignar índices, buscar y devolver datos valiosos a los usuarios, es decir, consiste en buscar documentos que exhiban un mayor parecido a la pregunta formulada por un usuario (Baeza-Yates, 1999).

Los sistemas de RI han evolucionado en la medida que los medios de almacenamiento y los recursos de información ofrecen mayores potencialidades, estos sistemas han evolucionado desde la automatización de catálogos para realizar simples búsquedas basadas en nombres, palabras claves hasta el reciente uso de técnicas de inteligencia artificial para dotar a los sistemas de cierto juicio para seleccionar información relevante (Baeza-Yates,et al, 1999); (Grossman, et al, 2004).

1.1.1 RI en la Web

La Web fue creada para el intercambio fácil de información, pero no para que las computadoras pudieran manipularla y comprenderla. Los lenguajes para representar esta información no expresan nada sobre su significado y dificultan su manipulación (Castells, et al, 2001). A pesar de que la Web incrementó el proceso de las comunicaciones y permitió la interacción entre individuos sin importar su localización geográfica su carácter descentralizado ha fomentado su crecimiento exponencial (Grossman, et al, 2004).

Dicho incremento desmesurado de la Web ha ocasionado serias limitaciones para el manejo, organización y recuperación de la información disponible, entre las cuales se citan las siguientes:

- Ineficiencia en la recuperación de información, ya que ésta debe limitarse a la búsqueda de palabras claves en documentos HTML.
- Enormes pérdidas de tiempo en procesos adicionales de clasificación y filtrado de la gran cantidad de información recuperada, este último proceso es realizado la mayoría de las veces por los usuarios debido a la incapacidad de las aplicaciones para filtrar la información obtenida.
- Pérdida de gran cantidad de información que existiendo en la red no llega hasta el usuario, debido a que las herramientas de búsqueda no tienen acceso a la totalidad de la información.
- Dificultad para la reutilización de información presente en la Red debido a la diversidad de formatos e idiomas utilizados en su publicación
- La falta de estandarización en la información disponible en la Web limita la capacidad de los motores de búsqueda para interpretarla y satisfacer los requerimientos de los usuarios.
- Diversidad en los tipos de datos que aloja la Web.

En el contexto de la Web, se puede definir el objetivo de la recuperación como la identificación de una o más referencias de páginas web que resulten relevantes para satisfacer una necesidad de información. La entidad final buscada puede variar mucho: direcciones, datos estadísticos, archivos de imágenes, referencias bibliográficas y documentos de texto completo.

Los programas que realizan la RI en Internet toman principalmente dos morfologías llamadas directorios y buscadores. La información que contienen los directorios normalmente está almacenada y organizada manualmente por expertos en una estructura jerárquica. Los usuarios recorren esta estructura para buscar la información que necesitan. Los buscadores en cambio usan programas que van recogiendo la información de la red y la organizan en una base de datos. A cada petición de información unos programas buscan en esta base de datos y crean un documento con los resultados de la búsqueda. En la tabla 1.1 se resumen las características de los directorios y buscadores.

Tabla 1.1 Características de Directorios y Buscadores. Para ambos casos se enmarcan las siguientes características: Descubrimiento de recursos, Representación del contenido del documento, Representación de la consulta y Presentación de los resultados.

	Descubrimiento de recursos	**Representación del contenido del documento**	**Representación de la consulta**	**Presentación de los resultados**
Directorios	La realizan personas	Clasificación manual	Implícita (mediante navegación por las categorías)	Páginas creadas previamente a la consulta. Poco exhaustivos, muy precisos
Buscadores	Principalmente de forma automática mediante robots	Indización automática	Explícita (mediante palabras clave o conceptos, operadores, delimitadores, etc)	Páginas creadas de forma dinámica para cada consulta. Muy exhaustivos, poco precisos.

La mayoría de buscadores suele guardar en sus bases de datos un resumen (abstract) o sumario (summary) de cada documento web. En muchos casos este resumen consiste meramente en las primeras *n* palabras de un documento, con lo que la última frase que se

recoge puede quedar cortada o incluso se corta la última palabra. Además muchas veces esta información carece de valor como representación del contenido del documento.

1.1.2 Generación Automática de Resúmenes y Resumen por Extracción

Por generación automática de resúmenes de texto se entiende el proceso por el cual se identifica la información sustancial proveniente de una fuente (o varias) para producir una versión abreviada destinada a un usuario particular (o grupo de usuarios) y una tarea (o tareas) (Bueno, 2005), (Salazar, 2004).

Las características del resumen son las siguientes:

- Claro
- Ordenado
- Expresa lo esencial
- Fiel por que refleja los elementos de mayor relevancia sin aportar ideas nuevas

Todo lo anterior podría significar ventajas para el lector, sin embargo la complejidad de un resumen radica en que, al menos teóricamente, sólo el autor del documento será capaz de resumirlo adecuadamente.

Debido a la proliferación de textos repetitivos el resumen centra su propósito fundamental en la economía de tiempo; por que ayuda al interesado a decidir si debe leer el texto completo; además de que proporciona la mayor cantidad de datos en menor tiempo (Bueno, 2005).

El resumen puede verse como una técnica de representación de un documento, de manera abreviada, que refleje el contenido más relevante del mismo. Sin embargo se han usado otras técnicas de representación como son: el modelo booleano, modelo vectorial, variaciones de ambos, heurísticas periodísticas y funciones de similitud (Téllez, 2005), (Bueno, 2005), (Leal, 2005), (Salazar, 2004) que por lo general representan a todo el documento.

Para finalizar diremos que el resumen es una versión abreviada de un texto y, por tanto un extracto es el conjunto de oraciones del documento con las que se construye un resumen

Los resúmenes por extracción actúan sobre uno o varios documentos, vistos como una colección de oraciones; de estas oraciones se extraen y presentan aquellas consideradas como importantes o que responden a determinados criterios.

1.2 Párrafo Virtual (PV)

El PV es una técnica de selección de oraciones significativas de un texto y es empleado en este trabajo para confeccionar un resumen de determinado documento, dicha técnica se apoya en el Punto de Transición (PT), el cual surge a partir de las observaciones de George Kinsley Zipf (Bueno, 2005), quién formuló la ley de frecuencias de palabras de un texto (Ley de Zipf), donde clasifica a los vocablos de un documento en las siguientes categorías:

- Palabras con mayor frecuencia absoluta, es decir, palabras cerradas (preposiciones, conjunciones, artículos, pronombres).
- Palabras con menor frecuencia, son aquellas que reflejan el estilo y riqueza del vocabulario.
- Palabras que aparecen en la zona media de la función de distribución de frecuencias y que son las que representan a los documentos.

El PT es la frecuencia de un término del texto que divide en dos a los términos de un vocabulario (en términos de alta y baja frecuencia). Esto significa que los términos más cercanos al PT, tanto de alta y baja frecuencia, juegan un papel muy importante ya que estos van a determinar de qué trata el documento, es decir lo relevante.

La fórmula usada para el cálculo del PT es la siguiente:

$$PT = \frac{\sqrt{1 + 8 * I_1} - 1}{2} \qquad \textbf{(1)}$$

Donde I_1 representa el número de palabras que tienen frecuencia 1. Booth (Bueno, 2005) derivó la ley de términos de baja frecuencia de la cual proviene la ecuación 1, sin embargo, se presenta un inconveniente con respecto a los documentos que son demasiado pequeños, ya que para este tipo de textos, el valor obtenido para el PT regularmente se encuentra fuera de las frecuencias obtenidas en su vocabulario. Por tanto, el PT puede ser obtenido por inspección descendente, eligiendo dentro del vocabulario el primer término con la frecuencia más baja que no se repita. A partir de este punto se toma un porcentaje del 25% de términos de alta y el 25% de baja frecuencia para obtener un rango, dichos términos conforman lo que se denomina el párrafo virtual.

1.3 Ontologías

Actualmente el desarrollo de ontologías para el área de recuperación de información ha tomado diversas facetas, desde sistemas que recomiendan a los usuarios qué sitios acceder para obtener información (Middleton, et al, 2001), así como sistemas que emplean a las ontologías para ampliar los términos de búsqueda (Alani, et al, 2003). Además es muy amplia la gama de aplicaciones pues han sido empleadas también para lenguajes de comunicación entre agentes, en la web semántica, microbiología, agronomía, entre otras (Bautista, et al,2005). Esto demuestra que su uso se ha proliferado a diversas áreas de conocimiento pues cada vez se requieren más sistemas que organicen y representen el conocimiento.

1.3.1 Tesauros y Ontologías

Un tesauro es una colección de palabras relacionadas que representan el conocimiento de un dominio. En un tesauro, cada palabra objetivo es introducida junto con una lista ordenada de términos relacionados. Por inspección, es posible apreciar qué términos relacionan a los diferentes sentidos de la palabra objetivo (Salazar, 2004).

Los tesauros se utilizan en sistemas de RI para ampliar las consultas, es decir, se busca por aquéllos términos relacionados del tesauro y de esta forma mejorar los resultados de búsqueda.

Por ejemplo en el tesauros de LIN para la palabra objetivo “estrella” se tienen los siguientes vecinos: “superestrella, jugador y actor”, mientras que otros vecinos más alejados son: “galaxia, sol, mundo y plantea” las cuales están relacionadas a otro sentido de “estrella”.

La estructura de la terminología de un tesauro está basada en relaciones de tipo jerárquicas (describen términos más generales o más específicos), así como sinónimos y homónimos.

Por otra parte, el término “ontología” proviene de la Filosofía pero en el área de la Inteligencia Artificial tiene otra connotación, pues las ontologías son teorías que representan el conocimiento relativo a cierto dominio (Bautista, 2005). Las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada; y esta conceptualización deber ser representada de manera formal, legible y utilizable por las computadoras.

Las ontologías han adquirido gran importancia, debido a que los sistemas actuales necesitan describir la semántica de un dominio y que de esta forma el humano lo entienda y las

computadoras lo procesen. La complejidad de migrar un tesauro a una ontología radica en la precisión de la semántica a partir de las relaciones existentes en los tesauros.

Las similitudes presentadas entre los tesauros y las ontologías son los siguientes:

- Se utilizan para catalogar y organizar recursos de información.
- Proporcionan una representación del conocimiento de un dominio.
- Utilizan jerarquías para agrupar términos en categorías.
- Son desarrolladas por expertos en el área de dominio a tratar.

Las diferencias principales se presentan en las ontologías pues estas son superiores a los tesauros por las siguientes razones.

- Representan el conocimiento en un nivel más alto de abstracción y descripción, con ello la semántica es más profunda.
- Son reusables.
- Permiten compartir y procesar el conocimiento.
- Poseen mayor número y variedad de relaciones entre los conceptos.
- En cuanto a la representación del conocimiento, los tesauros la presentan de manera limitada y no formalizada, por lo contrario de las ontologías que se realiza de manera explícita y formalizada.

1.3.2 Características de las Ontologías

Este apartado describirá lo referente al desarrollo de ontologías, desde sus componentes, clasificación, metodologías de desarrollo, ventajas y lenguajes de representación.

Los componentes de una ontología son los siguientes: (Bautista, 2005).

1. Conceptos: Son ideas básicas que se intentan formalizar, a los cuales también se llaman clases de objetos.
2. Relaciones: Representan la interacción entre los conceptos del dominio, las relaciones pueden ser:
    a. Taxonómicas: Son estructuras jerárquicas de sub o super conceptos.
    b. Asociativas:
        i. Nominativas: (nombres de conceptos)
        ii. Asociaciones de lugar: posiciones de un concepto respecto a otro.

iii. Asociaciones funcionales: describen una función entre los conceptos.
iv. Asociaciones causales.

3. Funciones: son casos especiales de relaciones donde se identifican elementos mediante el cálculo de una función.
4. Instancias: son elementos identificables que constituirán a los individuos concretos que representa la ontología.
5. Axiomas: Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología.

Existen diversos criterios para clasificar a las ontologías, entre ellas (Bautista, 2005):

- Dependencia de contexto
  - Ontologías de dominio
  - Ontologías generales
  - Ontologías genéricas
  - Ontologías de tareas
- Por sujeto de conceptualización
  - Ontologías de aplicación
  - Ontologías de dominio y genéricas
  - Ontologías de representación de conocimientos
  - Ontologías de dominio-tarea
- De acuerdo a la granularidad de la conceptualización
  - Ontologías terminológicas
  - Ontologías de información
  - Ontologías de modelado de conocimiento

Este trabajo de investigación hace referencia a las Ontologías de domino, la ontología proporciona el vocabulario necesario para describir un dominio en particular. Se incluyen los términos relacionados con: los objetos del dominio y sus componentes, un conjunto de verbos o frases que representan las actividades y procesos que se ejecutan en el dominio, relaciones y fórmulas que regulan o rigen el dominio.

Una de las metodologías para el desarrollo de ontologías es la propuesta por Unschold y King (Unschold, et al, 1995), a continuación se describe en los siguientes puntos.

- Identificar el propósito de la ontología.
- Construir la ontología.
    - Capturar la ontología: Se identifican los conceptos claves y relaciones del dominio.
    - Codificación: Representar el conocimiento adquirido en el paso anterior mediante un lenguaje formal.
    - Integrar ontologías existentes.
- Documentar

Uno de los lenguajes que han sido desarrollados para la construcción de ontologías es OWL (Web Ontology Language). En OWL, los elementos utilizados son: individuos (instancias), propiedades (relaciones) y clases. Existen, sin embargo, aplicaciones para la construcción de ontologías, como Protégé que genera código OWL, en las que existen diferencias sutiles, por ejemplo, sus elementos son casos (instancias), slots (relaciones) y clases.

Las ventajas que motivan al uso de las ontologías son los siguientes:

- Permiten compartir la interpretación de la estructura de la información entre personas o agentes, es decir, el establecer una ontología sobre un dominio permite que dos agentes puedan entenderse sin ambigüedad y sepan a que se refieren.
- Permiten reusar el conocimiento, pues al hacer una descripción de un dominio permite que ésta pueda ser usada por otras aplicaciones que necesiten tratar con ese conocimiento.
- Permiten analizar el conocimiento del dominio, ya que una vez que se tiene una especificación del conocimiento se puede analizar utilizando métodos formales.
- Separan el conocimiento del dominio del conocimiento operacional, pues permiten hacer independientes las técnicas y algoritmos para solucionar un problema del conocimiento concreto del problema.

Sin embargo se debe tener en claro que no existe un modo correcto y único para modelar un dominio, así como considerar que el desarrollo de una ontología es un proceso iterativo.

1.3.3 Ontologías en RI

La visión actual del creador de la Web, Tim Berners-Lee y de muchos otros investigadores, el que la información contenida en las páginas web se convierta en conocimiento, referenciando datos dentro de las páginas web a metadatos con un esquema común consensuado sobre algún dominio (Alani, et al, 2003);(Middleton, et al, 2001). De esta forma, se mejorará la búsqueda, ya que además de que la información es más precisa, se podrán realizar inferencias relacionadas con el contenido de una página y con los requerimientos de un usuario.

Es en este marco de referencia donde el conocimiento de la web necesita ser representado de forma legible por las computadoras, esté consensuado y reutilizable, y las ontologías tienen mucho que aportar en el área de recuperación de información y a la web semántica.

Una de las aplicaciones en las que ya se emplean ontologías es en Quickstep (Middleton, et al, 2001), el cual es un sistema que discretamente monitorea las búsquedas y preferencias de un usuario cuando éste examina diversas páginas web. Esta herramienta propone un algoritmo que clasifica los URL visitados y, conforma un perfil para cada usuario. El sistema posteriormente recomienda al usuario documentos de acuerdo al perfil que le fue confeccionado. La ontología utilizada por Quickstep es dmoz, una taxonomía de temas de ciencias de la computación. Los resultados de Quickstep fueron alentadores después de un periodo de prueba de mes y medio, ya que al hacer uso de una ontología en lugar de una simple lista de temas, incrementó la exactitud en los perfiles de usuario y las recomendaciones proporcionadas fueron de utilidad para los mismos.

Otro trabajo que se propone es Artequakt (Alani, et al, 2003), cuyo dominio es el de artistas, pues elaborará biografías de ellos. El objetivo de este sistema es emplear herramientas apoyadas en lenguaje natural que automáticamente extraigan y consoliden conocimiento proveniente de diversas páginas web. Posteriormente una ontología influirá en la información recuperada y realizará los extractos.

1.4 Teoría de Agentes

Actualmente el paradigma de agentes ha robado la atención de varios investigadores, y su colaboración ha proporcionado gran cantidad de literatura al respecto, incluyendo definiciones, clasificaciones, diseño de arquitecturas, metodologías, lenguajes y protocolos de comunicación. Existen también organizaciones que trabajan en la estandarización de sistemas de agentes, a su vez organizan y realizan eventos para congregar a más personas interesadas en el tema. Gracias a estas aportaciones se han desarrollado ya gran cantidad de aplicaciones abordadas mediante el paradigma de agentes, que han sido utilizados en control de procesos, comercio electrónico, herramientas de desarrollo, software de interfaces de usuario, manejo de redes y tareas de recuperación de información.

1.4.1 Agentes

Un agente es algo que actúa, (agente proviene del Latín *agere* que significa, hacer). Un agente es una entidad con objetivos, que alcanza ejecutando una serie de acciones, mediante un dominio de conocimiento y situado en un entorno concreto.

Wooldridge y Jennings definen a un agente como un sistema informático situado en un entorno y que es capaz de realizar acciones de forma autónoma para conseguir sus objetivos de diseño (Wooldrige, 2002). Además, es caracterizado por una serie de calificativos, los cuales denotan ciertas propiedades a cumplir.

- **Autonomía:** El agente percibe su entorno y puede operar por sí sólo sin la intervención de humanos u otros agentes.
- **Reactivo**: El agente es capaz de responder a cambios en el entorno en el que se encuentra situado.
- **Pro-activo**: El agente debe ser capaz de intentar cumplir sus propios planes u objetivos.
- **Social**: El agente debe de poder comunicarse con otros, mediante algún tipo de lenguaje de comunicación de agentes.

Sin embargo existen otras particularidades que se suelen atribuir a los agentes en mayor o menor grado para resolver problemas específicos, tales características han sido descritas por

autores tales como Franklin, Graesser y Nwana, algunas de ellas como la continuidad temporal, la racionalidad, la adaptatividad, la movilidad, la veracidad, la benevolencia, entre otras.

1.4.2 Agentes Móviles

Los agentes móviles, son programas o procesos que pueden moverse de un nodo a otro en una red (Kotz, et al, 1999). La historia de este tipo de sistemas se remonta desde la necesidad de intercambiar información entre nodos computacionales o entre nodos que comparten recursos como servidores o dispositivos de Entrada/Salida. Además, actualmente la información se encuentra en sistemas distribuidos, de gran magnitud, abiertos y heterogéneos por lo que los nodos no operan de manera solitaria. Por ello la complejidad para acceder a la información radica en los grandes volúmenes de datos, a su vez que estos se encuentran ubicados en distintos puntos geográficos (Bussetta, et al, 1998). Es así como la comunicación entre entidades en un sistema de cómputo distribuido se vio apoyado por diferentes tipos de paradigmas, como lo son el paso de mensajes, RPC y la arquitectura cliente servidor.

Sin embargo las siguientes características hacen que los agentes móviles sean atractivos para múltiples aplicaciones.

Los agentes móviles agregan dos cosas (Kotz, et al, 1999):

- Datos (datos recopilados y estados de procesos) y;
- Código (instrucciones que dirigen la conducta).

El agente se mueve de un nodo a otro, llevando sus datos y el código. Después de su llegada, continúa su ejecución donde éste se detuvo (no desde el inicio).

Un agente móvil debe ser hábil para ejecutarse en cualquier máquina dentro de una red, independiente del tipo de procesador o sistema operativo. El código del agente no debe haber sido instalado en cada máquina que el agente pudiera visitar potencialmente; este debe moverse con datos del agente (Leal, 2005).

Para lograr sus tareas y por razones de seguridad, los agentes móviles deben llevar datos acerca de ellos mismos y acerca de sus metas, es decir, no sólo su código, sino también su estado de ejecución e incluso algún recurso. Dadas estas necesidades, deberán cumplir los siguientes atributos (Horvat 2000), (Kotz, et al, 1999):

- Un identificador que permita distinguirle de forma unívoca en el sistema.
- Información referente al tiempo y lugar en que el agente fue creado.
- Un identificador del usuario o entidad propietaria del agente
- Conjunto de conductas o tareas a realizar por el agente y que marcarán el objetivo para el cual dicho agente ha sido creado.
- Conocimiento y líneas de razonamiento para el cumplimiento de sus tareas.
- Mensajes o resultados de las distintas tareas.
- Limitaciones impuestas al agente.
- Información de autenticación que le permita a la infraestructura por verificar su acceso y por consiguiente a los recursos.

Lo que diferencia a los agentes móviles del resto de paradigmas de código móvil, es que los agentes móviles son capaces de transportar el estado de ejecución en el que se encontraban en el momento de comenzar la migración.

Este paradigma resuelve una de las principales limitaciones de los paradigmas de código móvil, que es la dependencia de una conexión de red. Con los agentes móviles no es necesaria una conexión permanente por parte del cliente, mientras se completa el servicio (Kotz, et al, 1999). Utilizando agentes móviles el cliente, una vez lanzado el agente, puede desconectarse puesto que el agente posee la información necesaria para realizar el servicio, esperando a que el usuario se vuelva a conectar para devolver los resultados.

Las principales ventajas que los agentes móviles proporcionan son las siguientes (Bussetta, et al, 1998); (Kotz et al, 1999):

- Deben ser portable a través de plataformas.
- Capaces de elegir cuando y donde transportarse a sí mismos.
- Capaces de duplicarse a sí mismos.
- Se comunican con otros agentes para intercambiar información.
- Recolectan información en nombre de su dueño y regresan a casa después de ejecutar las tareas delegadas por su usuario.
- Aptos para suspender su ejecución, transportarse ellos mismos a otro nodo en la red, y de reanudar la ejecución desde el punto en el cual ellos fueron suspendidos.

- Consumen pocos recursos de red y pueden soportar sistemas que no tienen una permanente conexión a la red, tales como computadoras móviles.
- Son objetos que consisten de código, datos y estado de ejecución que puede ir más allá de dominios protegidos.

Sin embargo aún poseen algunas desventajas con respecto a:

- Autenticación: es un problema saber si los agentes son quién dicen ser y que representen a quién dicen.
- Seguridad: especialmente un agente móvil puede ser confundido con virus y para ello como protegerse contra ellos, además de garantizar que los agentes no consuman recursos indiscriminadamente.
- Privacidad: garantizar que el agente mantenga sus secretos y la información que lleva consigo no sea revelada.

A continuación se muestra el diseño de la aplicación desarrollada.

Capítulo 2

Análisis y Diseño del Sistema

Este capítulo está enfocado a describir los principales servicios que brinda el sistema, para ello se utiliza la metodología del Proceso Unificado de Desarrollo de Software (PUDS) y el lenguaje que se emplea es UML (Jacobson, et al, 2000). Dicha metodología está centrada en los diagramas de caso de uso, en una arquitectura y cabe resaltar que es un proceso iterativo. Para la fase de análisis se muestran los diagramas de caso de uso y diagrama general de clases, la fase de diseño incluye los diagramas de clase y de secuencia.

2.1 Modelo de Análisis

2.1.1 Análisis de Servicios y Diagramas de Casos de Uso

El agente móvil es el principal agente con el que cuenta el sistema, pues es el que tiene la habilidad de clonarse en diversos nodos y regresar con su usuario propietario a reportarle los resúmenes. Sin embargo para que algunas de sus actividades se lleven a cabo, éste solicita ayuda de un subagente[1], el cual hemos nombrado subagente ontológico, pues como su nombre indica, es el responsable de consultar la ontología y generar el TEB (Texto Enriquecido de Búsqueda) de cada usuario. En la tabla 2.1 se muestran los principales actores y los servicios que desencadena cada uno.

[1] Módulo con capacidades de comunicación y realización de trabajos autónomas.

Tabla 2.1 Actores y Servicios.

Actor	Servicios
Usuario	Activación de repositorios de Información.
Agente	Elaboración de resumen.
Agente	Visualización de resultados.
Sub Agente Ontológico	Consulta ontológica.

En el siguiente diagrama de casos de uso se identifican los actores y servicios del sistema, vea figura 2.1.

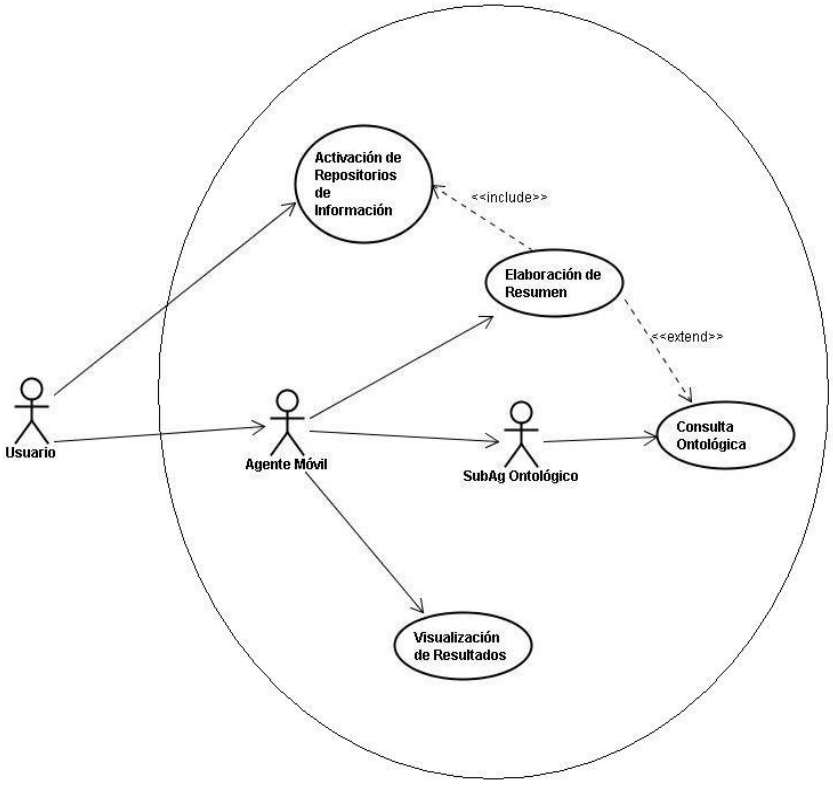

Figura 2.1 Diagrama de casos de uso de los servicios que ofrece el sistema. El usuario tiene acceso al servicio de activación de repositorios. Los servicios que el agente móvil provee son la elaboración del resumen y presentar los resultados obtenidos al usuario. Este agente interactúa con el subagente ontológico para obtener la consulta ontológica.

Cada servicio para que se lleve a cabo satisfactoriamente requiere que se cumplan los siguientes objetivos:

Servicio Activación de repositorios de Información:

- Identificar nodos potenciales.

Servicio Elaboración de resumen:

- Procesar la consulta y el nivel de un usuario.
- Solicitar TEB al subagente ontológico.
- Clonarse en los nodos activos. Este objetivo involucra:
    - Mover el clon hacia los nodos activos.
    - Autenticarse en cada nodo.
    - Se obtiene el nombre del clon y la máquina en la que se clonó.
- Confeccionar resumen. Este objetivo conlleva:
    - Seleccionar las páginas html.
    - Preprocesar los documentos.
    - Elaborar resumen mediante Punto de Transición y Texto Enriquecido de Búsqueda.
    - Elaborar mensaje (nombre de clon, nombre máquina, nombre del archivo, título, resumen).
- Enviar información recuperada al usuario.

Servicio Visualización de resultados:

- Desplegar la información al usuario.
- Almacenar el historial de consultas.

Servicio Consulta ontológica:

- Obtener la expansión de los términos de la consulta de acuerdo al nivel del usuario.

2.1.2 Arquitectura del Sistema

Los componentes principales que se han identificado para el desarrollo de este trabajo se ilustran en la figura 2.2. Los componentes se muestran en recuadros mientras que el intercambio de información se muestra mediante flechas en una o dos direcciones según sea el caso.

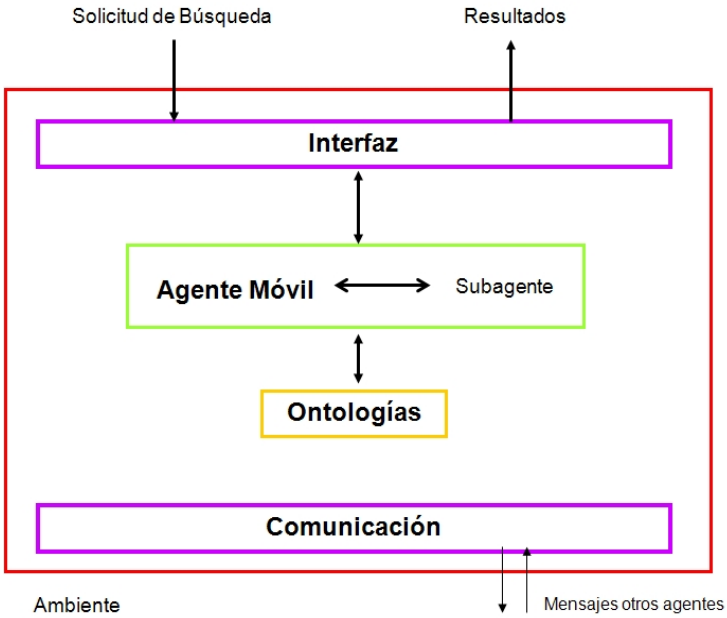

Figura 2.2 Arquitectura del Sistema

- Interfaz con el usuario: Es la encargada de recibir las solicitudes de búsqueda de un usuario, es decir, la palabra de consulta y el nivel de conocimiento del usuario acerca del tema en cuestión. Posteriormente esta información es enviada al agente móvil. La interfaz también es la encargada de mostrar los resultados obtenidos.
- Agente móvil: Tiene su propio comportamiento, es decir, reacciona a los diferentes mensajes que recibe. Tiene asociado un conjunto de metas que le permiten clonarse en diferentes nodos y con ello satisfacer las solicitudes de búsqueda de un usuario.
- Subagente ontológico: Auxilia al agente móvil por medio de la información que obtenga de la ontología de dominio.
- Ontologías: Permiten obtener a los agentes información sobre lo que los rodea. Se emplean ontologías para la comunicación y movilidad de los agentes. La ontología que maneja el subagente ontológico es para obtener términos del dominio relevantes para los usuarios. El conjunto de términos recuperados conforma el *Texto Enriquecido de Búsqueda* (TEB).
- Módulo de comunicación: Se encarga de enviar y recibir los mensajes entre los agentes mediante protocolos de transporte.

2.1.3 Diagrama General de Clases

La clase *MobileAgent* es la clase principal, pues es la que lleva el control de todas las actividades a ejecutar. Contiene métodos para llevar a cabo la clonación del agente, además de ejecutar determinadas tareas ya sea antes o después de la misma. Es la clase encargada de procesar los datos de entrada del usuario al igual que los métodos para mostrar la información recuperada, para llevar a cabo esta última tarea se auxilia de la clase *escritura*.

La clase *SubAgOntologico* requiere los datos procesados del nivel y de la consulta del usuario para su correcto funcionamiento. Agrupa a los métodos que realizan la búsqueda con base en la ontología desarrollada y obtiene una lista de términos relevantes al usuario.

La clase *Location* se enfoca al manejo de nodos, ya que para que el agente móvil pueda clonarse sobre los diferentes nodos.

Las clases destinadas para la interfaz del usuario son: *MobileAgentGui*, la cual despliega los elementos gráficos necesarios para que el usuario introduzca sus solicitudes de búsqueda. A través de esta misma interfaz le permite al usuario visualizar los resultados obtenidos. *LocationTableModel* es la clase que despliega sobre la interfaz los nodos activos y los nodos visitados. Ambas clases se ubican en el paquete GUI.

En el diagrama se ilustran tres paquetes más, *Gestión de Archivos*, en este paquete se agrupan tres clases necesarias para la búsqueda, la lectura y la selección de archivos relevantes, las clases son *BuscadorArchivos, AbrirArchivo* y *SeleccinaArchivos*. El paquete *PreProcesamiento,* incluye las clases *PreProcesamiento, RemoveTags y StopWords*, y como su nombre indica, son las clases especializadas para eliminar los tags y palabras cerradas de los documentos a procesar. El paquete *Punto Transición* en el que se albergan tres clases más, *ClaseArbolBin, ClaseArbolBinBus y PuntoTransicion*, son las clases que obtienen la frecuencia de términos y posteriormente calcular el valor para obtener el punto de transición.

En la figura 2.3 se ilustra la manera en la que se distribuyen y relacionan las clases y paquetes mencionadas en este apartado.

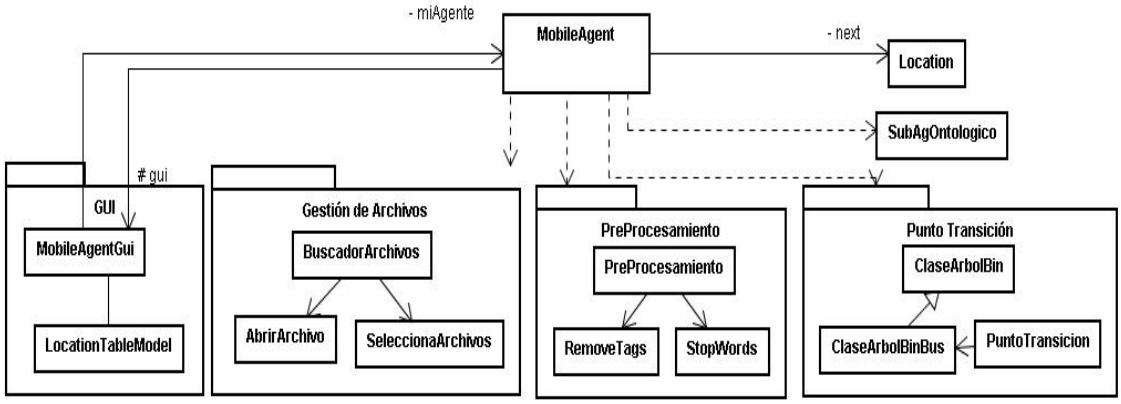

Figura 2.3 Diagrama general de clases del sistema. El diagrama presenta la interacción de la clase principal MobileAgent con los paquetes para la Interfaz, Gestión de archivos, Preprocesamiento de documentos y Punto de Transición. Además las clases para el manejo de nodos y subagente ontológico.

2.2 Modelo de Diseño

En este apartado se discutirá el diseño del agente móvil, el cual también abarca el diseño de la ontología y el diseño para la obtención de extractos mediante el PT y el TEB. De cada uno de ellos se mostrarán sus respectivos diagramas de clases y diagramas de secuencia.

2.2.1 Diseño del Agente Móvil

La clase *MobileAgent* tiene los siguientes atributos y métodos, los cuales se detallan a continuación. Vea la figura 2.4.

Las constantes `EXIT`, `MOVE_EVENT`, `CONTINUE_EVENT`, `REFRESH_EVENT`, `CLONE_EVENT` y `CONSULTA_EVENT`, son empleadas por la interfaz para enviar dichos eventos al agente.

Las variables estáticas `n_clo` y `ciclo` son las encargadas de llevar un control del número de agentes creados durante la ejecución.

Las variable respuesta, `query` y `profile` son empleadas para obtener la información proveniente de cada usuario, es decir, su consulta y nivel, respectivamente. Las variable `onto` recupera el TEB generado para cada usuario dependiendo de sus datos de entrada. El valor de

estas variables se obtiene una vez que la clase *MobileAgent* solicita la intervención de la clase *SubAgOntologico*.

Las variables `archivo, arch2` se encargan de obtener el nombre del archivo a procesar por el agente y de escribir los datos obtenidos de cada consulta de un usuario en un archivo diferente mediante la variable lineadevuelta.

El método `setup()` es un método que inicializa el lenguaje que emplearán los agentes, se debe registrar la ontología de movilidad, se crea y muestra la interfaz además de obtener la lista de nodos disponibles que se desplegarán en la interfaz.

`takeDown()` es el método que permite eliminar a los agentes una vez que han regresado de visitar un nodo.

`init()` es el método en el que se reciben las plantillas que sirven para ejecutar ciertos llamados, por ejemplo si un agente aun no ha migrado se obtiene una lista de los nodos activos dentro de la plataforma, posteriormente la almacena y la agrega por medio del método `addBehaviour` para que viaje con esta lista.

`beforeClone()`: es el método que se ejecuta antes de que el agente móvil migre, pues se recurre al subagente ontológico para que se obtenga el TEB de cada usuario.

`afterClone()`: El agente ya una vez inicializado los datos del usuario, realiza la búsqueda de archivos html en el nodo remoto, aplica la técnica de recuperación de resumen, construye y envía el mensaje en el que almacena la información obtenida, para posteriormente regresar con su usuario propietario.

Los métodos `beforeMove()` es empleado para verificar si el agente ya está listo para clonarse, y `afterMove()` emplea la clase *escritura* para que la información obtenida se guarde en un archivo diferente por cada consulta que realiza el usuario.

Los método `escribe_pagina(), crea_pag()` y `escons(),` son los métodos encargados de desempacar el mensaje enviado por los agentes, es decir, ordenan los datos de manera que sean entendibles al usuario.

El método `onGuiEvent()` recibe que evento fue ejecutado por el usuario, ya sea lanzar al agente, visualizar su información, actualizar la tabla de los nodos disponibles o bien salir de la aplicación.

`SendData()` es el método encargado de enviar los datos procesados del usuario al subagente ontológico.

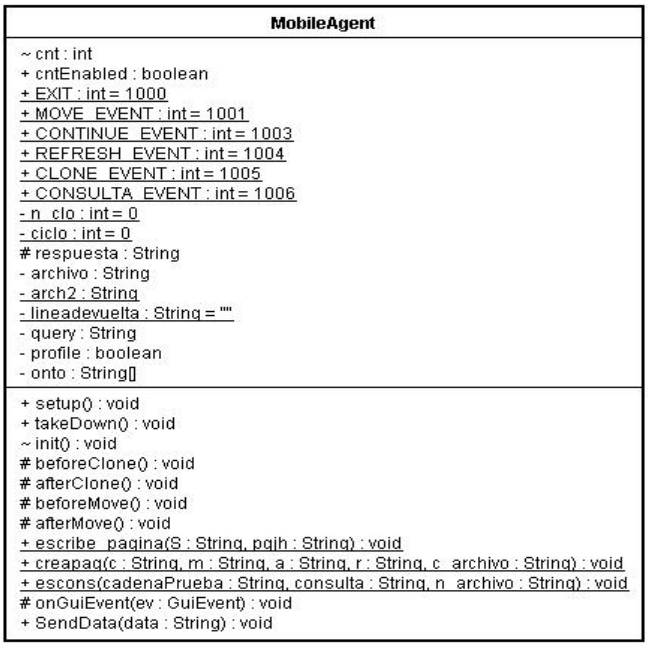

Figura 2.4 Diagrama de la clase MobileAgent. En dicho diagrama se ilustra si el atributo o método es privado (-), protegido (#) o bien público (+). También se ilustra el tipo de dato para el caso de los atributos y para los métodos si estos son tipo void() o requieren de algún parámetro para su funcionamiento.

La clase *SubAgOntologico* consta de tres métodos, el primero `processOnto()` realiza la expansión de términos para construir y enviar al agente móvil el TEB de cada usuario. La manera en la que se obtiene el segundo método `TEB()` se describe de manera detallada en el apartado 2.2.2**.** Se incluye un tercer método encargado de obtener los datos provenientes del usuario, es decir, su consulta y su nivel de conocimiento. En la figura 2.5 se ilustra el diagrama de clase la clase descrita.

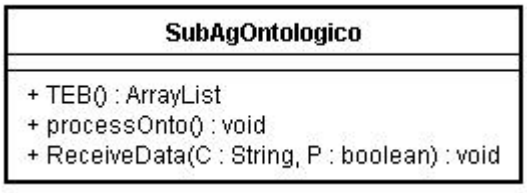

Figura 2.5 Diagrama de la clase SubAgOntologico. En dicho diagrama se ilustran los métodos públicos processOnto(), ReceiveData() y TEB(), este último regresa un conjunto de términos tipo String, los cuales son relevantes para cada usuario dependiendo de su consulta y nivel.

Por cada consulta que realiza un usuario, ésta se almacena en un archivo diferente, para ello es necesario llevar un registro del último archivo que se generó. El archivo contiene datos importantes del proceso de búsqueda, a continuación se muestran la información a almacenar.

- Nombre del clon: *Nombre con el que se autentica el agente una vez que migra a un nodo destino*
- Nombre de la máquina: *Nombre del nodo donde llegó el clon.*
- Consulta
- Nivel del usuario
- Nombre del archivo
- Título del documento
- Resumen del documento

Es importante llevar un registro del nombre del clon y de la máquina, pues así el usuario tiene conocimiento de los clones que se han generado y que nodos se han visitado.

La clase *escritura*, figura 2.6, implementa esta tarea mediante 6 métodos. Los métodos `escribe()` y `abrir(),` como sus nombres lo indican son los encargados de escribir y abrir el archivo que llevará la secuencia de archivos generados. De la misma manera `borrar()` y `crear(),` borran y crean el archivo que se indique en el archivo de secuencia, respectivamente. El método `lee()` realiza la lectura del archivo de secuencia para que el método `getNextFile()` analice el registro del último archivo y obtenga el nombre del siguiente archivo que se creará para almacenar la consulta actual. La variable archivo además está inicializada, de tal forma que por default los archivos serán almacenados en dicha ruta.

Figura 2.6 Diagrama de la clase escritura. Es la clase encargada de generar un nuevo archivo por cada consulta realizada por el usuario.

La clase *Location* se emplea para acceder a la información acerca de varios nodos a los cuales un agente móvil se clona. Posee cuatro métodos que emplea para obtener el ID, nombre, protocolo y dirección de un nodo.

El paquete *GUI* incluye las siguientes clases *MobileAgentGui* y *LocationTableModel*, veáse figura 2.7.

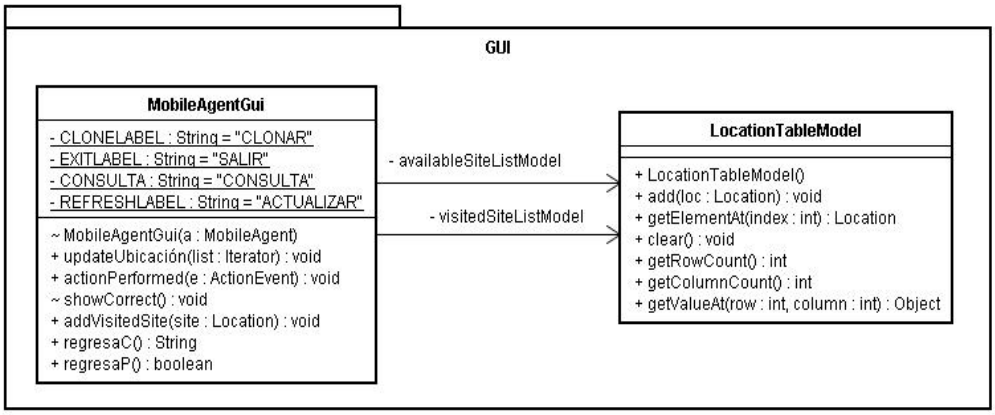

Figura 2.7 Paquete Gui. El diagrama ilustra los atributos y métodos de cada clase que interviene en dicho paquete, además de las relaciones presentadas entre ellas.

La clase *MobileAgentGu*i cuenta con los siguientes atributos y métodos.

Las variables `CLONELABEL, CONSULTALABEL, EXITLABEL` y `REFRESHLABEL` se emplean en la interfaz para indicar al usuario las funciones que puede realizar, es decir, clonar al agente, visualizar el historial de consultas, salir, y actualizar la tabla de nodos disponibles.

`updateLocations()` es le método encargado de actualizar la lista de nodos activos.

`regresaC()` y `regresaP()` son los métodos encargados de capturar consulta y nivel de un usuario.

`showCorrect()` despliega los elementos de la interfaz.

`addVisitedSite()` es el método mediante el cual se seleccionan los nodos a los que el agente móvil viajará.

La otra clase que interviene en el paquete *GUI* y que es utilizada por la clase *MobileAgentGui* es la clase *LocationTableModel*, figura 2.7, además del constructor, los métodos que se implementan son principalmente utilizados para agregar o eliminar nodos activos (`add()` y `clear()` respectivamente). `getElementAt()` se emplea para obtener el valor de cierto nodo seleccionado. `getColumnCount()` trabaja en conjunto con el método `getRowCount()` y `getValueAt(),` pues dependiendo del reglón y columna seleccionada se puede obtener, ya sea el ID de la máquina, nombre, protocolo o dirección.

Para que un agente ejecute alguna tarea se requiere de un objeto tipo behaviour, los comportamientos se pueden agregar en cualquier momento, cuando un agente comienza (en el método `setup()`) o dentro de otros comportamientos.

Cada clase que hereda de la clase *Behaviour* debe implementar los métodos `action()`, el cual define las operaciones a ser ejecutadas cuando el comportamiento está en ejecución. Por otro parte, el método `done()` específica si un comportamiento ha sido completado para removerlo de la pila de comportamientos que el agente todavía tiene que ejecutar.

El algoritmo de cómo se clonan los agentes móviles a los nodos se describe a continuación.

Entrada: Consulta *Q* y nivel del usuario *N*
Salida: Conjunto de Resúmenes *R*

1. Por medio de *Q* y *N* el subagente ontológico obtiene *TEB*.
2. Activar desde el contenedor principal a los contenedores remotos.
3. Para cada contenedor activo (es decir, esté registrado en la plataforma)
    a. Mientras haya páginas html *D* del dominio de aplicación
        i. Recuperar resumen, aplicar algoritmo sección 2.2.3.
4. Construir mensaje: [nombre de clon, nombre máquina, nombre del archivo, título, resumen].
5. Regresar al clon al contenedor principal con la información.
6. Crea fichero de almacenamiento con los resúmenes obtenidos.
7. Destruir la instancia del clon.

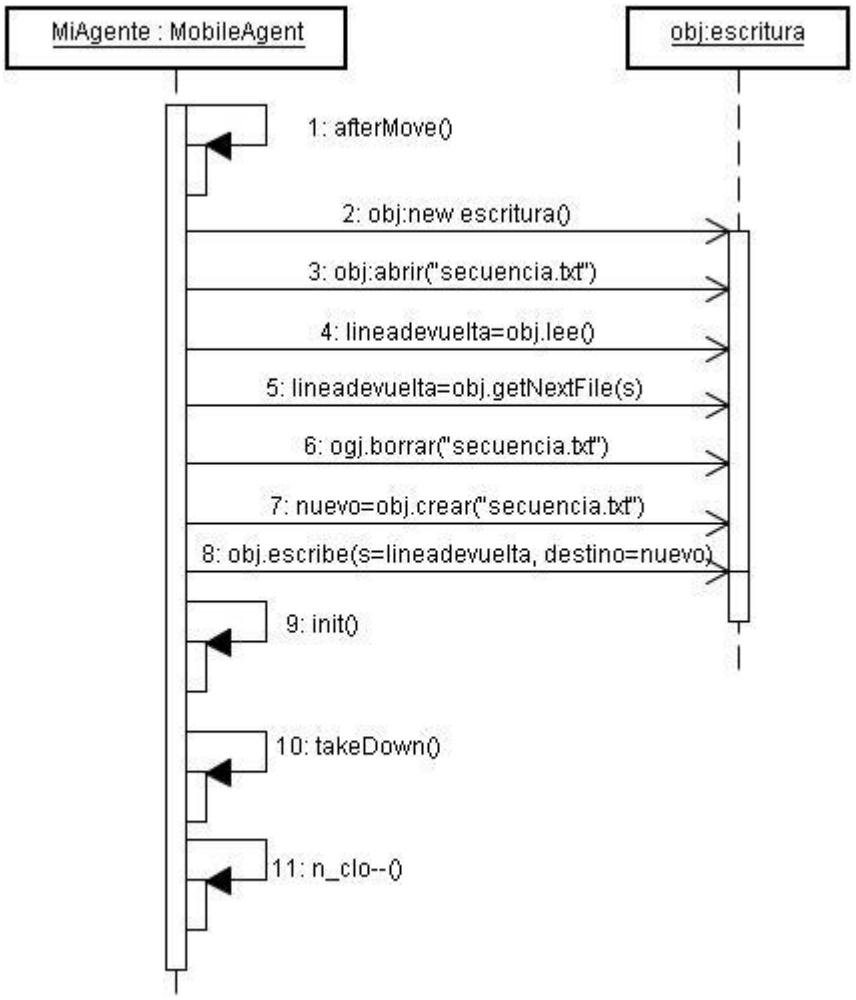

Figura 2.8 El diagrama de secuencia muestra el servicio de visualización de resultados**.** Una vez que el agente se ha clonado prepara los archivos para el almacenamiento del resumen actual. Posteriormente se elimina este clon.

Para el servicio relacionado con la activación de repositorios remotos, es necesario que haya una concentración más fuerte en el área de hardware que en la de software. Existen ciertos problemas como lo son: que el nodo destino esté apagado o desconectado de Internet, o que la plataforma no esté inicializada. Los dos primeros problemas serán solucionados de la manera tradicional, dado que los nodos no necesariamente estarían bajo el control del usuario; el problema de la inicialización de la plataforma si será solucionado mediante una propuesta que utilice RPC. Esta solución será descrita en el capítulo 3. Dado que el servicio no es creado junto con el sistema que se propone, no será modelado como parte del mismo.

2.2.2 Diseño de la Ontología

A partir la metodología propuesta por Unschold y King (Unschold, et al, 1995) se desarrolló la ontología en el dominio de la teoría de agentes. Dicha metodología sigue los siguientes pasos.

a) Identificar el propósito de la ontología

Recuperar términos de relevancia dentro de un dominio acorde a la consulta y nivel de conocimiento de un usuario. Dicho conjunto de términos es conocido como *Texto Enriquecido de Búsqueda (TEB),* denominado así pues refleja los requerimientos y el nivel del usuario.

Para ello el usuario tendrá asociado un nivel de conocimiento en el área, para tales efectos se proponen tres niveles. Los niveles son asociados a cada clase de la ontología y dicho criterio se basa en la complejidad del tópico o comprensión que requiera cada tópico por parte del usuario. La medida se asigna a través de un valor numérico de la siguiente manera:

Nivel de conocimiento	**Básico**	Grado de expertise de un usuario definido en un rango del 0 a .3
	Intermedio	Grado de expertise de un usuario definido dentro de un rango de .4 a .6
	Avanzado	Grado de expertise de un usuario definido dentro de un rango de .7 a 1

b) Construir la ontología

a) Identificar conceptos claves y relaciones del dominio: Los conceptos y sus respectivas definiciones se detallan en la tabla 2.2.

Tabla 2.2 Conceptos y relaciones de la ontología de agentes

Concepto	Definición
ACL	Lenguaje de alto nivel que utiliza los actos del habla para llevar a cabo la comunicación entre agentes.
Actos del Habla	Una directiva cuyo resultado es la comunicación entre dos o más agentes.
Áreas de aplicación	Es un software de aplicación que emplea tecnología de agentes.
Arquitectura	Refiere a las funciones principales y servicios que ofrece un sistema de agentes.
Autores	Investigadores involucrados en el área de agentes
Avanzado	Grado de conocimiento de un usuario definido dentro de un rango de .7 a 1
BioInformática	Utiliza tecnología de la información para organizar, analizar y distribuir información biológica con la finalidad de responder preguntas complejas en biología.
Básico	Grado de conocimiento de un usuario definido en un rango del 0 a .3
Comercio electrónico	Intercambio comercial de bienes y servicios realizado a través de agentes.
Comunicación	Intercambio y entendimiento de mensajes.
Control industrial	Aplicaciones industriales que emplean la teoría de agentes para resolver algún problema.
Cooperación	Interacción entre entidades para lograr un fin u obtener un bien.
Coordinación	Propiedad de un agente para no interferir en las actividades de otro.
Deliberativa	Arquitectura que emplea nociones mentales para su funcionamiento.
Desarrollo Formal de Agentes	Herramientas para el seguimiento en el desarrollo de agentes.
Domótica	Aplicación de la teoría de agentes, mecánica y electrónica para la automatización de las tareas domésticas.
Educación	Diferentes sistemas computacionales que tienen como finalidad apoyar el proceso de enseñanza y en particular de aprendizaje de los estudiantes.
Estándares	Familia de acuerdos que una comunidad adopta.
Formalismos	Propiedades que debe tener un agente y cómo representar y razonar formalmente esas propiedades.
Fundamentos agencia	Capacidad de una entidad a sensar su ambiente continuamente, tomar decisiones y actuar conforme a información de su entorno.
Fundamentos de Sistemas Multi – Agente	Es una colección de agentes de software que trabajan en conjunto unos con otros.
Híbrida	Arquitectura que combina módulos deliberativos y reactivos.
Implementación	Herramientas para el desarrollo de agentes.
Ingeniería de Software	Ingeniería de software orientada al desarrollo de agentes.
Interactiva	Arquitectura que emplea nociones conjuntas.
Interfaces Humanas	Interfaces hombre-máquina cuyo objetivo es mejorar la eficiencia, efectividad, y naturalidad de la interacción hombre máquina representando, razonando o actuando de acuerdo a una serie de modelos (usuario, dominio, tareas, discurso, contenidos, etc).
Intermedio	Grado de conocimiento de un usuario definido dentro de un rango de .4 a .6
Internet/Redes	Aplicación de la teoría de agentes en redes de computadoras e internet.
Lenguajes	Mecanismos de comunicación entre los agentes.
Lenguajes de Programación	Software con capacidades para implementar o simular el desarrollo de agentes utilizando lenguajes tradicionales de programación.

Negociación	Mecanismos de convencimiento para adquirir un bien.
Nivel agentes	Teoría básica de agentes.
Nivel sociedades de agentes	Fundamentos que definen el comportamiento de un agente en un entorno social.
Niveles de Conocimiento	Clasificación de los tipos de usuario.
Paradigmas	Patrón o modelo computacional a seguir para el desarrollo de agentes.
Protocolo Comunicación	Describe el lenguaje de comunicación entre agentes y el formato de los mensajes. Son las reglas que debe seguir un agente para comunicarse con los demás de manera coherente.
Protocolos Cooperación	Son las reglas que debe seguir un agente para ayudar a otros agentes.
Protocolos Coordinación	Son las reglas que debe seguir un agente para llevar a cabo sus actividades sin interferir en las de otros.
Protocolos Negociación	Patrón de interacción donde los agentes llegan a un acuerdo.
Reactiva	Arquitectura a base de estímulos y respuestas.
Recuperación de Información	Agentes que realizan la búsqueda de nuevas fuentes de información las cuales son inaccesibles para los usuarios.
Robótica	Aplicación de la teoría de agentes en diversas ramas de la robótica.
Salud	Aplicación de la teoría de agentes en las ciencias de la salud.
Simulación/Entre-tenimiento	Aplicación de la teoría de agentes para simular ambientes, comportamientos y entretenimiento.
Sistemas de Evaluación	Determinan analíticamente el nivel de la calidad del servicio que están dando los agentes.
Taxonomía	Clasificación de los diversos tipos de agentes según sus características y aplicaciones.
Tecnología de Agentes	Es el software, infraestructura, métodos y herramientas para el diseño, desarrollo e implementación de agentes.
Técnicas Cooperación	Conjunto de procedimientos que tienen como objetivo llevar a cabo la cooperación entre agentes.
Técnicas Coordinación	Conjunto de procedimientos que tienen como objetivo llevar a cabo la coordinación entre agentes.
Técnicas de Negociación	Conjunto de procedimientos que tienen como objetivo llevar a cabo la negociación entre agentes.

En el diagrama de clases UML de la figura 2.9 se hace referencia a las clases de la tabla 2.2 y a las relaciones que se describen a continuación. Las relaciones entre las clases identificadas son las siguientes:

La relación de herencia la conforman las clases Arquitectura, Fundamentos agencia y Taxonomía, dichas clases heredan de Nivel de agentes, para la clase Arquitectura se derivan otras cuatro clases: Deliberativa, Reactiva, Interactiva e Híbrida. De manera similar

Cooperación, Coordinación, Comunicación, Negociación y Fundamentos de SMA heredan de Nivel sociedades de agentes. De esta manera se plantea una jerarquización de clases.

Clases que no denotan herencia pero que de cierta manera se relacionan dentro del dominio y son clases de vital importancia dentro del dominio son: Autores, Niveles de Conocimiento y Tecnología de agentes. Sin embargo existe otro tipo de relaciones que se describen a continuación.

se clasifica en: para referirnos a los tres tipos de niveles de conocimiento.

tiene nivel: todas las clases poseen está relación, con al menos una clase de niveles de conocimiento.

definida por: relaciona a las diferentes arquitecturas, taxonomías y fundamentos de agencia con un autor(es) específico(s).

ejecuta: relaciona a las clases Cooperación, Coordinación y Negociación con determinada Técnica, lo mismo que la relación *contiene* que relaciona a dichas clases además de la clase Comunicación con sus respectivos protocolos.

requiere: para hacer énfasis en aquellas relaciones donde una clase solicita individuos de otra para llevar a cabo sus actividades. Tal es el caso de la clase Tecnología de agentes.

formado por e *incluye:* para indicar que ciertas clases son agregadas a otras, pues forman parte de ella.

c) Codificación

Representar el conocimiento adquirido en el paso anterior mediante un lenguaje formal. Este punto será descrito con mayor detalle en el siguiente capítulo.

d) Integrar ontologías existentes

Este paso fue omitido, pues no se tiene conocimiento de una ontología previamente desarrollada para este dominio.

e) Documentar

Este proceso se realizó mediante el modelado de la ontología en UML, puesto que de esta forma se tienen al alcance todos los conceptos involucrados, sus relaciones, así como sus respectivas definiciones.

Es importante destacar que para desarrollo de una ontología de dominio, ésta depende en gran parte de la visión y conocimiento que tengan los desarrolladores en el área. Pues pueden existir varias formas de abordar y representar determinado dominio.

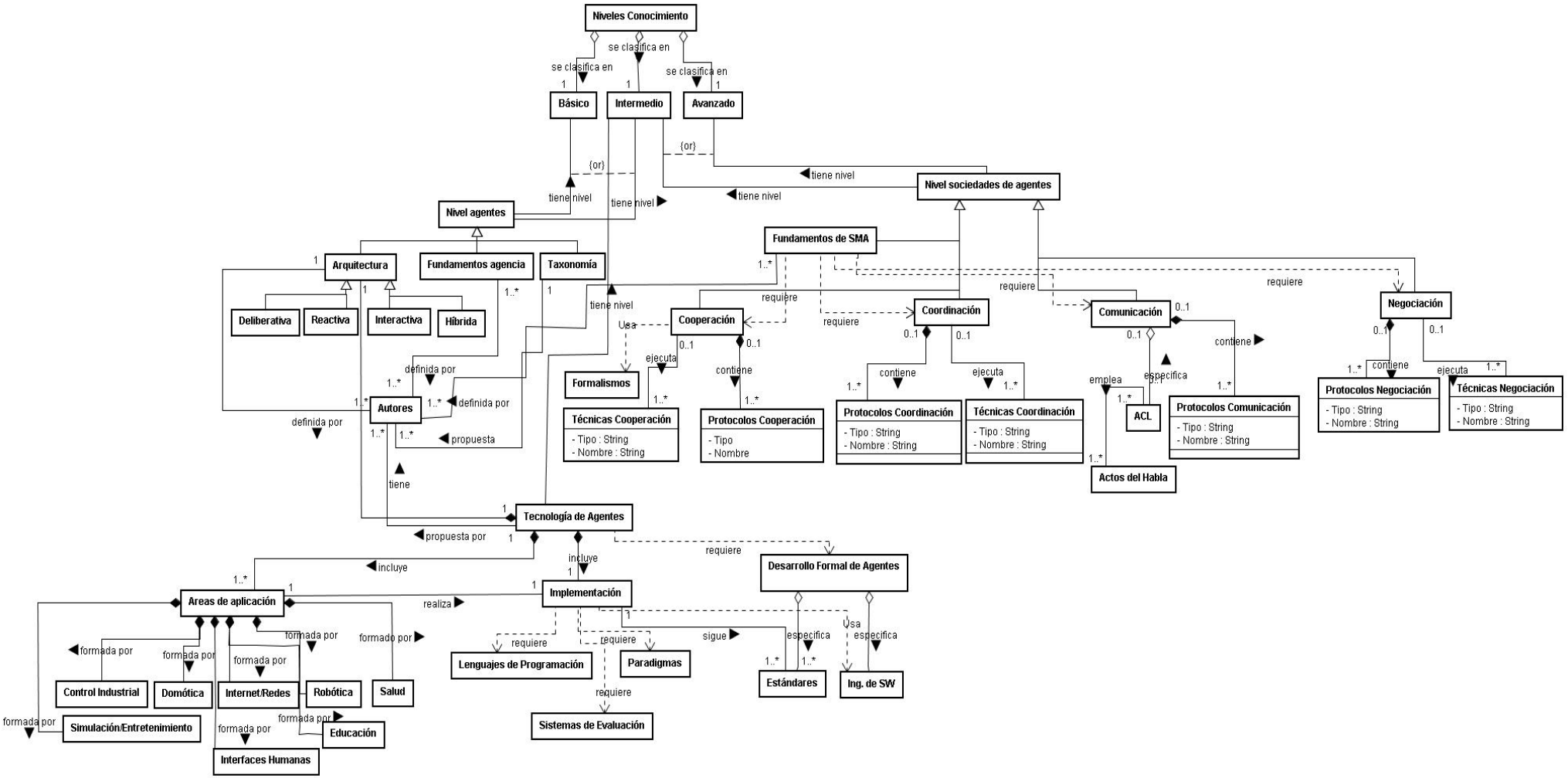

Figura 2.9 Diagrama de Clases de la Ontología de la Teoría de Agentes

Para nuestro caso en particular como se muestra en el diagrama de clases 2.9, para obtener el TEB, cada clase del dominio está asociada a un nivel de conocimiento. Esto es necesario para obtener términos representativos del dominio según las preferencias de los usuarios. Dichos términos se obtienen de la siguiente forma, una vez conocida la consulta y nivel de un usuario es necesario hacer un recorrido sobre la ontología, posteriormente se realiza una búsqueda de las clases que estén ligadas a la consulta y cumplan con el nivel indicado por el interesado. Las clases que cumplan con ambos requerimientos serán agregadas como términos al TEB. El límite de elementos para construir el TEB está limitado a 5 términos.

El algoritmo es el siguiente.

Entrada: Consulta del usuario *Q* y nivel del usuario *N*

Salida: TEB

1. Recorrer la ontología por el atributo "nombre de la clase"
2. Identificado el inicio de una clase
3. Analizar las relaciones y características de la clase

    Mientras el número de términos sea menor que 5

    Si se halló un término de interés

        Verificar si corresponde al nivel que indicó el usuario

            Si corresponde agregar la clase al TEB

            Incrementar contador de términos.
4. Regresar el conjunto de términos seleccionados, pues conforman el TEB.

En la figura 2.10 se ilustra el diagrama de secuencia mediante el cual se obtiene el TEB de un usuario una vez que este proporciona una palabra de consulta y su nivel.

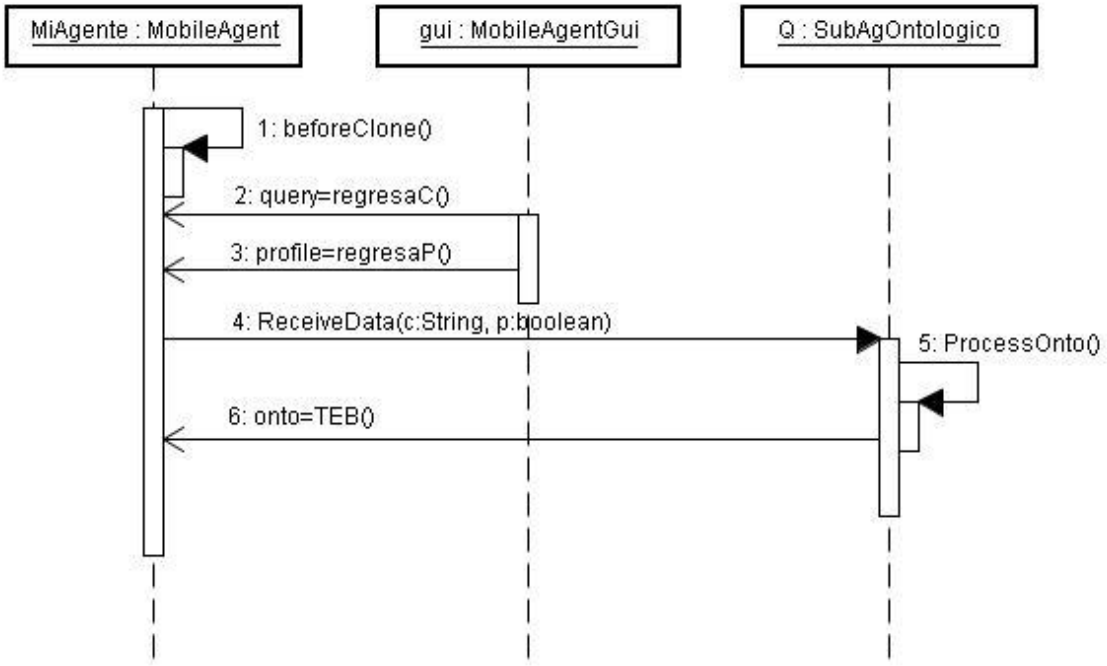

Figura 2.10 Diagrama de secuencia para la obtención de la consulta expandida del usuario. Para llevarse a cabo es necesario conocer y procesar los datos que el usuario proporciona, es decir, su consulta y su nivel.

2.2.3 Diseño para la Obtención de Extractos mediante el Punto de Transición y el Texto Enriquecido de Búsqueda

El proceso de generación del extracto de un texto se presenta a continuación:

Preprocesamiento: Se considera que un documento para nuestro caso es una página html, debido a esto se tienen que eliminar todos los tags, pues éstos no agregan información de utilidad para el usuario, y como tal no es parte de un vocabulario de interés para construir el PV. Se convierte a minúsculas el texto. Posteriormente se eliminan las palabras cerradas (artículos, preposiciones), y signos de puntuación, se realiza un tratamiento a las abreviaturas y por último el texto se particiona en oraciones.

Obtención del vocabulario: Se calcula la frecuencia de cada término del documento preprocesado.

Generación del párrafo virtual: Se aplica la técnica del punto de transición (PT) para obtener los términos significativos que se encuentran en una vecindad del 25% de términos de

frecuencias alrededor del PT. Se incluyen además los términos significativos obtenidos mediante la ontología y se obtiene el PV, es decir

$$PV=PT \cup TEB$$

Determinación de oraciones significativas: Se utiliza la función de similitud de Jaccard $sim(D,q)=\frac{\#(D\cap q)}{\#(D\cup q)}$. Con ella se compara cada oración del documento (D) con el PV obtenido a través de PT y del TEB (q). Los extractos de cada documento se conforman mediante las diez oraciones más relevantes.

A continuación se describen las clases involucradas y su respectivo diagrama.

Las siguientes clases realizan la gestión de archivos para su futura lectura y obtención del resumen sobre determinado documento. Las clases se agrupan en el paquete que lleva por nombre *Gestión de Archivos* y se muestra en la figura 2.11.

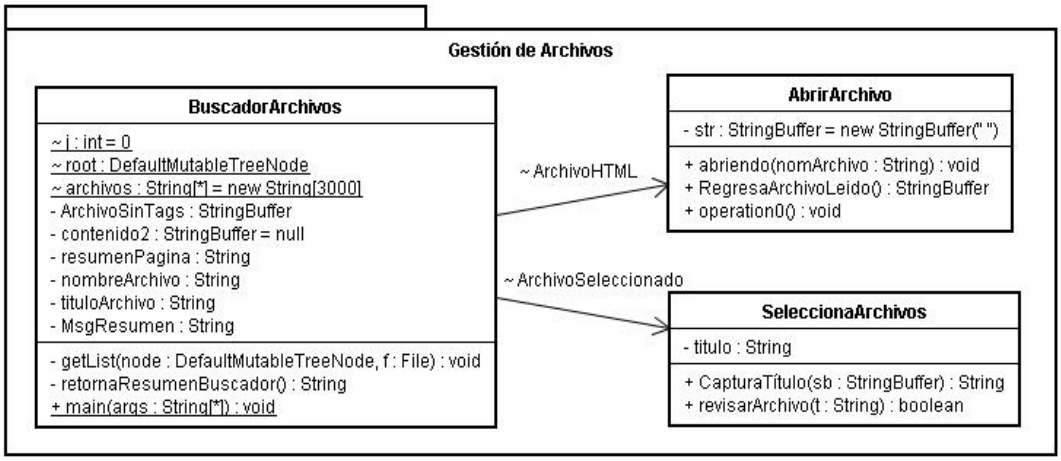

Figura 2.11 Paquete Gestión de Archivos. Incluye las clases *BuscadorArchivos*, *AbrirArchivo* y *SeleccionaArchivos*, además de las relaciones entre dichas clases. Para cada clase se despliegan sus respectivos atributos y métodos.

El paquete involucra las clases: *BuscadorArchivos,* esta clase es la encargada de realizar la búsqueda de todos los archivos html existentes en un nodo. *AbrirArchivos* la función principal de esta clase es de abrir el archivo en turno, además de retornar su contenido y *SeleccionaArchivos*, la clase selecciona archivos html que están relacionados con el dominio de los agentes.

El paquete *PreProcesamiento*, figura 2.12, incluye la clase *RemoveTags*, es la clase que realiza el primer preprocesamiento aplicado a un documento ya que elimina los tags de una

página html. *StopWords* es el segundo preprocesamiento, esta clase es la encargada de eliminar preposiciones, artículos y signos de puntuación, exceptuando el punto. Pues de igual forma que los tags, los stopwords no proporcionan un vocabulario de interés para el usuario. *PreProcesamiento*, es la clase realiza un análisis a las abreviaturas, segmenta el documento en oraciones y ordena las oraciones del resumen de mayor relevancia.

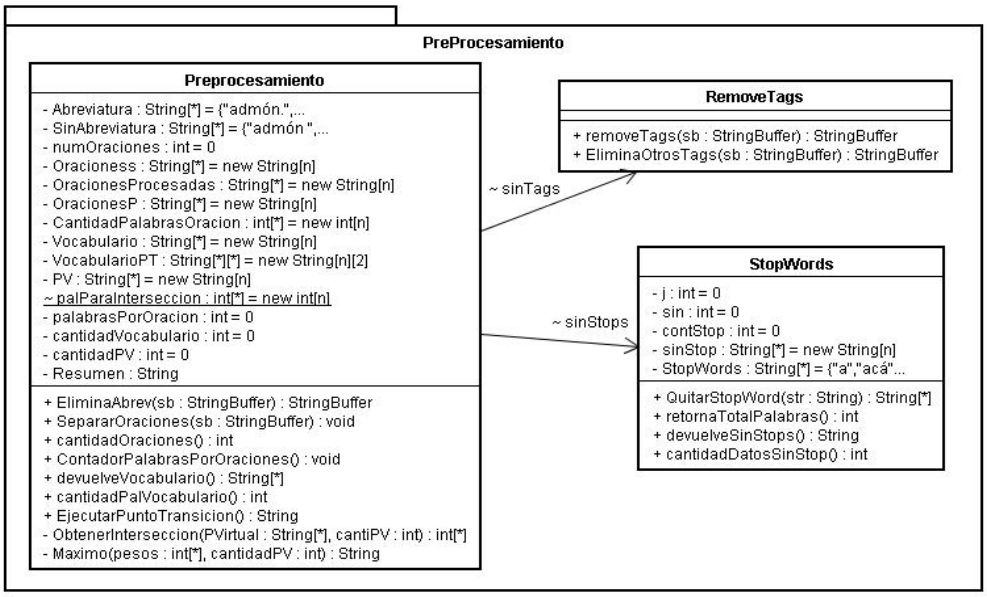

Figura 2.12 Paquete PreProcesamiento. Incluye las clases *Preprocesamiento*, *RemoveTags* y *StopWords,* además de sus relaciones. Para cada clase se despliegan sus respectivos atributos y métodos.

Las clases *ClaseArbolBin* y *ClaseArbolBinBus* del paquete *Punto Transició,*figura 2.13, almacenan el vocabulario de cada documento incluyendo la frecuencia de cada término, de esta forma la clase *PuntoTransicion* calcula el PT respectivo de cada documento, además considera la modalidad del PT para documentos demasiado pequeños. Posteriormente conforma el PV con los términos que están cercanos al PT y los términos del TEB.

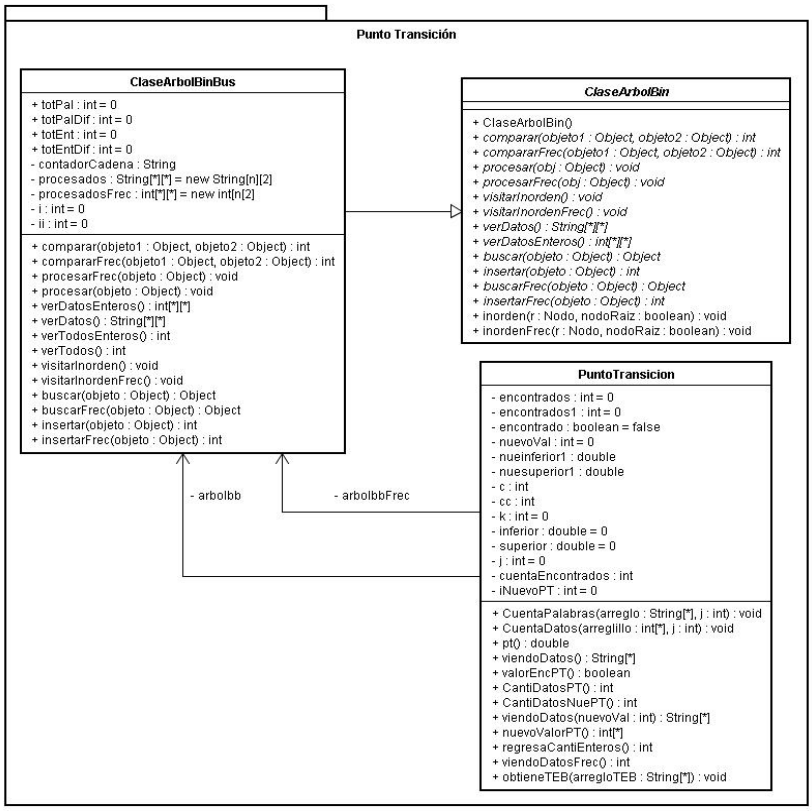

Figura 2.13 Paquete Punto Transicion. Incluye las clases *ClaseArbolBinBus*, *ClaseArbolBin* y *PuntoTransicion*, además de sus relaciones. Para cada clase se despliegan sus respectivos atributos y métodos.

El algoritmo para la obtención del resumen se muestra a continuación:

Entrada: Consulta del usuario *Q* y nivel del usuario *N*
Salida: Extracto *E*

1. Dado un documento *D* extraer mediante la ontología los términos relacionados con la consulta *Q* y nivel de usuario *N*, y con ello se obtiene *TEB*.
2. Para cada página web (*D*)

   Dividir el documento en oraciones $\{O_i\}_I$

   Se realiza el preprocesamiento

   Obtener el vocabulario *V* de *D*

   $V=[(t_1, f_1), \ldots, (t_n, f_n)]$, con $f_i \geq f_1 - 1$

   $PT = argmin_j \{f_{j-1} \mid f_j = f_{j+1}\}$

   $PV = \{t_k \mid f_k \in [PT*.75, PT*1.25], (t_k, f_k) \in V\}$
3. $PV = PV \cup TEB$

4. Para cada O_i de *D* obtener

   S_i=*similitud*(O_i, *PV*)

5. Tomar las 10 oraciones con mayor puntaje de acuerdo a: S_i: $E = \left|_{i \le S} \right. \{ O_i \mid S_i > S_{i+1} \}$

El siguiente diagrama de secuencia, figura 2.14, muestra el orden de los pasos a realizar para la obtención de resumen.

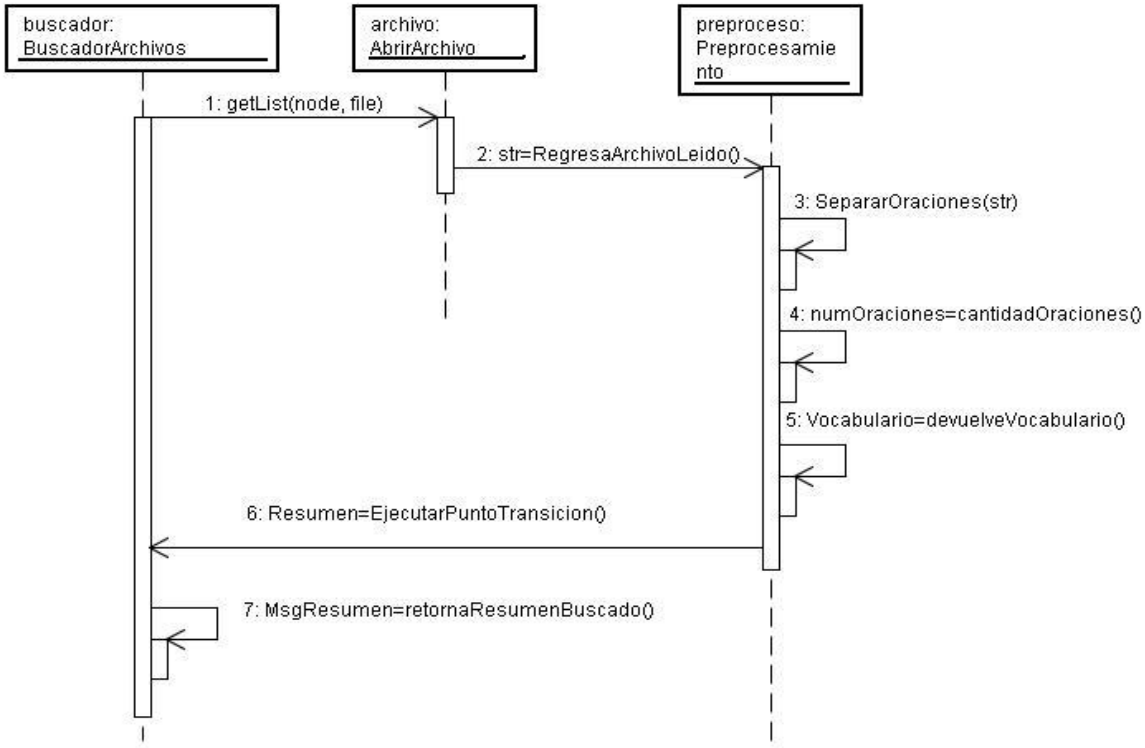

Figura 2.14 Diagrama de secuencia en el que se ilustra la interacción entre clases para que el agente obtenga el resumen de un documento.

Capítulo 3
Implementación

En este apartado se discutirán las herramientas que se emplearon para la implementación del agente móvil, ya que como lenguaje de programación se utilizó Java. También se incluyen los diagramas de componentes, además de detallar en cuanto a la implementación del agente móvil, la ontología de dominio y el algoritmo de obtención de resumen.

3.1 Herramientas de Implementación

3.1.1 JADE

Para el desarrollo de los agentes móviles se empleo JADE [20] (Java Agent DEvelopment Framework), es un middleware desarrollado en java. La plataforma JADE está basada en el concepto de "*contenedores*", es decir, donde viven los agentes. El conjunto de contenedores forman una "*plataforma*". De esta forma se provee al desarrollador de una capa homogénea para los agentes, sin que éste tenga que preocuparse por cuestiones de hardware, sistema operativo, tipo de red o JVM. JADE incluye:

- Un ambiente de ejecución, donde los agentes de JADE residen y el cual debe ser activado antes de que uno o más agentes puedan ser ejecutados en determinado nodo.
- Una librería de clases para los programadores.
- Herramientas gráficas que permiten la administración, depuración y monitoreo de las actividades que los agentes realizan durante el ciclo de vida de la aplicación.

La justificación por la que se decidió emplear JADE está dada en los siguientes puntos:

- Proactividad: Los agentes de JADE controlan su propio hilo de ejecución y, por lo tanto, ellos pueden programarse para comenzar la ejecución de acciones sin la intervención humana sólo en base a un objetivo y cambios de estado.

- Desarrollo de aplicaciones distribuidas compuestas de entidades autónomas además de esconder la complejidad de su arquitectura.
- Incluye librerías para la movilidad de agentes.

Es importante comprender la plataforma de desarrollo para la implementación de nuestra aplicación, además de las librerías que ofrece para la movilidad, para ello se explicará brevemente el funcionamiento de JADE.

Los principales componentes de JADE son los agentes, los contenedores y la plataforma. Vea figura 3.1.

- Los contenedores tienen uno o más agentes, cada agente es identificado con un único nombre y pueden comunicarse independientemente de su localización.
- La plataforma tiene un contenedor principal (Main container, el cual es el primer contenedor que se activa) y uno o más contenedores secundarios (Container).
- Los contenedores están distribuidos en uno o más nodos a través de la red de computadoras.
- La JVM es capaz de soportar en un solo nodo uno o más contenedores, cada contenedor principal tiene su propio ambiente de ejecución y servicios para uno o más agentes.

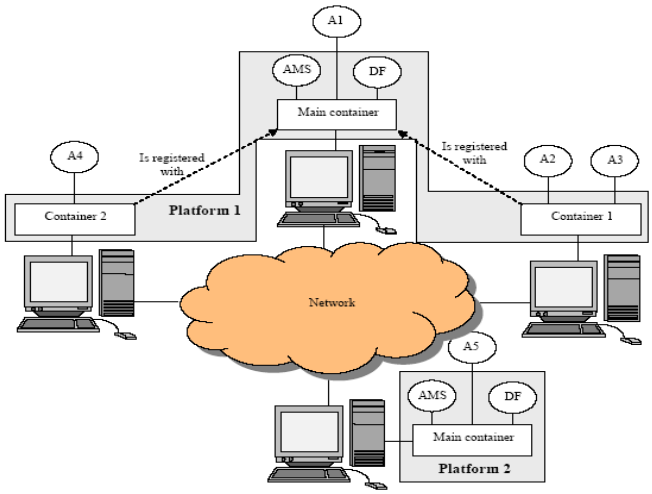

Figura 3.1 Diagrama donde se ilustra la plataforma JADE con un contenedor principal y dos secundarios. Además de una segunda plataforma con un contenedor principal.

- En el contenedor principal residen tres agentes especiales, Figura 3.2, que se inician automáticamente cuando un contenedor principal es activado.

- AMS (Agent Management System), es el encargado de proveer el servicio de nombres a los agentes, representa la máxima autoridad dentro de la plataforma pues tiene la facultad de crear y matar a agentes o contenedores remotos, administra los servicios de páginas blancas y ciclo de vida, mantiene el directorio de los identificadores de agentes (AID Agent Identifier) y su estado.
- DF (Directory Facilitator) proporciona las páginas amarillas, es decir, un agente puede solicitar servicios que alguno de los agentes de la plataforma provee, con el fin de lograr su objetivos.
- ACC (Agent Communication Chanel o Message Transport System), es el software que controla el intercambio de mensajes dentro de la plataforma.

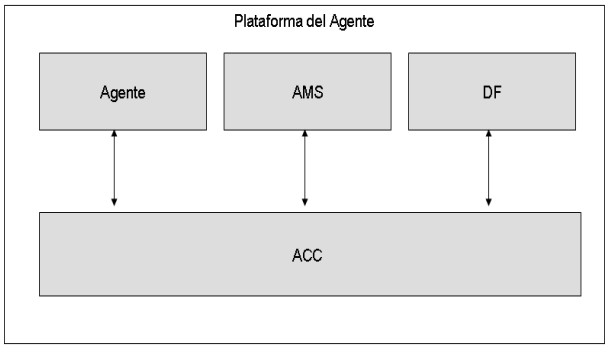

Figura 3.2 Diagrama que ilustra los agentes AMS, DF y ACC. Además de un agente nuevo que se creo, también se ilustra como ACC interactúa con esos tres agentes para el intercambio de mensajes.

La movilidad de un agente es la habilidad para que este migre o haga una copia de sí mismo (clonarse) a través de uno o múltiples nodos de una red. JADE soporta movilidad de código y estado de ejecución. Es decir, un agente puede dejar de ejecutarse sobre una máquina, migrando sobre diferentes nodos remotos, lo que involucra una transición de estados en el ciclo de vida de un agente (sin la necesidad de tener el código del agente instalado en ese nodo), y reiniciar su ejecución en el punto en que fue interrumpido (realmente, JADE implementa una forma de movilidad no tan débil, porque la pila y el registro del programa no pueden ser salvado en Java). Esta funcionalidad permite, por ejemplo, distribuir la carga computacional al tiempo de ejecutar los movimientos del agente, a las máquinas menos cargadas sin algún impacto en la aplicación.

La movilidad puede iniciarse por el propio agente o por el AMS. Para que el agente conozca su localización para decidir cuando y donde desplazarse, JADE ha desarrollado una ontología (jade.mobility.ontology) con conceptos y acciones necesarias (jade.domain.mobility).

Se ha estudiado a detalle la sección de movilidad de JADE, específicamente la clase Agent contiene los métodos para llevar a cabo el proceso de movilidad, estos son `doMove()`y `afterClone()`.

Sin embargo los métodos `doMove()` y `doClone()` requieren de ciertos parámetros, es decir, el lugar hacia el cual el agente creará un objeto tipo Jade.core.Location, el cual representa el destino requerido donde debe migrar el agente para el método. Por su parte el método `doClone()`lleva un parámetro extra el cual indica el nombre del nuevo agente que se creará con la copia del original indicado de la siguiente manera: `doClone(Location,String)`.

Mover un agente implica enviar su código y estado a través de la red (proceso de serialización y deserialización), algunos de los recursos usados por el agente móvil se moverán con él, mientras que otros serán desconectados antes de salir y reconectaos en el destino. JADE proporciona una serie de métodos para la gestión de recursos, estos son:

- `beforeMove()`, se invoca en la localización de partida antes de enviar al agente a través de la red.
- `afterMove()`, se invoca en la localización de destino tan pronto como el agente llega y se identifica.
- `beforeClone()` y `afterClone()` tiene la misma funcionalidad a los anteriores para el caso de realizar un proceso de clonación.

3.1.2 Protégé

El desarrollo más reciente en lenguajes estándar para ontologías es OWL del consorcio W3C (World Wide Web Consortium). En lo que respecta a la implementación de la ontología se utilizó **Protégé** [21], el cual proporciona el plugin Protégé-OWL que hace posible describir conceptos, pero también provee de nuevas facilidades para el desarrollo de ontologías. Contiene un rico conjunto de operadores por ejemplo *AND*, *OR* Y *NEGACIÓN*. Además el

modelo lógico en el que se basa, permite el uso de un razonador el cual puede verificar si todas las sentencias y definiciones de la ontología son mutuamente consistentes y también puede reconocer que conceptos concuerdan bajo que definiciones. El razonador de igual manera ayuda a mantener correctamente la jerarquía de conceptos.

Las ontologías OWL tienen componentes similares a las ontologías desarrolladas en Protégé, sin embargo la terminología empleada para describir los componentes varía un poco. Una ontología OWL se forma de Individuos, Propiedades y Clases; lo que en Protégé se designan como Instancias, Slots y Clases respectivamente.

Los individuos o instancias representan objetos del dominio en que se está interesado. Las propiedades o slots son relaciones binarias sobre los individuos, pues son las encargadas de ligar a los mismos. Por su parte las clases son una representación concreta de conceptos.

En la figura 3.3 se ilustra la interfaz de desarrollo de la herramienta donde se introducen los componentes de una ontología empleando la herramienta Protégé.

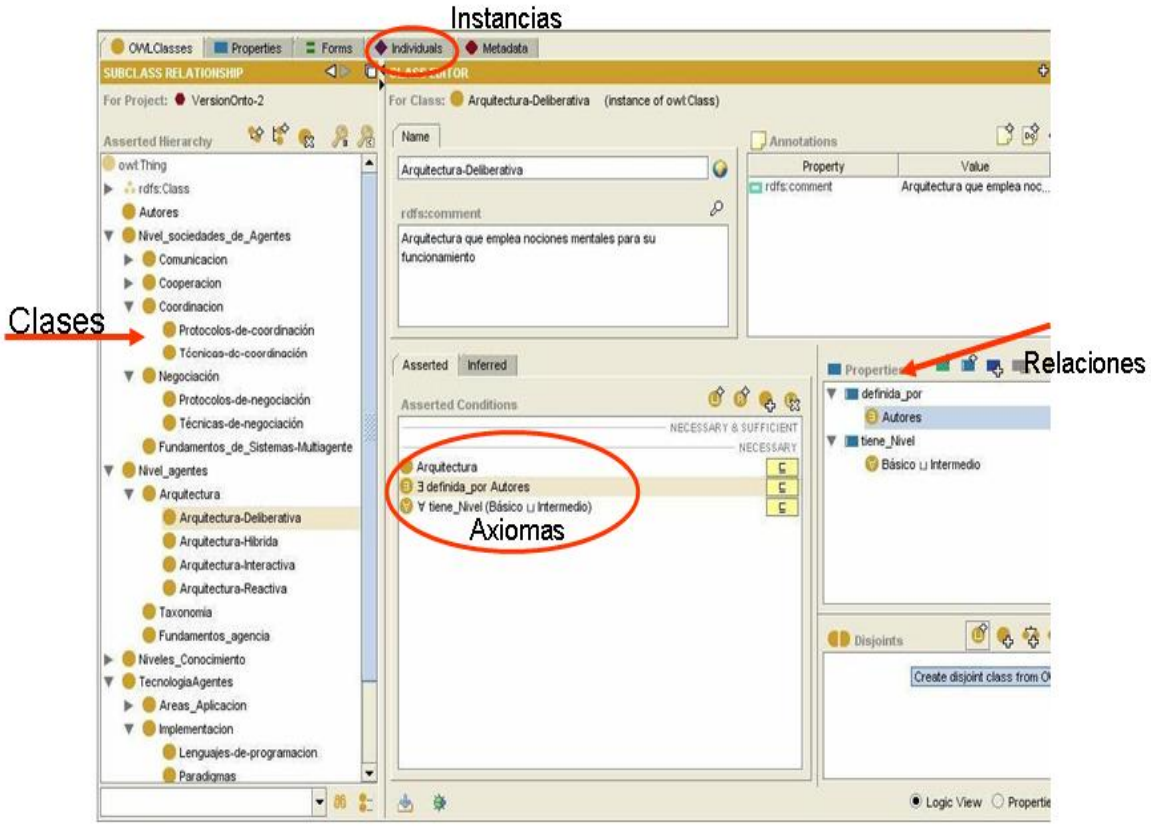

Figura 3.3 Interfaz de desarrollo de la plataforma Protégé. Se ilustra donde se capturan los elementos de una ontología, es decir, en que sección de dicha interfaz se introducen las clases, relaciones, axiomas e instancias.

3.2 Diagrama de Componentes

El diagrama de componentes tiene como función identificar aquellos elementos reutilizables de la aplicación y muestra la parte física del sistema. Compacta el código que implementa el sistema e ilustra las estructuras de almacenamiento empleadas.

Los componentes principales de la aplicación se ilustran en la figura 3.4

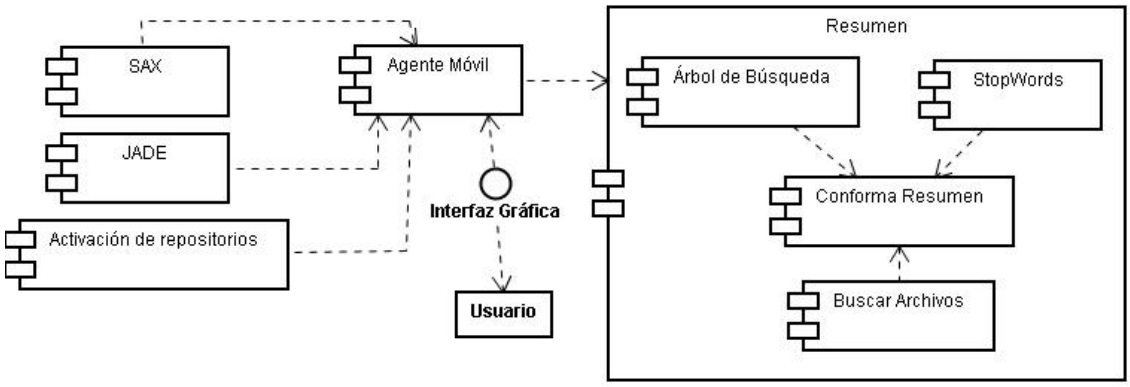

Figura 3.4 Diagrama de componentes del sistema. Los componentes que se visualizan son: Agente Móvil encargado de la movilidad, los componentes SAX y JADE. También se ilustra al usuario como actor que interactúa con la interfaz gráfica, la cual envía la información capturada al componente Agente Móvil.

El componente *AgenteMóvil*, se ve como un todo, que interactúa con la interfaz del usuario y el componente Resumen. La interfaz puede ser modificada según los requerimientos del usuario, los datos capturados se envían al agente móvil para que éste los procese. El componente Resumen es una tarea a realizar por el agente móvil una vez que llega a un nodo remoto, sin embargo ésta puede cambiar a otra tarea que no sea elaborar resúmenes. La implementación de está última no afecta el comportamiento de clonación del agente móvil.

El componente Resumen, conforma de otros 4 componentes que se describirán a continuación. El componente llamado "*Árbol de Búsqueda*" está asociado a las clases *ClaseArbolBin* y *ClaseArbolBinBus,* dichas clases implementan un árbol binario de búsqueda para llevar el registro y conteo del vocabulario de un documento. Las facilidades que brinda son el almacenamiento en orden alfabético de los términos, la frecuencia de cada uno de ellos y su recorrido agiliza la búsqueda de términos que se seleccionan para conformar el PV. El árbol acepta dos tipos de datos, numéricos y cadenas. Esta estructura arbórea facilita la elaboración del PV, pues considera si el documento es grande o pequeño. Esta última implicación varía

dependiendo si puede construir el PV mediante el cálculo inicial del PT o bien, si se tiene que ordenar descendentemente el vocabulario y calcular un nuevo PT.

El componente "*StopWords*" está asociado a la clase que lleva el mismo nombre, si cambia el idioma para elaborar resúmenes, simplemente se actualizan las palabras cerradas para el idioma que se requiera. Por el momento el objetivo es elaborar resúmenes en español, por lo que contiene preposiciones, artículos, pronombres, entre otros, más empleados en el español.

El componente "*BuscarArchivos*" tiene la función implementar un buscador de archivos html en cada nodo, además de seleccionar aquellos que sólo son de interés al usuario. Los criterios de selección se pueden modificar en este componente, sin que afecten a la obtención del resumen.

El componente "*ConformaResumen*" tiene interacción con los 3 componentes anteriores, pues los datos que envía cada uno de ellos repercuten en la elección de oraciones principales. Este componente lleva a cabo la implementación del cálculo de la función de Jaccard para obtener la similitud del PV contra cada oración. Se puede modificar la cantidad de oraciones con las que se conformará el resumen.

Los componentes restantes (SAX, JADE y la Activación de nodos remotos) no son implementados en el sistema de software, más bien se toman ciertos atributos y métodos de los mismos para cubrir algunas de las tareas que realiza el sistema de software.

3.3 Implementación del Agente Móvil

Para que el usuario ejecute al agente móvil se requiere que introduzca la siguiente sentencia en la que se debe de incluir la ejecución de la plataforma Jade, el nombre del agente, la ruta o carpetas donde se encuentra el código fuente del agente y por último el archivo .class que contiene al agente móvil.

```
java jade.Boot nombreAgente:CarpetaCódigoFuente.ClaseAgente
```

Por ejemplo:

```
java jade.Boot AgenteMovil:mobil.MobileAgent
```

Una forma para evitar teclear lo anterior es introducir la sentencia en un fichero .bat.

Para activar los nodos remotos fue necesario ejecutar un demonio rsh, y que éste activara contenedores Jade en dicho nodo. Es necesario indicarle al demonio que va a iniciar un contenedor remoto el cual a su vez debe conectarse al nodo principal donde reside el usuario:

```
java jade.Boot –host IPdestino –container
```

La interfaz gráfica que se propuso es la siguiente, vea figura 3.5: se tiene dos paneles para visualizar los nodos disponibles y visitados durante la ejecución, además se proporcionan los campos necesarios para que el usuario introduzca sus requerimientos, lance al agente y posteriormente visualice los datos.

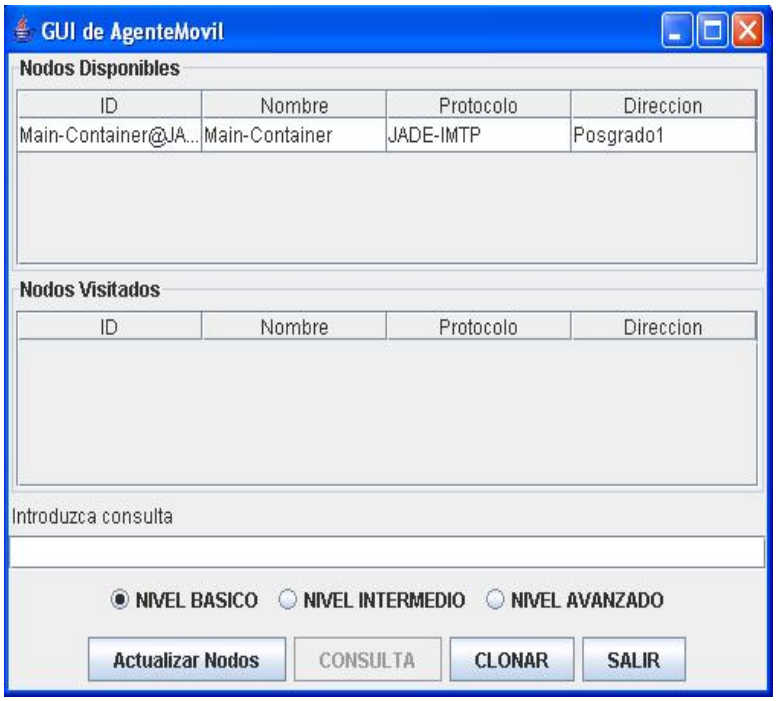

Figura 3.5 Interfaz del agente móvil. Se ilustra la tabla de nodos disponibles y nodos visitados. Una sección para introducir la consulta del usuario y botones para la selección de su nivel. Se cuenta con cuatro botones para visualizar los nodos actuales, un botón de consulta que se encuentra desactivado hasta que el agente regrese con sus datos. El botón clonar para lanzar al agente, sin embargo se muestra un mensaje de error si el usuario no ha introducido una consulta y seleccionado un nivel, por último el botón salir para abandonar la aplicación.

Para la implementación de la clase *MobileAgent*, ésta contiene dos comportamientos de vital importancia dentro de la ejecución, es decir, el contador de los agentes clonados y la lista de nodos visitados.

El método `setup()`crea la interfaz y agrega los comportamientos para: obtener la lista de nodos disponibles desde el AMS, recibir los mensajes entrantes e incrementar el contador de clones creados. Para llevar a cabo la comunicación, en el contenedor principal se registra del contenido del lenguaje SL0 y se registra el uso de la ontología de movilidad.

```
   getContentManager ().registerLanguage (new SLCodec (),
FIPANames.ContentLanguage.FIPA_SL0);
   getContentManager().registerOntology(MobilityOntology.getInstance(
));
```

Para la implementación de la clase o comportamiento *GetAvailableLocationsBehaviour,* esta hereda de la clase SimpleAchieveREInitiator para solicitar al AMS la lista de los nodos disponible hacia los que el agente puede migrar.

La clase o comportamiento *ServeIncomingMessagesBehaviour*, como su nombre lo indica, sirve para recibir todos los mensajes, en particular las siguientes expresiones son aceptadas como contenido de mensajes "request":

- (move <destino>): Mover al agente a otro contenedor. O bien
  (move (:ubicación (:nombre Container-1) (:transport-protocol JADE-IPMT) (:transport-address IOR:0000...)))
- (exit) : solicitar al agente salir.
- (stop): detener el contador de clones.
- (continue): continuar con el conteo de clones.

El método action() de la clase, obtiene un mensaje de la cola o bien, espera por un mensaje nuevo si es que la cola está vacía. En caso de que exista algún mensaje se identifica la acción a ejecutar, es decir, si se solicito que el agente se moviera, saliera, detuviera o continuara el conteo de clones.

3.4 Implementación de la Ontología

La primera tarea a realizar para el desarrollo de la ontología era introducir la jerarquía o taxonomía de clases según el diagrama 2.9 de la sección anterior. Las clases padre están a la izquierda de la taxonomía mientras que las subclases están cada vez más a la derecha. Es importante mencionar que Protégé agrega por default la clase owl:Thing, la cual representa el conjunto que contiene a *todos* los individuos, por que todas las clases que se añadan son subclases de owl:Thing. Una tarea recomendada para desarrollar ontologías en esta herramienta es escribir la primera letra del nombre de una clase con mayúsculas, no incluir caracteres especiales ni dejar espacios. En la figura 3.6 se ilustra la taxonomía de clases

desarrollada en Protégé. Las clases padre para este dominio de aplicación son: Autores, Nivel_agentes, Nivel_sociedades_de_Agentes, Niveles_Conocimiento, TecnologíaAgentes.

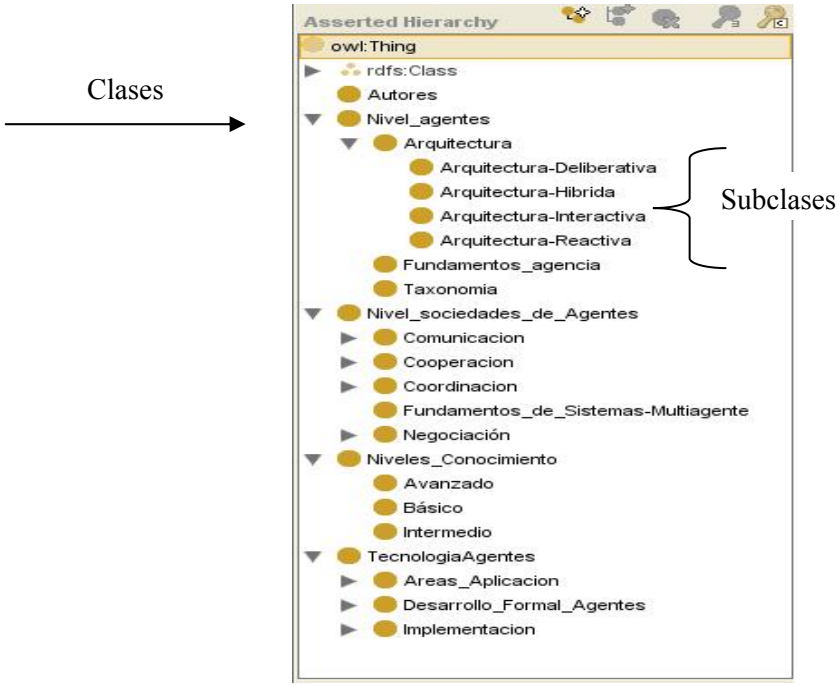

Figura 3.6 Jerarquía de clases del dominio de agentes dentro de la interfaz de desarrollo de Protégé. En la parte izquierda de la figura se aprecian las clases padre de la ontología, mientras que la parte derecha de la figura muestra las subclases o clases hijo.

La siguiente tarea a realizar es agregar las relaciones que una clase tenga con otras, también se puede agregar información adicional sobre esa clase mediante comentarios.

En la figura 3.7 se ilustra que la subclase *Arquitectura-Deliberativa* tiene un "comentario" que agrega más información sobre ella, también se identifican dos propiedades: *definida_por* y *tiene_Nivel*.

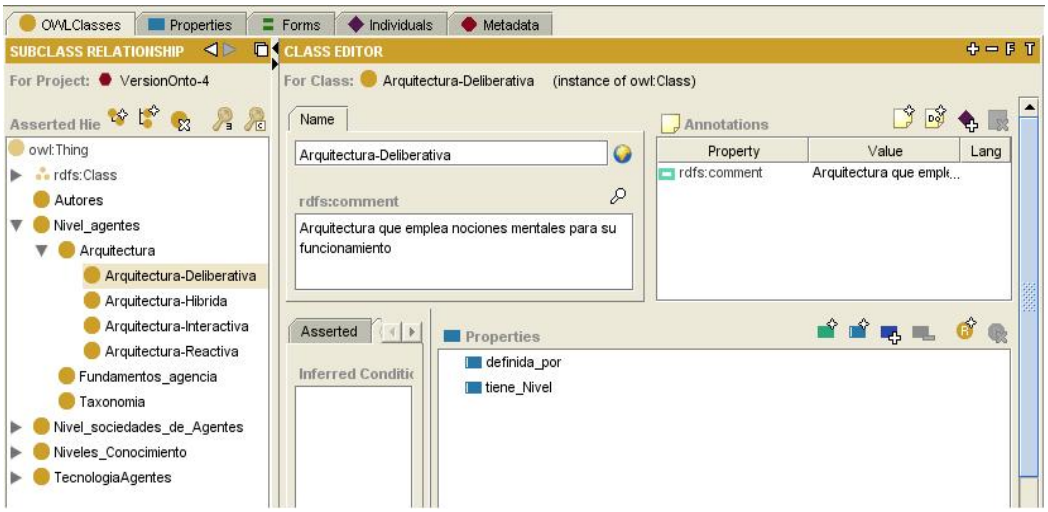

Figura 3.7 Propiedades de la clase Arquitectura-Deliberativa dentro de la interfaz de desarrollo de Protégé.

Para introducir los axiomas, Protégé proporciona restricciones mediante las cuales restringe las instancias que pertenecen a cierta clase. Las restricciones OWL incluyen los cuantificadores universal y existencial $\forall, \exists$ respectivamente. Además de operadores lógicos, en la figura 3.8 se ilustran.

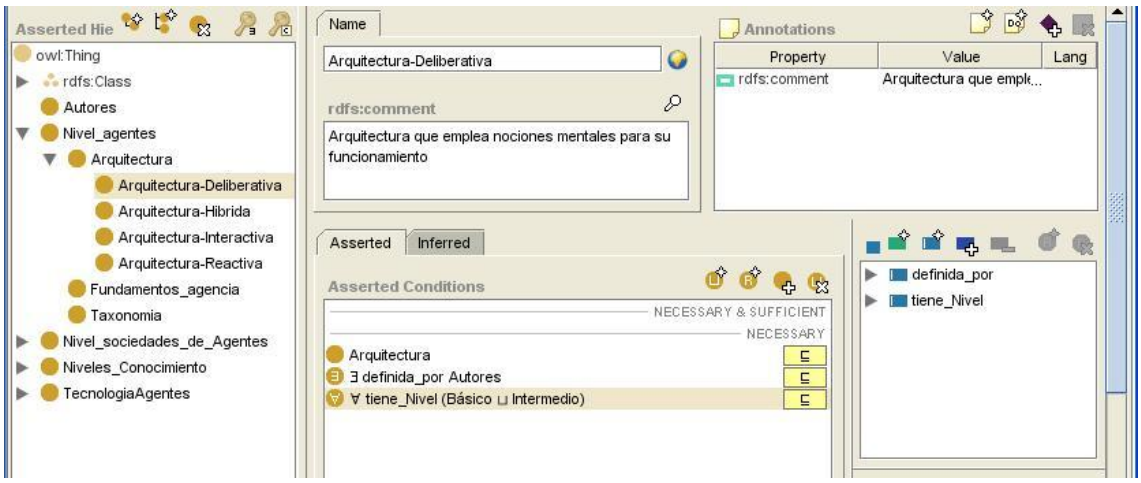

Figura 3.8 Axiomas de la clase Arquitectura-Deliberativa dentro de la interfaz de desarrollo de Protégé.

Para establecer axiomas se recurre a las propiedades, se ejemplifica para la clase *Arquitectura-Deliberativa* que hereda de la clase *Arquitectura*, hace referencia también a la clase *Autores*, con ello se quiere representar que determinada instancia de la clase *Arquitectura-Deliberativa* es creada por lo menos por un objeto(s) de la clase *Autor*. La tercera restricción utiliza la propiedad *tiene_Nivel*, como se mencionó anteriormente toda clase está

ligada a un nivel de conocimiento, para este caso en particular se define por la unión de las clases Básico e Intermedio.

Para introducir las instancias de una clase, Protégé permite realizarlo a través de su Instance Browser, recuerde que las instancias son objetos específicos de una clase. Para su ilustración se presenta la clase Autores, cuenta con 19 instancias que se ejemplifican en la parte derecha de la figura 3.9.

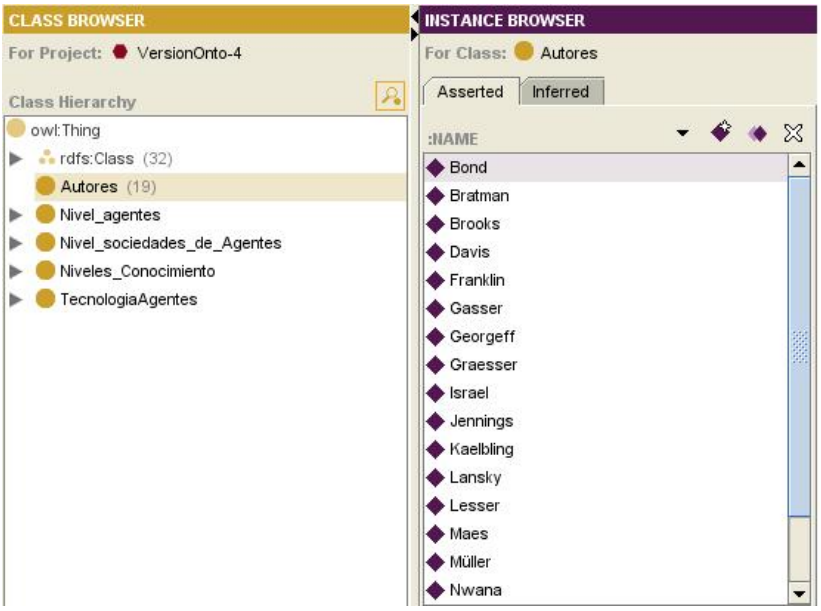

Figura 3.9 Instancias de la clase Autores dentro de la interfaz de desarrollo de Protégé.

Una vez generada la ontología en Protégé, se obtiene un documento OWL, en él quedan definidas las clases, sus relaciones, instancias y axiomas que fueron identificadas en el proceso de construcción de la ontología, además de alguna otra inferencia que realice automáticamente Protégé. El documento que se genera es un conjunto de *"metadatos"* desarrollados por Protégé, los cuales son empleados para describir la jerarquía de clases, es decir, localiza clases y subclases; identifica las características que le fueron especificadas a cada una de ellas, además de su relación con otras; también incluye información adicional que permite describir con mayor acercamiento el dominio de aplicación.

Para manipular el fichero XML que se generó se empleo un parser realizado en java. SAX (Simple API para XML) que a diferencia de otros parsers es fácil de utilizar y es ideal para manipular archivos de gran tamaño. La ventaja que ofrece SAX es que procesa la información por eventos, es decir, procesa la información en XML conforme esta sea presentada, trata cada

elemento a un determinado tiempo, sin incurrir en uso excesivo de memoria. El mecanismo mediante el cual se extraen los términos relevantes de la ontología se expuso en la sección 2.2.2 con la ayuda del parser SAX.

3.5 Implementación de Extractos mediante el Punto de Transición y el Texto Enriquecido de Búsqueda

A continuación se presenta el extracto obtenido para una página html determinada, aplicando el algoritmo especificado en el apartado 2.2.3.

Tabla 3.1 Documento sin procesar

```
<META content="Mozilla/4.03 [en] (Win95; I) [Netscape]"
name=GENERATOR>
      <CENTER><B><FONT color=#000099><FONT size=+3>Hechos, Ficción y
      Pronóstico</FONT></FONT></B><BR><BR></CENTER>
      <CENTER><FONT size=-1>Topicos Avanzados de Informatica
II</FONT></CENTER>
      <CENTER><FONT size=-1>Universidad Blas Pascal</FONT></CENTER>
      <CENTER><FONT size=-1>Resumen y  REC ute n : <I>Hugo Andres
      Pelaez</I></FONT></CENTER>
      <P>
      <HR width="100%">
      <BR><B><I><FONT  REC=Arial><FONT size=-1>En el futuro, los
agentes de
      software inteligentes nos ayudarán a navegar en la
supercarretera de la
      información sirviendo como guías de viajes o conductores de
taxi. Mejor
      aun, tienen el potencial de actuar como conserjes sofisticados,
que hacen
      innecesario para nosotros enfocar la autopista en
       REC ute.</FONT></FONT></I></B>
      <P>Lejos de "desaparecer", la supercarretera de información
parece ser
      mayor y más prominente cada día. Los agentes de software
inteligentes
      pueden revertir esta tendencia de dos modos:
      <DIR><IMG height=13
      src="Agentes Inteligentes en
InternetSinrevisar_archivos/Bullet5.gif"
      width=13><B> Abstracción: </B>El agente oculta los detalles de
la
      tecnología subyacente. La persona plantea qué información
requiere y el
      agente determina donde buscar la información y como recuperarla.
```

Continúa…

Tabla 3.2 Documento preprocesado.

1. Se eliminan tags 2. Se eliminan stopwords y signos de puntuación. Se analizan las abreviaturas 3. Se segmenta en oraciones.
1 hechos ficción y pronóstico topicos avanzados de informatica ii universidad blas pascal REC andres pelaez futuro, agentes software inteligentes ayudarán navegar supercarretera información sirviendo guías viajes conductores taxi. **2** mejor potencial actuar conserjes sofisticados innecesario para nosotros enfocar autopista absoluto. **3** lejos desaparecer supercarretera información parece mayor prominente día. **4** agentes software inteligentes revertir tendencia modos abstracción agente oculta detalles tecnología subyacente. **5** persona plantea información requiere agente determina buscar información recuperarla. **6** distracción carácter personalidad agente ayuda distraer persona tarea tediosa compleja. **7** agentes software inteligentes abstracción ganará distracción. **Continúa…**

La siguiente tabla muestra el vocabulario obtenido del documento anterior, muestra la palabra y su frecuencia de ocurrencia.

Tabla 3.3 Vocabulario obtenido de documento.

Palabra	**FREC.**	**Palabra**	**FREC.**	**Palabra**	**FREC.**
Softbot	38	modelos	6	arquitectura	2
información	34	preguntas	6	actividades	2
agentes	25	comandos	5	acción	1
internet	18	interfaz	5	actúan	1
agente	17	servicios	5	adaptarse	1
objetivos	12	administrador	5	agencia	1
usuario	12	acciones	4	autónomo	1
software	10	búsqueda	4	buscadores	1
indexados	7	entorno	4	ciencias	1
web	7	adaptabilidad	3	componente	1
universidad	7	algoritmos	3	dueño	1
				Frecuencia 1	615

Los datos necesarios para obtener el PV se detallan en la tabla 3.4.

Tabla 3.4 El Párrafo Virtual.

Total de términos en el documento con frecuencia 1	626
Consulta	"Agentes de Internet"
PT	35
Límite superior	44
Límite inferior	27
Términos PV	{información, softbot}
TEB	{Internet, web, redes, aplicación, recuperación}

Las oraciones obtenidas con las que se conforma el resumen aplicando la función de Jaccard.

1. *Es relativamente directo representar la semántica de esa información al Softbot.*
2. *Una razón por la que el Internet Softbot tiene éxito es su enfoque en la información estructurada de los servicios de bases de datos e información.*
3. *Lejos de "desaparecer", la supercarretera de información parece ser mayor y más prominente cada día.*
4. *En lugar de eso, el Softbot confía en un modelo del servicio para entender la semántica precisa asociada con la información suministrada por el servicio.*
5. *Modelos del dominio de Internet: Los modelos de dominio de Internet proveen al Softbot con información secundaria acerca de Internet.*
6. *¿qué es un agente? un agente de software es un programa de computadora que se comporta en una manera análoga a un agente humano.*
7. *propiedades del agente el softbot es altamente autónomo, orientado a objetivo, flexible, y auto iniciable.*
8. *si pudiera aprender reglas de control de búsqueda efectivas automáticamente, el softbot lograría muchos más objetivos.*
9. *los investigadores han propuesto las siguientes características como cualidades deseables en los agentes de software: autonomía: toma la iniciativa y ejerce el control sobre sus propias acciones en la siguiente forma: orientado a objetivo: acepta requerimientos de alto nivel y decide cómo y dónde satisfacerlos.*
10. *el internet softbot es un excelente ejemplo de un agente orientado a objetivos.*

Capítulo 4
Pruebas

4.1 Descripción de los Datos

Un corpus es una colección de documentos y de consultas supervisadas, cuyo objetivo es apoyar la investigación en el área del procesamiento del lenguaje natural, proporciona la infraestructura necesaria para la evaluación de metodologías de la RI (Bueno, 2005). Existen diversos tipos de corpus, los más comunes son aquellos que se conforman por una recopilación de noticias de diversos periódicos.

Sin embargo para efectos de este trabajo el tipo de corpus mencionado anteriormente era difícil para realizar las pruebas, pues se requería de un conjunto de documentos referentes a la teoría de agentes.

Por ello se realizó la confección de un pequeño corpus en la temática de la teoría de agentes. Los documentos seleccionados fueron exclusivamente páginas web, sin importar el tamaño de las mismas. El número inicial de documentos que se recolectaron para conformar el corpus fueron 145, sin embargo debido al tipo de formatos que se utilizan para representar información a la web (Windows 1252, ISO 8859, HTML, entre otros) se seleccionaron sólo aquellos documentos con el formato UTF-8, ya que el preprocesamiento de las páginas web que se realiza en este trabajo está bajo dicho estándar. Con esta segunda selección de documentos, el corpus quedó conformado por 70 páginas html. En el corpus se incluye también un resumen de cada página web, el cual fue realizado por un resumidor base, para este trabajo se empleó la herramienta de distribución libre Copernic Summarizer [22]. Con el objetivo de comparar la calidad de los resúmenes generados por el agente.

4.2 Evaluación de resultados sin utilizar la Ontología

Es realmente difícil realizar una evaluación sobre la calidad de los resúmenes obtenidos. En diversos trabajos se ha empleado la ayuda de jueces humanos quienes elaboran su propio resumen (Salazar, 2004). La complejidad se vuelve aun mayor cuando los jueces no extraen frases del documento original; sino que elaboran resúmenes con sus propias ideas, uniendo 2 o 3 frases relevantes de un documento para conformar una oración del resumen.

El primer experimento consiste en evaluar las palabras claves proporcionadas por el método del Párrafo Virtual contra las palabras claves de la herramienta Copernic Summarizer. Con el objetivo de analizar la similitud entre las palabras clave ofrecidas por el agente y por la herramienta Copernic se propone utilizar las métricas de precisión y recuerdo. La precisión mide que tan cerca estuvo el sistema de producir una salida igual a la que produce la herramienta. El recuerdo refleja lo que el sistema olvidó (Baeza, et al, 1999).

Para ello se define como *correctas* al número de palabras clave extraídas por el agente y por la herramienta base. *Incorrectas* es el número de palabras claves extraídas por el agente pero no por el humano; y por último las *olvidadas*, son el número de palabras clave extraídas por la herramienta pero no por el sistema. Así se tiene lo siguiente:

$$P = \frac{correctas}{(correctas + incorrectas)} \text{ (Precisión)}$$

$$R = \frac{correctas}{(correctas + olvidadas)} \text{ (Recuerdo)}$$

Los resultados promedio obtenidos fueron los siguientes:

Precisión	**Recuerdo**
0.389619929	0.35865628

El valor de precisión indica que aproximadamente el 39% de las palabras clave que recuperó el agente fueron buenas, el valor de recuerdo refleja que aproximadamente el 36% de palabras clave fueron olvidadas por el agente. Para evaluar un sistema de recuperación de información y decir que éste es bueno el valor de precisión debe ser mayor que el recuerdo.

Además cabe resaltar que por el momento no se han reportado sistemas de recuperación de información que tengan una precisión de por lo menos el 50%, por lo que consideramos como alentadores estos resultados.

El segundo experimento utiliza el índice de utilidad propuesto en (Otterbacher, et al, 2002). Dicha métrica se basa en la hipótesis que no todas las frases que se seleccionan para conformar un resumen tienen la misma importancia. Por ello cada juez asigna una ponderación que va de 1 a 10.

El índice de utilidad se calcula dividiendo la suma de las ponderaciones de las oraciones seleccionadas por el sistema entre la suma de ponderaciones de las oraciones que conforman el resumen de referencia.

Para dicho experimento, cada página web fue analizada por un juez humano experto en el área de agentes, a cada oración le asignó una ponderación que va desde 0 hasta 10. Se incluyó el 0 por si las oraciones seleccionadas no aportaban información relevante al juez. Las pruebas de este experimento fueron realizadas con una muestra aleatoria de 15 páginas web y los resultados se muestran a continuación. La muestra pareciera ser muy pequeña sin embargo el esfuerzo humano que implica es muy grande.

Como puede apreciarse en la gráfica, Figura 4.1, en 6 de los documentos Copernic recuperó oraciones más importantes para el experto, contra que en 9 de los casos el agente obtuvo mayor número de oraciones dentro de las relevantes para el juez humano 1.

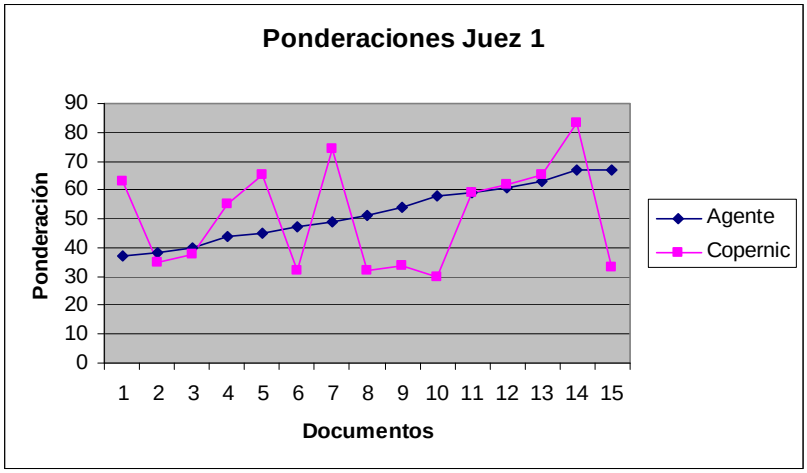

Figura 4.1 Gráfica comparativa de las ponderaciones del Agente y Copernic con respecto al Juez 1

En aras de demostrar que el agente siempre recupera un mayor número de oraciones en las que el experto ponderó con mayor valor, se realizó la prueba de proporciones. Estableciéndose como hipótesis de nulidad $H_0 : p_1 - p_2 = p_0$, que no existe diferencia significativa entre las oraciones con mayor ponderación brindadas por el experto y por el sistema, contra la hipótesis alternativa de que la proporción obtenida es diferente $H_1 : p_1 - p_2 \neq p_0$.

Los resultados son los siguientes:

```
Sample  X   N  Sample p
1       6  15  0.400000
2       9  15  0.600000

Difference = p (1) - p (2)
Estimate for difference:  -0.2
95% CI for difference:  (-0.550609, 0.150609)
Test for difference = 0 (vs not = 0):  Z = -1.12  P-Value = 0.264
```

Analizando los resultados obtenidos para un nivel de significación de 0.05, puede verse que no se puede rechazar la hipótesis de nulidad, por lo que no se puede afirmar de manera categórica que el sistema siempre va a recuperar un mayor número de oraciones con mayor ponderación.

Considerando los resultados obtenidos por el juez 1, se decidió considerar la valoración dada en términos de ponderación ofrecida por un segundo juez humano. Las ponderaciones obtenidas por el sistema y por Copernic para el juez 2 se muestran en la siguiente gráfica, figura 4.2.

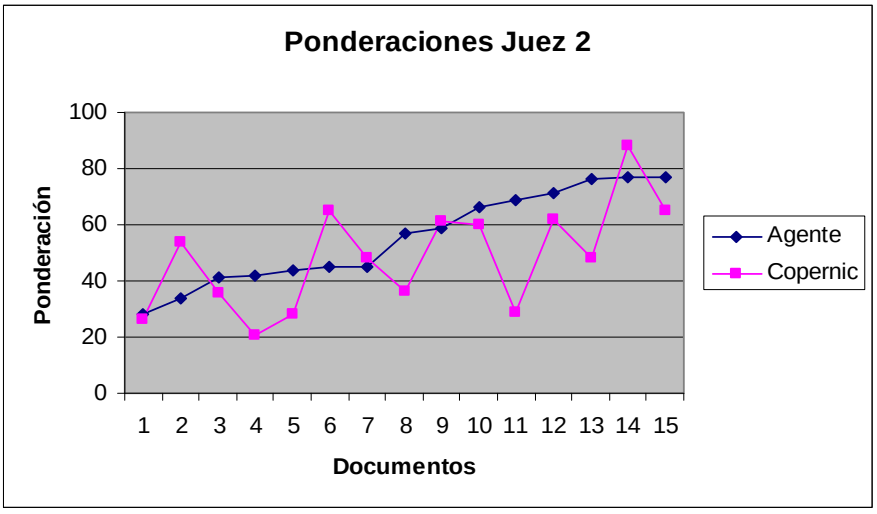

Figura 4.2 Gráfica comparativa de las ponderaciones del Agente y Copernic con respecto al Juez 2

Para 5 páginas html, de las 15 analizadas, la herramienta Copernic ofreció mejores resúmenes compuesto por oraciones que el experto dio mayor relevancia. De manera similar en 10 páginas el agente superó a la herramienta.

Se realizó el mismo experimento obteniéndose los siguientes resultados:

```
Sample   X   N  Sample p
1        5  15  0.333333
2       10  15  0.666667

Difference = p (1) - p (2)
Estimate for difference:  -0.333333
95% CI for difference:  (-0.670707, 0.00404035)
Test for difference = 0 (vs not = 0):  Z = -1.94  P-Value = 0.053
```

Con los resultados obtenidos no se puede rechazar la hipótesis de nulidad por lo que nuevamente no puede afirmarse que el agente siempre obtenga las oraciones de mayor relevancia para el segundo juez.

4.3 Evaluación de resultados utilizando la Ontología

Con el objetivo de mostrar que al enriquecer los términos de la consulta se puedan recuperar mayor cantidad de oraciones que contengan términos significativos se realizó la prueba del índice de utilidad.

Las páginas html que conformaron la muestra aleatoria tratan sobre diferentes temas de la teoría de agentes, cuando se realiza una consulta al agente, los resúmenes que presentaron mayores ponderaciones fueron aquellos que se relacionaban directamente con la temática de la consulta, esto es totalmente lógico pues el Texto Enriquecido de Búsqueda TEB se confecciona adicionando a la consulta los términos recuperados por medio de la ontología. Los resultados obtenidos se muestran a continuación, en las figuras 4.3 y 4.4.

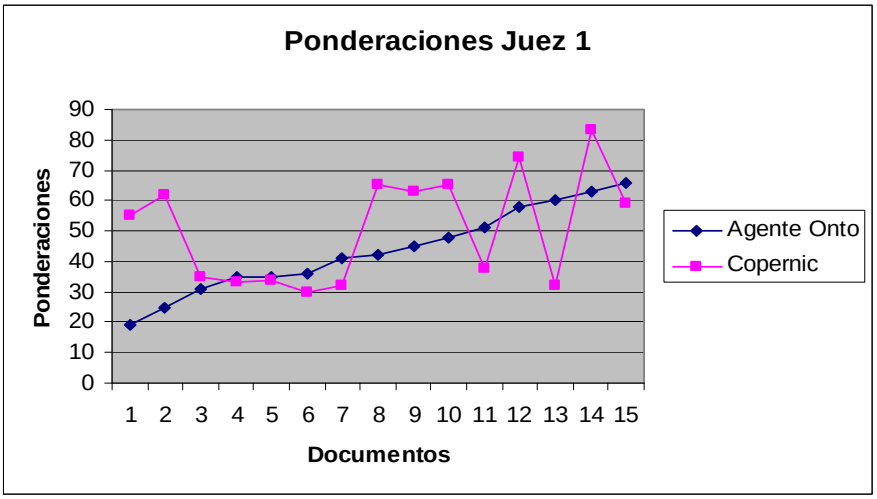

Figura 4.3 Gráfica comparativa de las ponderaciones del Agente más Ontología y Copernic con respecto al Juez 1

De manera similar se obtuvieron los siguientes resultados para el Juez 2.

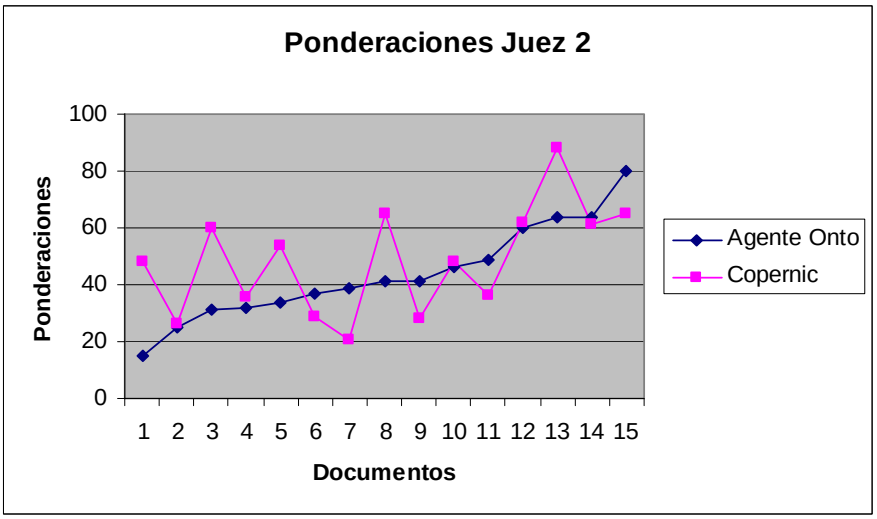

Figura 4.4 Gráfica comparativa de las ponderaciones del Agente más Ontología y Copernic con respecto al Juez 2

Estos resultados son totalmente lógicos por la forma en que se construye el resumen. Para medir la importancia de cada oración dentro del documento se aplica la función de similitud de Jaccard, que en términos generales brinda una medida de importancia de la misma al considerar en el numerador la intersección de los términos y en el denominador la unión. Si el documento está muy alejado de la consulta, los términos de la intersección no crecen, contrario a la unión que se incrementa sustancialmente por la incorporación de nuevos términos a partir

de la ontología. Esto provoca que la medida que se obtenga sea más pequeña y distorsiona con esto la relevancia de cada oración.

Es por ello que las páginas que no tienen ninguna relación con la consulta, no se logra recuperar las oraciones que están marcadas por los expertos como más relevantes.

Por otra parte, en ambos tipos de pruebas se enfrentó al problema de que las páginas web están asociadas a diversos formatos para presentar información, además las páginas en español muestran mayor complejidad para tratarlas por la existencia de acentos y tildes. Otro detalle fue que algunas páginas seleccionadas no cuidan los errores ortográficos, pues omiten los acentos o signos de puntuación, haciendo que el proceso de recuperación se complique dado que este trabajo no propone la realización de análisis sintáctico sobre el documento. Con lo cual se confirma la necesidad de estandarizar la información contenida en la Web.

Conclusiones

Actualmente la importancia y la necesidad de implantar herramientas en sistemas distribuidos ha crecido considerablemente. Esto se ve reflejado en las diversas tecnologías desarrolladas para mantener comunicados equipos remotos que ejecuten tareas y tengan acceso a información. Los agentes móviles son la consecuencia de la evolución en las tecnologías de código móvil debido a la capacidad de migración que ofrecen.

La generación automática de resúmenes es una línea de investigación actual en Computación y el hecho de ligar a los agentes móviles con el área de recuperación de información enriquece el trabajo sobremanera.

Este trabajo de tesis propone el desarrollo de un sistema de software que utiliza agentes móviles que recuperan información, específicamente mediante la elaboración de resúmenes de páginas web. Para la elaboración del resumen se emplea la técnica del Párrafo Virtual; conjunto de términos caracterizados como importantes dentro de un documento. Sin embargo la necesidad de los usuarios que utilizan la Web requiere que los mecanismos de búsqueda ofrezcan resultados apegados a sus intereses. Por ello se elaboró una Ontología de la teoría de agentes. Dicha ontología consideraba la consulta y nivel de conocimiento del usuario para devolver términos relevantes de acuerdo a los requerimientos señalados por un usuario. Además se proporciona una interfaz para que el usuario realice las consultas deseadas.

Los experimentos realizados sobre la plataforma JADE permiten concluir que el agente puede moverse de un nodo a otro sin dificultad alguna dentro de una intranet. Esta plataforma permite la movilidad de los agentes de manera transparente al usuario y garantiza que pueda visitar un número determinado de nodos y regresar a su máquina servidor con la información recuperada, a diferencia de otras plataformas como ProActive, que se utilizó en trabajos anteriores (Leal, et al, 2005). Se diseñó e implementó la estrategia de lanzamiento del agente aprovechando las bondades de JADE, obteniéndose un buen desempeño con respecto al sistema desarrollado en ProActive.

En lo que respecta a la obtención de los extractos se ha presentado un algoritmo para obtener las oraciones representativas de un texto, las cuales constituyen lo que es llamado un extracto de texto. Muchas de las técnicas empleadas en el área de RI son probadas en corpus especializados, sin embargo lo que realmente se necesita es utilizar dichas técnicas en ambientes heterogéneos como Internet.

Los problemas identificados en este trabajo para realizar resúmenes de páginas web, son los diversos estándares de codificación utilizados para presentar información: las páginas en español muestran mayor complejidad para ser tratadas debido a la existencia de acentos y tildes; es por ello que se decidió realizar las pruebas sobre páginas con formato UTF-8 únicamente. Además algunas páginas seleccionadas no cuidan los errores ortográficos, pues omiten los acentos o signos de puntuación, haciendo que el proceso de recuperación se complique dado que este trabajo no propone la realización de análisis sintáctico sobre el documento. Sin embargo esta problemática refleja la necesidad de estandarizar la información contenida en la Web.

Se emplea una versión demo de la herramienta Copernic Summarizer para verificar la calidad de los resúmenes.

Como mejoras futuras a este trabajo se mencionan: la realización de una lematización de los términos de un documento, con ello se asegura considerar la raíz de una palabra, analizando posteriormente con esto si se mejoran los resultados.

Es importante desatacar que la ventaja de expandir los términos con la ontología sólo se aprecia si el documento trata directamente con la palabra de consulta y en el que los términos de la ontología brinden mayor información. Otro trabajo futuro puede consistir en desarrollar un Tesauro parecido a WordNET [23], pero en el idioma español, que es tan rico en sinónimos, homónimos y anáforas.

Referencias

1. Alani H., Kim S., Millard D., Weal D., Hall W., Lewis P., Shadbolt N., "Web based Knowledge Extration and Consolidation for Automatic Ontology Instantiation", Proceedings of Knowledge Capture (K-Cap'03), Workshop on Knowledge Markup and Semantic Annotation, 2003, pp. 1-7.

2. Baek J., Kim G., Yeom H., "Timed Mobile Agent Planning for Distributed Information Retrieval", Proceedings of the Fifth International Conference on Autonomous Agents, ACM Press, 2001, pp. 120-121.

3. Baeza-Yates. R., Ribeiro. B., "Modern Information Retrieval", Addison Wesley, 1999, pp. 144-148, 163-186.

4. Bautista A., López F., "Ontologías y Herramientas para su diseño", La computación en Puebla en el siglo XXI, BUAP, 2005, pp. 231-235.

5. Bueno C., Tesis de Maestría: "Métodos para la Generación de Extractos mediante el uso del Párrafo Virtual", BUAP Facultad de Ciencias de la Computación, 2005, pp.8-13.

6. Busetta P., Ramamohanarao K., "An architecture for Mobile BDI Agents", Proceedings of the 1998 ACM Symposium on Applied Computing, ACM Press, 1998, pp. 445-452.

7. Castells P., Macías A., "Un sistema de representación dinámica en entornos web para representaciones personalizadas del conocimiento", Revista Iberoamericana de Inteligencia Artificial, No. 20, 2001, pp. 34-52.

8. Grossman D., Frieder O., "Information Retrieval Algorithms and Heuristics", Springer, Segunda Edición, 2004, pp. 1-8.

9. Horvat D., Cvetković D., Milutinović V., Koćović P., Kovaćevic V., "Mobile Agents and Java Mobile Agents Toolkits", Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000, pp. 1-10.

10. Jacobson I., Booch G., Rumbaugh J., "El Proceso Unificado de Desarrollo de Software", Addison Wesley, 2000, pp. 105-123, 125-133, 255-262.

11. Kotz D., Gray R., "Mobile Agents and the Future of the Internet", Department of Computer Science / Thayer School of Engineering Dartmouth College, In ACM Operating Systems Review 33(3), 1999, pp. 7-13.

REFERENCIAS

12. Leal L., Vilariño D., López F., “Agentes Móviles para Extractos de Documentos”, 6to. Congreso Estudiantil de Computación, Research on Computing Science, 2005, pp. 63-74.

13. Lozano, “Ontologías en la Web Semántica”, I Jornada de Ingeniería Web’ 01, 2001, pp. 1-4.

14. M. Wooldridge, “An Introduction to Multiagent Systems”, John Wiley & Sons, 2002, pp. 1-18.

15. Otterbacher J., Winkel A., Radev D., “The Michigan Single and Multi-document Summarizer for DUC 2002”, pp. 1-4.

16. Salazar H., Tesis de Maestría: “Obtención de Extractos de Textos con base en un Corpus”, BUAP, Facultad de Ciencias de la Computación, 2004, pp. 1-27.

17. Téllez A., “Extracción de Información con Algoritmos de Clasificación”, INAOEP, 2005, pp. 26-27.

18. Unschold M., King M., “Towards a Methodolgoy for Building Ontologies”, Workshop on Basic Ontological Issues in Knowledge Sharing” ,IJCAI 95, 1995, pp. 4-14.

19. Weiss G., “Multiagent Systems A Modern Approach to Distributed Artifical Intelligence”, The MIT Press, 2000, pp. 1-70.

20. http://jade.cselt.it/

21. http://protege.stanford.edu/

22. http://www.copernic.com/

23. http://wordnet.princeton.edu/

Apéndice A
Instalación de JADE

En esta sección se describirán los pasos que hay que seguir para instalar la plataforma *JADE* de forma adecuada en sistema operativo Windows y Linux. El único requerimiento software es poseer una versión de Java *JDK 1.2* o superior.

Windows

Una vez que se han descargado los archivos `jadeBin.zip` y `jadeExamples.zip, éstos deben` ser descomprimidos.

A continuación se cambia la variable `CLASSPATH` del sistema para incluir los ficheros *.jar* contenidos en el subdirectorio *lib* y el directorio actual.

```
set CLASSPATH=%CLASSPATH%;.;c:\jade\lib\jade.jar;c:\jade\lib\jadeTools.jar;c
:\jade\lib\Base64.jar; c:\jade\lib\iiop.jar
```

Linux

Se deben modificar las variables de entorno, en el archivo /etc/enviroment (o donde corresponda según la distribución). En el ejemplo, la ruta de instalación de jade es /home/jade.

```
JAVA_HOME=/usr/bin/java/
JADE_HOME=/home/jade/
CLASSPATH=$CLASSPATH:/home/jade/lib/jade.jar:
CLASSPATH=$CLASSPATH:/home/jade/lib/http.jar:
CLASSPATH=$CLASSPATH:/home/jade/lib/iiop.jar:
CLASSPATH=$CLASSPATH:/home/jade/lib/Base64.jar:
CLASSPATH=$CLASSPATH:/home/jade/lib/jadeTools.jar:
export JAVA_HOME JADE_HOME CLASSPATH
```

O de una manera más simplificada:

```
export JAVA_HOME=/usr/bin/java/
export JADE_HOME=/home/jade/
export
CLASSPATH=$CLASSPATH:/home/jade/lib/jade.jar:/home/jade/lib/http.jar:\/home/
jade/lib/iiop.jar:/home/jade/lib/Base64.jar:/home/jade/lib/jadeTools.jar:\
```

Para arrancar la RMA y probar así que su instalación es correcta podemos ejecutar el siguiente comando:

```
java jade.Boot –gui
```

Si se realiza correctamente, debe aparecer una ventana de consola (agente RMA) a partir de la cual se pueden crear agentes y enviar mensajes entre ellos.

Dentro de la librería *jade/src/examples* se encuentran una serie de ejemplos escritos en Java, que son agentes que también se pueden ser ejecutados desde la línea de comandos. Para ejecutar algunos de estos ejemplos se debe situar en el directorio *jade/src* y hacer referencia a los agentes con el nombre completo del paquete. Todos los ejemplos están situados en subpaquetes del paquete '*examples*', por ejemplo la clase *ComplexBehaviourAgent* en el subdirectorio '*behaviours*' del directorio '*examples*' debe ser arrancada desde el directorio *jade/src* con el comando:

```
java jade.Boot -platform a1:examples.behaviours.ComplexBehaviourAgent
```