UNIVERSIDAD POPULAR AUTÓNOMA
DEL ESTADO DE PUEBLA

Departamento de Maestría en Tecnologías de
Información y Análisis de Decisiones

"Desarrollo de agentes móviles para generar extractos de documentos usando el párrafo virtual"

TESIS PROFESIONAL

QUE PARA OBTENER EL GRADO DE
MAESTRO EN TECNOLOGIAS DE INFORMACION Y ANALISIS DE DECISIONES

PRESENTA:
JOSE ABRAHAM CRUZ VERGARA

ASESOR:
DRA. DARNES VILARIÑO AYALA

Puebla, Pue. Agosto 2007

Tabla de contenido

Indice de figuras y tablas

Introducción

Internet se ha convertido sin duda alguna en la mayor fuente de información disponible. Sin embargo la información se expande por la Web de forma caótica en incontrolada, originando serias limitaciones para su manejo, organización y recuperación; se sabe que hay un nuevo servidor cada dos horas.

- En el año 1995 había 5 millones de documentos.
- En el año 1998 había 320 millones de documentos.
- En el año 2000 había más de 1.000 millones de documentos [1].

Los índices se vuelven obsoletos muy rápidamente, lo cual hace necesario disponer de mejores medios para descubrir recursos y extraer conocimiento, el 99% de la información no interesa al 99% de las personas. A pesar del manejo de grandes cantidades de documentos la cobertura de la web es limitada, la mayoría de datos útiles se encuentran en bases de datos y las búsquedas se realizan orientadas a palabras clave.

Los procedimientos para proporcionar información y accederla son relativamente fáciles, desde el punto de vista de usuario (proveedor/cliente), pero no lo es tanto desde el punto de vista tecnológico de los recursos de la red, debido a que el sistema de gestión de la información no se puede controlar. Si todos los usuarios de la Web fuesen humanos esta situación no representaría mayor problema. Sin embargo, en la actualidad no todos los usuarios son humanos, pues típicamente encontraremos robots de búsqueda Web, que son programas de computadora que recorren la estructura hipertexto de la Web en forma recurrente, recuperando todos los documentos que coinciden con un criterio específico.

Los robots software [2] (algunas veces llamados Agentes) juegan un rol muy importante en el ambiente Web, y una de sus tareas primordiales es recuperar información. Una gran cantidad de información útil se almacena en servidores Web alrededor de todo el mundo, pero no puede ser accesada a menos que el usuario especifique una URL (Universal Resource Locator). Para obtener la URL, se necesita de un servicio de búsqueda, y la tecnología de robots de software se usa para construir los índices de búsqueda automáticamente, esta tecnología también se usa para administrar máquinas servidor y clientes, analizar estadísticas, mantener la consistencia entre documentos hipertexto y sus mirrors, etc.

La terminología usada para designar a los robots en la Web da la impresión de que estos en realidad viajan a través de toda la red, lo cual es una imprecisión pues todos estos programas son estáticos. Casi todos los robots de búsqueda convencionales se conectan a los servidores Web usando el protocolo HTTP, solicitan un documento HTML, lo reciben, lo analizan, crean índices del documento, extraen las URL's de otros sitios para luego accederlos.

La tecnología de robots software móviles se convierte entonces en una tecnología clave en el ambiente Web, dado que permite tener más control sobre los recursos de la red, como es el ancho de banda. Si se usaran robots de búsqueda móviles, los robots migrarían a sitios Web, obteniendo y analizando los documentos HTML en forma local, y extrayendo nuevas URL's. Luego el robot móvil debería regresar a su punto de partida o migrar a otro sitio Web para obtener más información.

La teoría de agentes –y por extensión la relativa a los agentes móviles– establece una serie de elementos que pretenden dar un paso más allá en el tratamiento informático distribuido, añadiendo características como la localización o la situación, y permitiendo la interacción dinámica de componentes autónomos y heterogéneos.

Los agentes móviles añaden una singularidad especial al concepto de agentes: la posibilidad de trasladarse de una máquina a otra. Esta característica ofrece ciertas ventajas respecto al tratamiento de la información en modo cliente/servidor, sobre todo en la computación a través de Internet. El auge mundial de "la red de redes" ha permitido el rápido desarrollo de nuevas técnicas inteligentes para búsqueda, filtrado y gestión de datos.

El objetivo del presente trabajo es desarrollar un agente móvil que sea capaz de migrar a un determinado host, revisar todas las páginas HTML y realizar el correspondiente resumen de las mismas, aplicando técnicas de Recuperación de Información, en particular para la obtención de dicho resumen se aplica la técnica del párrafo virtual.

Para el desarrollo de la aplicación se seleccionó la plataforma JADE debido a sus características de movilidad, etc. Además se analiza la técnica de representación del documento, llamada punto de transición, y aplicando ésta, se procede a la generación del párrafo virtual, con éste y aplicando la similitud de Jaccard, se extraen las oraciones del documento más relevantes, conformando de esta manera el resumen de cada documento.

Organización del trabajo

La tesis se desarrolla de la siguiente manera: en el capítulo 1se hace una revisión del estado del arte de los sistemas de agentes y las plataformas.
En el capítulo 2 se describe la plataforma que se va a utilizar para modelar el agente, se presenta una revisión de las características y facilidades de la plataforma Jade, base de la propuesta que es utilizada para la programación de los agentes que van a servir para cumplir nuestro objetivo.
En el capítulo 3 se presentan los antecedentes de mayor importancia para la realización de este trabajo, se describe la base teórica para la obtención del párrafo virtual.
El capítulo 4 se muestra a través del lenguaje UML el diseño del agente.
El capítulo 5 describe la implementación del agente, así como la estrategia para el lanzamiento del mismo y los resultados de las pruebas realizadas.
Por último, en el capítulo 6 se muestran las conclusiones de este trabajo y las recomendaciones finales.

I. Aspectos Teóricos

A continuación se describirán los conceptos generales que se han usado para el desarrollo de esta tesis, como es el de agentes, agentes móviles, plataformas para el desarrollo de agentes, metodologías para el diseño de agentes, entre otras.

I.1 Agentes y Agentes móviles

Varios investigadores y grupos de investigación han definido el término de agente desde diferentes puntos de vista, esto ha ocasionado que en la actualidad existan diferentes definiciones de lo que es un agente.

Según Wooldridge (1996) [3] se distinguen dos usos generales del término agente:

Noción de agente débil: sistema de hardware o un sistema de cómputo basado en software que contiene las propiedades de autonomía, habilidad social, reactividad y proactividad.

Noción de agente fuerte: sistema computacional que, además de las propiedades mencionadas, es conceptualizado e implementado usando conceptos que son usualmente aplicados a humanos.

Entendiendo lo que es un agente, se pueden relatar las propiedades de los mismos de una manera más precisa [4].
Autonomía: los agentes proceden sin la intervención directa de las personas. Poseen la capacidad de razonamiento para generar cursos de acción.
Sociabilidad: los agentes interactúan con otros agentes mediante algún mecanismo de comunicación.
Reactividad: los agentes perciben su ambiente (mundo real, interfaz gráfica, conjunto de otros agentes, Internet o combinación de estos) y responden a los cambios de éste.
Proactividad: los agentes no sólo actúan en respuesta a su ambiente, sino que son capaces de tener comportamiento orientado a metas.

Además de las propiedades que Wooldridge menciona, en 1999 se le agregaron otras [5], que de manera general pueden resumirse en:

Cooperativo: en consecuencia a la sociabilidad, la cooperación entre agentes es una razón de ser para tener múltiples agentes para resolver problemas.
Aprendizaje: los agentes para ser inteligentes requieren la propiedad de poder aprender del ambiente que les rodea.
Veracidad: un agente no comunica deliberadamente información falsa.

Benevolencia: los agentes no tienen metas conflictivas, harán siempre lo que se les pida.
Racionalidad: un agente actuará para alcanzar sus metas en la medida que sus creencias, conocimientos y capacidad de razonamiento se lo permitan.
Movilidad: algunos agentes tienen la habilidad de viajar en una red de computadoras (en Internet).

Además los agentes pueden clasificarse desde diferentes puntos de vista que se mencionan a continuación:

En base a sus capacidades de resolver problemas [6] se clasifican en:
Agentes reactivos: reaccionan a cambios en su ambiente o a mensajes provenientes de otros agentes. No son capaces de razonar acerca de sus intenciones. Sus acciones se realizan como resultado de reglas que se establecen.
Agentes intencionales: son capaces de razonar acerca de sus intenciones y conocimientos, crear planes de acción y ejecutar dichos planes. Los agentes intencionales pueden ser considerados como sistemas de planeación.
Agentes sociales: poseen la capacidad de los agentes intencionales. Mantener los modelos de los otros agentes, razonar sobre el conocimiento incorporado a estos modelos. Toman decisiones y crean sus planes con respecto a los modelos de otros agentes.

Sobre la base de la autonomía, el aprendizaje y la cooperación [7], se clasifican en:
Agentes colaborativos: destacan su autonomía y cooperación (con otros agentes) para realizar su tarea, pueden aprender.
Agentes de interfaz: ponen énfasis en su autonomía y aprendizaje para realizar sus tareas. Los agentes de interfaz asisten y dan soporte al usuario para aprender el uso de una aplicación.

Otra clasificación de agentes propuesta en [8]:
Agentes móviles: Los agentes móviles son procesos de software que son capaces de transitar por una red, generalmente una WAN, interactuando con computadores alejados, reuniendo información para el usuario y volviendo a su origen cuando las tareas fijadas por el usuario se hayan completado. Las tareas que se pueden realizar son por ejemplo reservaciones de vuelos, manejo de una red de telecomunicaciones entre otras.
Los agentes móviles traen consigo grandes beneficios aunque no son funcionales, esto quiere decir que una tarea que realiza un agente móvil puede ser realizada por un agente colaborativo, la diferencia radica en que para movilizar el agente se requiere de un costo muy alto de recursos.

Algunas de las ventajas que se pueden conseguir al usar agentes móviles son:
• *Reducen el costo de comunicación*, por ejemplo cuando en una ubicación hay un gran volumen de información que necesita ser examinada y transmitida, esto ocuparía una gran cantidad de recursos en la red y consumiría mucho tiempo. En este caso el agente móvil puede establecer la información relevante al usuario y transmitir un resumen comprimido de esta información.

• *Facilitar la coordinación*, es más fácil coordinar un cierto número de requerimientos remotos e independientes al comparar sólo los resultados localmente.
• *Reduce los recursos locales*, los agentes móviles pueden elaborar sus tareas en computadoras diferentes de la máquina local, de tal manera que no consuman recursos de procesamiento, memoria y almacenamiento en estos.
• *Computación asíncrona*, mientras que un agente móvil ejecuta su tarea, el usuario puede ir ejecutando otra, de tal manera que después de un tiempo el resultado del agente móvil sea enviado al usuario.

Agentes de información/internet: realizan la tarea de administrar, manipular o recolectar información proveniente de varias fuentes distribuidas. Los agentes de información pueden ser estáticos o móviles, no cooperativos o sociales y pueden o no aprender.
Agentes híbridos: son aquellos que en su funcionamiento tienen la combinación de dos o más de las capacidades de los tipos mencionados.

Desde al punto de vista de los agentes móviles: dado que un agente móvil es capaz de navegar por diferentes redes, puede ser mas eficiente y barato que un agente estático en la búsqueda de información, pues visitaría cada host para realizar sus operaciones y regresar finalmente con toda la información y operaciones realizadas, a la computadora que lo originó.

Existen diferentes metodologías para el diseño de Sistemas Multiagentes, entre las que se puede citar a INGENIAS, Vowel Engineering, MAS-CommonKADS, Mase, ZEUS, y GAIA entre otras, pero para el desarrollo de un sistema de agentes móviles Nava propone los siguientes modelos:
Modelo de Agente. Este modelo define la estructura interna del agente como parte del agente móvil, en esencia define las características de autonomía, aprendizaje y cooperatividad.
Modelo de Ciclo de Vida. Define los diferentes estados de ejecución de un agente móvil.
Modelo Computacional. Define las habilidades computacionales de un agente, como la manipulación y control de instrucciones.
Modelo de Seguridad. Se encarga de dos protecciones: la de los agentes sobre un nodo y la de los nodos sobre los agentes.
Modelo de Comunicación. Es la implementación de un protocolo.
Modelo de Navegación. Representa los aspectos de movilidad.

Sin embargo para el diseño del sistema se utilizó UML, por no ser objetivo del trabajo el desarrollo de los diferentes modelos planteados.

I.2 Plataformas para el desarrollo de Agentes y Sistemas MultiAgentes

Las plataformas son herramientas de programación para la construcción de agentes. Existen diversas plataformas en las que se pueden construir los sistemas de agentes móviles, la mayoría de las plataformas de agentes móviles están basadas en Java [9]:

JADE	Desarrollada en Italia, cumple con las especificaciones FIPA. Las aplicaciones de este framework se pueden ejecutar en un amplio rango de ambientes, entre ellos en dispositivos móviles (PDA's y celulares). Es de notar que esta aplicación, como la gran mayoría de las desarrolladas en Java, poseen librerías adicionales o se integran fácilmente con otras como JESS (Java Expert System Shell); esta librería permite representar el conocimiento heurístico de un experto humano.	
ADK	El kit de desarrollo de agentes (ADK) es una plataforma comercial de agentes que se enfatiza en los aspectos de mobilidad y seguridad. Es utilizado en muchos proyectos comerciales especialmente para la integración de sistemas de herencia.	
Agent Factory	Agente Factory se ha desarrollado como parte de una investigación en la Universidad de Dublín, que trata de crear "framework" cohesivo que apoya un acercamiento estructurado para el desarrollo y despliegue de aplicaciones orientadas a agentes.	
Agent Toolkit Java Edition	Esta herramienta permite desarrollar sistemas multi-agentes con acceso multi-protocolo, permite además administrar información común a través de SNMP (Simple Network Management Protocol), RMI, HTTP, CORBA y TL1 (Protocolo definido dentro de la herramienta y basado en XML). Este software es uno de los más llamativos por permitir implementar el acceso por varios protocolos de una manera muy intuitiva y transparente.	

AgentBuilder	Herramienta de software integrada que permite desarrollar rápidamente agentes de software inteligentes y aplicaciones basadas en agentes. Características: • Compatible con Java 1.4 • Windows 98/ME/NT/2000/XP, Solaris, Linux. • Comunicaciones: KQML, TCP/IP, CORBA. Versiones:
FIPA-OS	Como plataforma para el desarrollo de agentes, cumple con el estándar FIPA para el diseño e implementación de sistemas multi-agentes, así como permite desarrollos en Java. Está centrada en proveer mecanismos de comunicación entre agentes. Carece de la funcionalidad del manejo del conocimiento para lo cual se apoya en JESS (Java Expert System Shell) y del manejo de la movilidad de agentes (Mikko00). Este framework se puede integrar con CATNAgentToolkit para el diseño de comportamientos sociales, aunque es dispendioso establecer las relaciones necesarias. Entre las herramientas para la construcción de agentes evaluadas en este estudio, se puede decir que ésta es una de las más completas y más favorables para trabajar, ya que cuenta con una documentación completa tanto para el desarrollador de sistemas multi-agentes, como para los desarrolladores que deseen realizar aportes al código existente por ser una aplicación de código abierto.
MadKit	Para esta plataforma los agentes están caracterizados por roles y grupos. Como pasos necesarios para la implementación en esta plataforma se dota a cada agente con una capacidad determinada, esta capacidad determina el rol que el agente cumple dentro del grupo en el que se desenvuelve. Por ejemplo: el termostato posee el rol de mantener la temperatura deseada con relación a la actual; el sistema tiene el rol de informar la temperatura

actual y monitorear la deseada por el usuario. Una vez que el sistema comienza su ejecución, los agentes negocian los roles de acuerdo a sus capacidades dentro del grupo y de ser aceptado, el agente será el encargado de desempeñar el rol.

Grasshopper	Grasshopper es una plataforma de agentes que está especialmente interesada con la movilidad y los dispositivos móviles. Cumple con los estándares MASIF y FIPA.	
April	La plataforma de agentes April (AAP) es una plataforma ligera que cumple con FIPA y que ha sido escrita en el lenguaje de interacción de procesos de agentes (April).	
Lost Wax	Lost Wax ha creado uno de los primeros sistemas diseñado específicamente para las empresas comerciales. El framework de agentes Lost Wax proporciona un ambiente poderoso y fácil de usar para el diseño, desarrollo y despliegue de sistemas multiagente. Ha sido diseñado desde el inicio de forma escalable y permite la ejecución paralela de centenares de procesos de agentes distribuidos a través de múltiples servidores.	
SAGE	El proyecto SAGE tiene el objetivo de desarrollar una plataforma de agentes distribuida, descentralizada, tolerante a fallos, escalable y ligera según las nuevas especificaciones de FIPA.	
A-Globe	A-globe es una plataforma de agentes diseñada para probar escenarios experimentales sobre la posición de los agentes y la inaccesibilidad de comunicación, pero también puede usarse sin estas funciones extendidas. La plataforma provee las funciones para agentes residentes, tales como la infraestructura de comunicación, almacenamiento, servicios de directorio, funciones de migración, despliegue de servicios, etc. La comunicación en Aglobe es	

	muy rápida y la plataforma es relativamente ligera.
Aglets	Aglets Software Development Kit (ASDK) de IBM, ofrece una interfaz gráfica identificada como una agencia, en la cual los agentes se ejecutan y existen; brindándole servicios de movilidad cifrando el código y los datos de un aglet (agente) utilizando el método de serialización de Java (JOS) y trasladando los agentes utilizando el Protocolo para el Transporte de Aglets (ATP). Adicional a esto, ofrece servicios de mensajería y manejo de eventos como: creación de agentes, clonación, expedición, retractación, eliminación, activación, desactivación y paso de mensaje entre agentes.
Voyager ORB	Es una plataforma para el desarrollo de agentes que se centra en el manejo de su movilidad, obviando las otras características propias de los mismos. En cuanto a seguridad, provee soporte para comunicaciones de red seguras sobre el protocolo estándar SSL, permitiendo la comunicación remota sobre un canal encriptado y autenticado. Además soporta tunneling a través de firewalls. Posee soporte de transacciones a través de interfaces, que son fáciles de utilizar y aseguran la terminación completa de las mismas (usando "two-phase commit"). Voyager permite a múltiples recursos participar en transacciones a través de múltiples máquinas virtuales.

II. Plataforma Jade

JADE (Java Agent Development Framework), es la plataforma con la que se ha desarrollado este proyecto de tesis, la cuál permite a los agentes comunicarse entre si, mediante el lenguaje de comunicación de agentes (ACL) estándar, desarrollado por FIPA, en conjunto con los desarrolladores de JADE [10].

Inspirados por la visión de que los agentes seguirán siendo solamente un sueño si no se preserva la interoperabilidad end-to-end a través de los diferentes fabricantes y operadores, en 1996 TILAB (anteriormente CSELT) promovió la creación de FIPA (Fundation for Intelligent Phisical Agents), una asociación internacional sin fines de lucro de compañías y organizaciones que comparten el objetivo y el esfuerzo para producir las especificaciones estándar para la tecnología de agentes.

Basado en el primer conjunto de especificaciones publicadas en 1997, al final del año 2002 FIPA finalmente publicó la norma. El objetivo estándar es la interoperabilidad y, como una consecuencia, este enfoca un comportamiento externo de los componentes del sistema, quedando abierta la implementación detallada y la arquitectura interna. De hecho, la arquitectura interna de JADE es única aun si esta cumple totalmente con FIPA. El estándar FIPA fue aceptado totalmente por el paradigma del agente y, en particular, define al modelo referencial de una plataforma de agente y un conjunto de servicios que deben proporcionarse. La colección de estos servicios, y sus interfaces estándar, representan las reglas normativas que permiten una sociedad de agentes existir, operar, y ser manejada.

Siendo los agentes sociables y necesitando comunicarse, el lenguaje de comunicación de Agentes es uno de los recursos principales del estándar de FIPA. El FIPA ACL está basado en la teoría del acto de conversar y en los supuestos y requisitos del paradigma de los agentes. FIPA regularizó una biblioteca extensible de 22 actos comunicativos que permiten la representación de diferentes intenciones comunicativas (como pedir, proponiendo, informando, preguntando, requiriendo una propuesta, negándose a, etc.). FIPA también definió la estructura de un mensaje que permite representar y llevar la información útil para identificar al remitente y receptores, el volumen del mensaje y sus propiedades (por ejemplo las codificaciones y el idioma de la representación), y, en particular, información útil para identificar y seguir hebras de conversación entre agentes y representar las interrupciones para la comunicación. Los modelos comunes de conversaciones también han sido definidos por FIPA, los así llamados protocolos de la interacción que les proporcionan una biblioteca de modelos a agentes para lograr las tareas comunes como delegar una acción, requiriendo una propuesta, etc., la Figura 1 representa el modelo de comunicación definido por FIPA y las relaciones entre sus elementos. Uno de los recursos principales de estas normas de FIPA es su estado de normal, definitivo y aceptado por la comunidad de agentes.

JADE incluye las bibliotecas requeridas para desarrollar aplicaciones de agentes y el ambiente de ejecución que proporcionan los servicios básicos y que deben ser activados sobre el dispositivo antes de que los agentes puedan ser ejecutados. Cada instancia de

ejecución en JADE es llamada contenedor (del hecho que él contiene agentes). El conjunto de todos los contenedores se llama plataforma y proporciona una capa homogénea que esconde a los agentes (y la aplicación desarrollada también), la complejidad y la diversidad de los equipos (el hardware, sistemas operativos, los tipos de red, JVM).

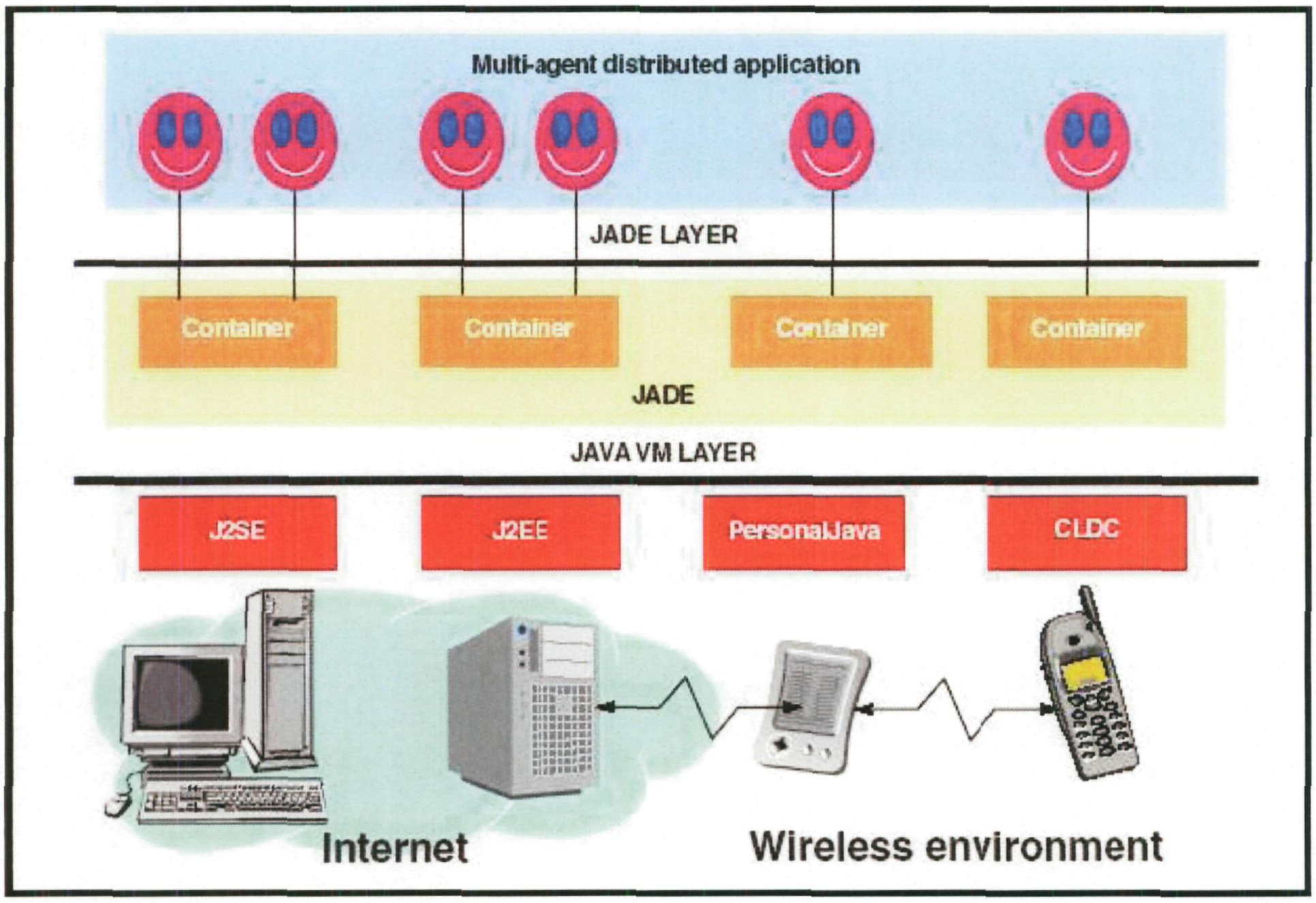


FIGURA 1: MODELO ARQUITECTÓNICO DE LA PLATAFORMA

Un contenedor único principal, "Main container" siempre debe estar activo en una plataforma y todos los otros contenedores se registran con él tan pronto como ellos empiecen.

Además de la habilidad de aceptar el registro de otros recipientes, un contenedor principal difiere de un contenedor normal en que el sostiene a dos agentes especiales (que automáticamente inician cuando el contenedor principal se lanza).

El AMS (Sistema de Dirección de Agentes) proporciona el servicio de denominación (es decir asegura que cada agente en la plataforma tiene un único nombre) y representa la autoridad en la plataforma (por ejemplo es posible crear/matar agentes en contenedores remotos solicitándolo al AMS).

El DF (Facilitador de Directorio) proporciona el servicio de las páginas amarillas que es el medio por el que un agente puede encontrar a otros agentes que proporcionan los servicios que él exige para lograr sus metas. Ver figura 2.

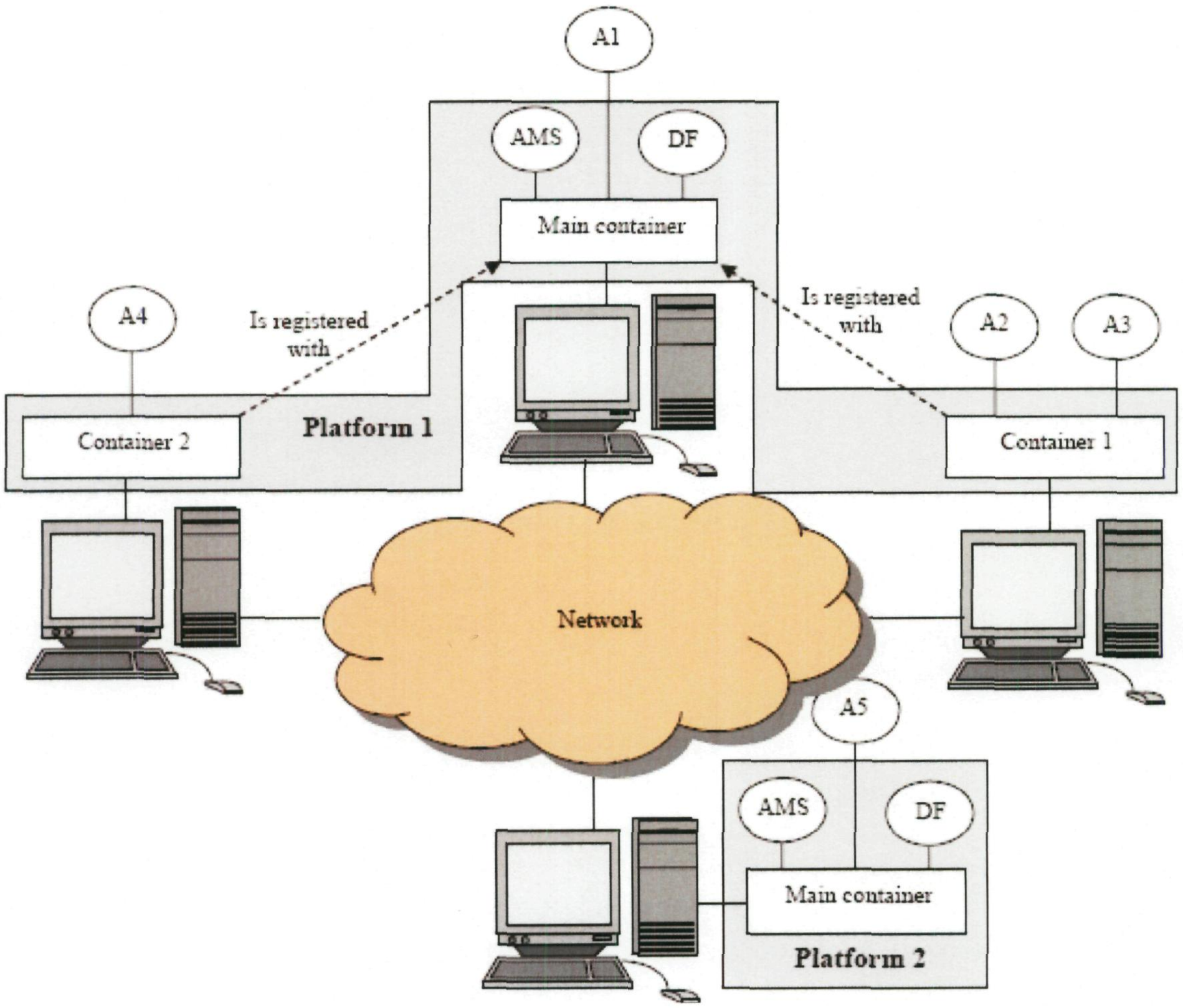


FIGURA 2. CONTENEDORES Y PLATAFORMAS

El servicio de "páginas amarillas" permite a los agentes publicar uno o más servicios que ellos proporcionan para que otros agentes los puedan encontrar y consecutivamente pueden aprovecharlos como se describe en la Figura 3.

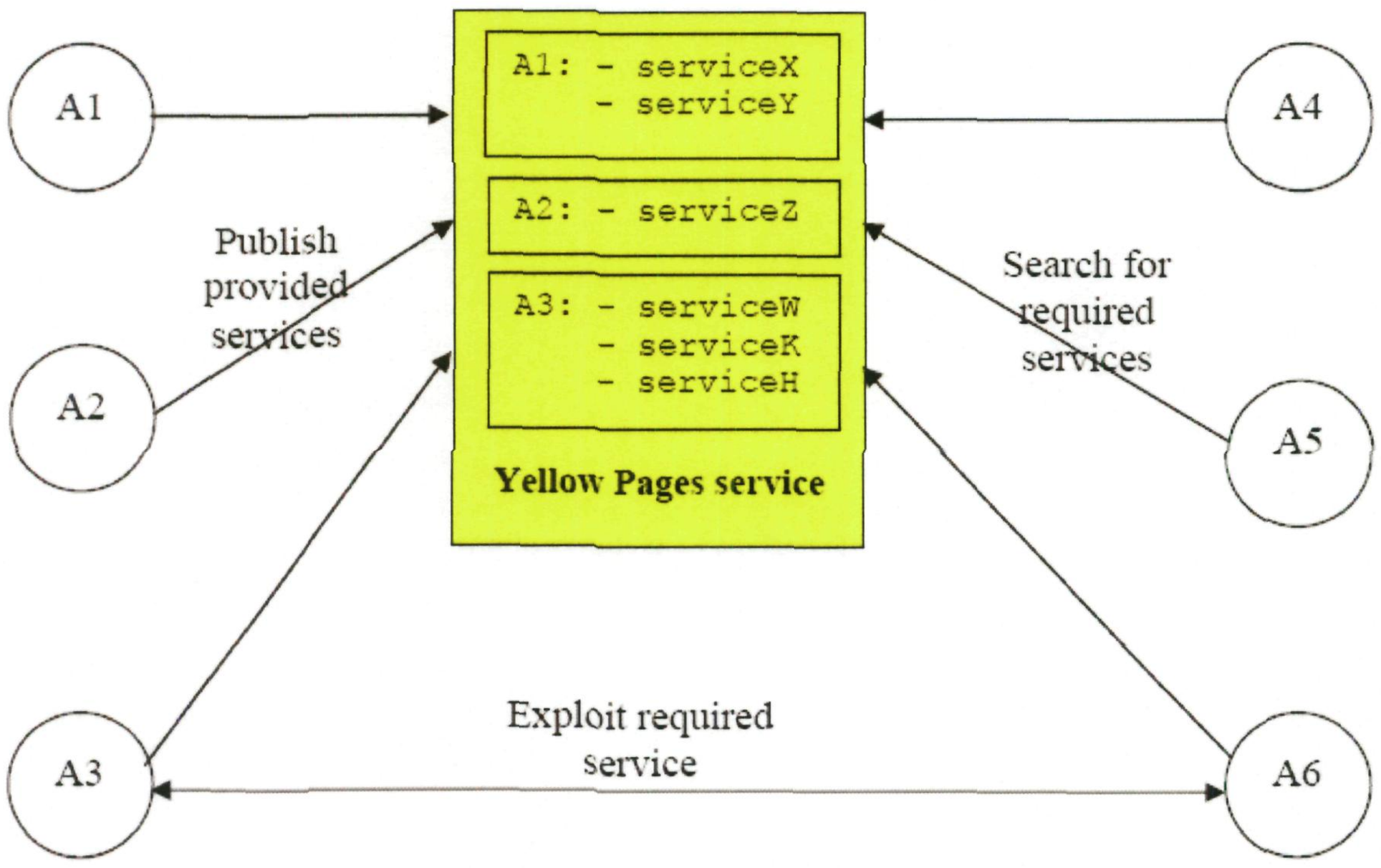


FIGURA 3. EL SERVICIO DE PÁGINAS AMARILLAS

El servicio de las páginas amarillas en JADE (según la especificación de FIPA) está proporcionado por un agente llamado DF (Facilitador de Directorio). Cada plataforma que cumple con FIPA organiza a un agente DF predefinido (cuyo nombre local es "df"). Otros agentes DF pueden activarse y varios agentes de DF (incluyendo el agente por defecto) pueden confederarse para que proporcione un sólo catálogo de las páginas amarillas distribuidas.

Siendo el DF un agente, es posible actuar con el recíprocamente como de costumbre intercambiando mensajes ACL, que usan un lenguaje de contenido propio (el lenguaje SL0) y una ontología apropiada (la ontología de FIPA de administración de agentes) según la especificación de FIPA. Sin embargo para simplificar estas interacciones, JADE proporciona la clase jade.domain.DFService que nos ofrece los servicios para publicar y buscar a través de llamadas de métodos.

II.1. Soporte de movilidad en JADE

De acuerdo con (Bellifemine et al., 2005a) [11] JADE se restringe al soporte de movilidad entre contenedores de una misma plataforma. Se pueden desarrollar agentes móviles, con capacidades de migrar o clonarse ellos mismos a lo largo de múltiples nodos de una red, lo que involucra una transición de estados en el ciclo de vida de un agente.

La movilidad puede iniciarse por el propio agente o por el AMS (de acuerdo al estándar FIPA ACL). Un agente móvil necesita conocer su localización para decidir cuando y donde desplazarse, por lo que JADE ha desarrollado una ontología (jade.mobility.ontology) con conceptos y acciones necesarias (jade.domain.mobility), proporciona un API para movilidad, la clase Agent dispone de los métodos: doMove(), recibe como parámetro un jade.core.Location, que representa el destino requerido donde debe migrar el agente.

doClone(), junto a un parámetro jade.core.Location lleva un string indicando el nombre del nuevo agente que se creará con la copia del actual. Jade.core.Location es una interfaz abstracta, es decir las aplicaciones de agentes no pueden crear sus propias localizaciones, sino que tienen que preguntar al AMS la lista de localizaciones disponibles y elegir una, un agente también le puede preguntar al AMS la localización donde está un agente determinado.

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 moverían con él, mientras que otros serían desconectados antes de salir y reconectados en el destino. JADE proporciona una serie de métodos para la gestión de recursos dentro de la clase Agent: beforeMove(), se invoca en la localización de partida antes de enviar el agente a través de la red; afterMove(), se invoca en la localización de destino tan pronto como el agente llega y se identifica en el lugar; beforeClone() y afterClone(). JADE proporciona una ontología para la movilidad, la ontología jade.mobility.ontology contiene todos los conceptos y acciones necesitadas para soportar la movilidad de agentes (jade.domain.mobility.MobilityOntology). Esta ontología extiende a JADE Management Ontology la cual se integra por cinco conceptos: mobile-agent-description, mobile-agent-profile, mobile-agent-system, mobile-agent-languaje, mobile-agent-o, y posee dos acciones: move-agent, clone agent.

El AMS puede llevar a cabo las dos acciones presentes en la jade-mobility-ontology, cada acción puede solicitarse al AMS a través de un protocolo FIPA-request con jade-mobility-ontology como ontología y FIPA-SL0 como lenguaje de contenido. La acción move-agent necesita como parámetro un mobile agent- description. Esta acción mueve el agente (identificado por name y address), a la localización destino (destination). La acción Clone-agent funciona de manera similar, pero tiene un argumento adicional de tipo string para indicar el nombre del nuevo agente que se obtiene como resultado del proceso de clonación. El AMS dentro de la movilidad también soporta otras acciones: where-is-agent, Query-platform-locations.

II.2. Entorno de ejecución en JADE

RMA (Remote Monitoring Agent)

El agente *RMA* permite controlar el ciclo de vida de todos los agentes residentes en la plataforma. Se lanza poniendo la opción *-gui* cuando ejecutamos un ejemplo escrito en Java.

El agente *RMA* es en realidad un objeto Java, una instancia de la clase *jade.tools.rma.rma* por lo que puede ser también lanzado desde la línea de comandos como cualquier otro agente (*java jade.Boot myConsole:jade.tools.rma.rma*).

Entre las funcionales que proporciona el agente *RMA* se destaca la posibilidad de crear, suspender y matar un agente, así como clonarlo y moverlo hacia otro contenedor. Ver figura 4.

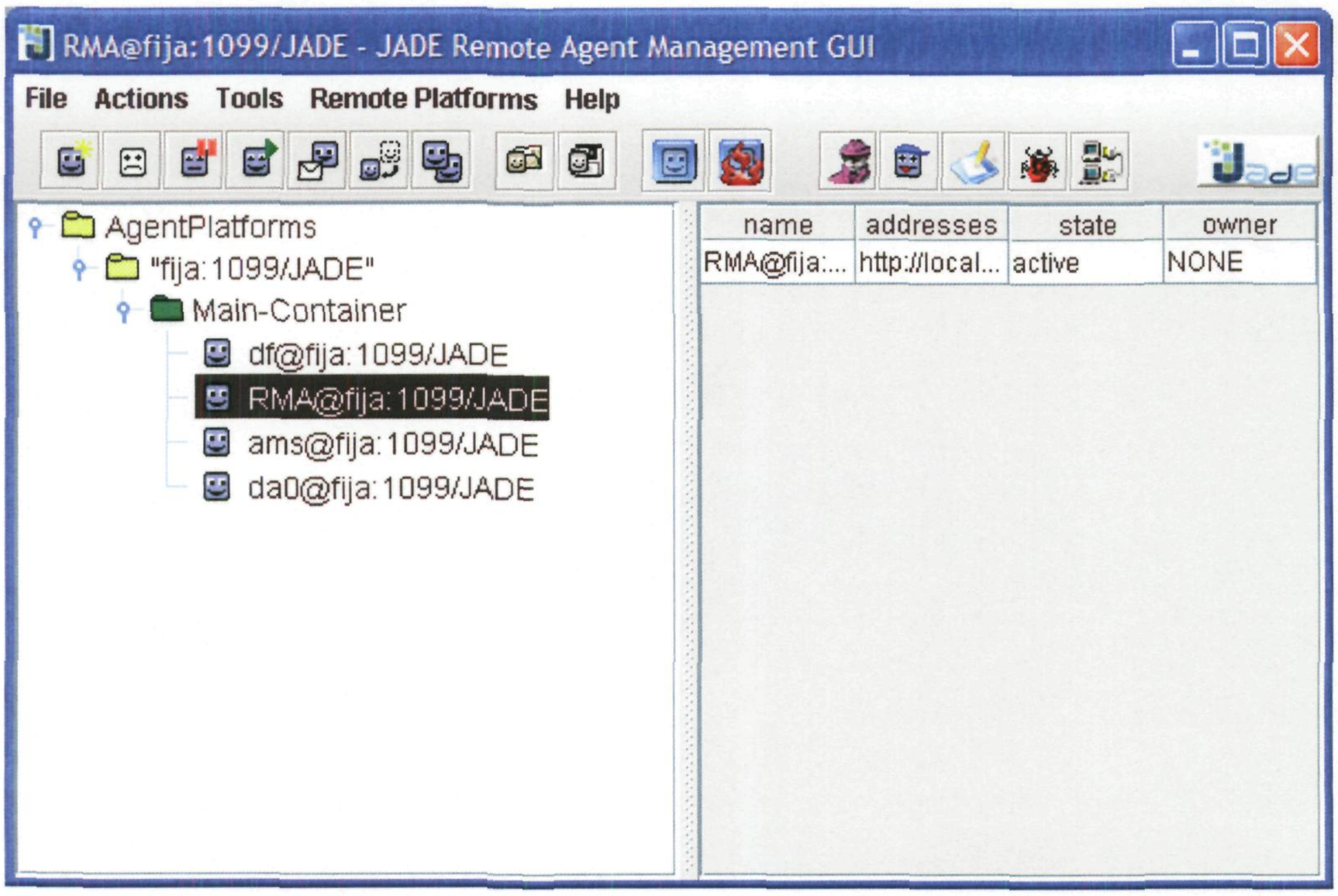


FIGURA 4. AGENTE RMA

II.3. Directorio de agentes y servicios

Existen dos situaciones diferentes en las cuales se puede necesitar obtener la localización de un agente, la primera se da cuando, sabiendo el nombre de un agente, no sabemos su localización en el sistema. La otra se presenta cuando un agente necesita invocar un servicio concreto en otro agente y, en principio, no sabe que otro agente puede ofrecerlo.
Para hacer frente a la primera situación, todo agente FIPA [12] debe darse de alta en un directorio junto con una dirección en donde poder localizarlo. Así, otro agente que desee contactar con él podrá, accediendo al directorio, obtener su localización. De la misma forma, un agente que quiere ofrecer sus servicios ha de subscribirse al directorio de servicios mediante el servicio de directorio de servicios. Los datos de su suscripción incluirán un identificador único globalmente, el tipo de servicio dentro de una categoría y un conjunto de pares atributo-valor describiendo el acceso al servicio y otras características de interés.

El siguiente fragmento de código ilustra como un agente que ofrece un servicio de concatenación de cadenas de caracteres, se subscribe con una descripción del mismo en el facilitador de directorio.

```
public class ConcatAgent extends Agent
{
...
public void setup()
{
...
ServiceDescription s = new ServiceDescription();
s.setName("ConcatStrings");
s.setType("Strings");
Property p = new Property();
p.setName("MaxLengthLeftString");
p.setValue("25");
s.addProperties(p);
p.setName("MaxLengthRightString'');
p.setValue("25");
s.addProperties(p);
DFAgentDescription df = new DFAgentDescription();
df.addServices(s);
try{ DFService.register(this,df); }catch(FIPAException e){
e.printStackTrace(); }
...
}
...
}
```

El concepto de servicio es algo abstracto y FIPA no trata de estandarizar los mecanismos para su invocación completa. La invocación de un servicio puede estar implícita en un acto comunicativo como por ejemplo en query-if. Es en el cuerpo ontológico utilizado en donde

debería quedar definido plenamente cada uno de los servicios usados. Cuerpo ontológico que deberían conocer tanto el agente que ofrece el servicio como el que lo invoca.

La interacción entre agentes se realiza a través de diferentes protocolos FIPA, estos son Una de las propiedades que distingue a los agentes de los objetos es que pueden realizar interacciones complejas, en lugar del simple paso de mensajes que realizan estos son, por ejemplo, FIPA-query, FIPApropose, FIPA-brokering, etc. Su funcionamiento básico es similar en todos. El agente iniciador envía un mensaje. Posteriormente el receptor (responder) responde bien con un not-understood, bien con un refuse ó agree con lo que proporciona una respuesta inicial a la petición. Una vez que el receptor ha llevado a cabo la acción, envía un último mensaje al iniciador del tipo inform ó failure para notificar el resultado. En nuestro caso particular no se realiza interacción entre agentes, el agente desarrollado migra a un host y aplica una técnica de Recuperación de Información, a continuación se explica la teoría adecuada que permite seleccionar una técnica en particular.

III. Recuperación de la información

El procesamiento de Lenguaje Natural (PLN) es un área de investigación orientada a desarrollar sistemas de cómputo que coadyuven a la solución de problemas lingüísticos y computacionales. Los problemas que se plantean en PLN cada vez son más importantes, y no sólo es por el hecho de saber que la cultura de la humanidad está en buena medida contenida en los textos, sino porque las necesidades actuales de procesamiento de información demanda la ayuda de nuevos mecanismos que enfrenten el fenómeno de la "informatización" de la sociedad [14].

La WWW tiene un crecimiento que no está en relación proporcional a su disponibilidad "efectiva", puesto que es requisito el procesamiento de dicha información para mejorar su aprovechamiento [14]. El crecimiento que está experimentando la Web supera las expectativas de los más optimistas. No obstante lo verdaderamente relevante no es la cifra, sino el crecimiento que la Web viene experimentando desde su nacimiento. En un dramático crecimiento debido a nuevos sitios en Internet de negocios pequeños y sitios que recopilan información cronológicamente de textos y/o artículos (blogs), la firma de investigación de Internet Netcraft ha reportado que el número total de sitios en Internet ha sobrepasado la marca de los 100 millones. La investigación realizada por Netcraft en Noviembre del año 2006 revela que 3.5 millones de sitios surgieron durante el mes de octubre, llegando a la increíble cantidad de 101, 435,253, aunque el número de sitios que actualmente están activos es de aproximadamente 51 millones. La compañía atribuyó el reciente crecimiento a servicios de hospedaje de sitios en Internet que recopilan información cronológicamente de textos y/o artículos (blogs) de las compañías Microsoft y Google. En tan sólo dos años, el Internet ha duplicado su tamaño, después de la marca de 50 millones establecida en Abril de 2004. El crecimiento de Internet tiene un comportamiento exponencial, obteniendo 20 millones de sitios en Internet en los últimos 7 meses del año 2007[15].

El problema de acceso a la información ha sido tratado por las Bases de Datos y la RI. La primera se dedica a responder consultas sobre información almacenada de manera estructurada, la segunda, tiene la misma función pero sobre información no estructurada, como son los textos. Si bien la Recuperación de información ha mostrado éxito, ésta no mantendría su efectividad en el Internet pues el volumen de documentos es significativamente mayor al que esta área trabaja normalmente. No por ello se descarta el empleo de RI, sino que, por el contrario, se buscan soluciones para la variedad de problemas que se presentan en la RI. Por lo mismo también se ha generado una difícil manipulación aún para sistemas de búsqueda de información tan complejos como Google [16].

La importancia de estos problemas ha generado propuestas que conducen al establecimiento de más líneas de investigación, particularmente la extracción de información, que trata, de concebir a un conjunto de documentos como una base de datos formada por el resultado de la extracción, según un patrón de hechos o eventos que se encuentran en los mismos.

También es subsidiaria a la clasificación de documentos la "elaboración del resumen de un texto". El empleo del resumen de un documento es atractivo, pues reduce el espacio de almacenamiento y facilita el acceso de información relevante. El resumen automático de documentos comparte o puede compartir propiedades con la obtención y la extracción de información.

De hecho las dos grandes líneas de investigación en el campo del resumen se corresponden por analogía con estas dos tecnologías. Entendemos por obtención (o recuperación) de información lo que por ejemplo, hacen los buscadores de Internet: dado un conjunto enorme de documentos, obtener aquellos que corresponda a determinados criterios; en general, ciertas palabras claves. Por otra parte, la extracción de información consiste en tratar uno o varios documentos para extraer de ellos una determinada información que nos interesa y generar a partir de ésta un nuevo documento que contenga únicamente esta información relevante. Las dos líneas básicas de investigación en la confección de resúmenes automáticos son análogas a éstas. Los resúmenes por extracción actúan sobre uno(o varios) documentos, vistos como una colección de oraciones; y de estas oraciones se extraen y presentan aquellas consideradas más relevantes o que responden a determinados criterios. En este caso el resumen por extracción es un subconjunto de las oraciones del texto original [17].

Los resúmenes por abstracción utilizan técnicas más sofisticadas de tratamiento del lenguaje, ya que el resultado no consiste en determinadas oraciones entresacadas del texto original, sino en un documento de nueva redacción generado a partir del tratamiento de la información contenida del primero. Para conocer el estado actual con respecto a la confección de resúmenes de documentos se debe atender primeramente a los sistemas comerciales existentes, y por otro lado al estado de la investigación. En el primer caso se debe acudir a los resumidores de Inxight(Xerox) [18], al más reciente Copernic [17], o al típico y elemental sistema de autorresumen de Microsoft Word[17]. En cuanto a sistemas en vías de desarrollo que operen en línea, se puede probar algunos muy interesante como es el caso de SweSum [19] y Extractor [20], que ofrecen resúmenes en el idioma español. Por lo que respecta al estado de la investigación, son muy relevantes las páginas de la Universidad de Ottawa [21], la del investigador de la Universidad de Columbia Dragomir Radev[22]. En el Estado Español, se está construyendo un prototipo de sistema resumidor de noticias periodísticas en el marco del proyecto de Hermes [23]. Por otro lado también se presenta un sistema resumidor para textos en castellano que combina técnicas clásicas dentro del campo del resumen automático con otras menos frecuentes, como son la detección de anáforas y de marcadores discursivos. Pero los programas de mayor envergadura hay que localizarlos en el marco de las acciones impulsadas por el Departamento de Defensa de los EE.UU.(DARPA), como el programa TIDES para la detección, extracción y resumen de información multilingüe [24].

III.1. Resumen por Extracción

El proceso de generación del extracto de un documento consiste en obtener oraciones pertenecientes al mismo documento consideradas como de mayor relevancia [25]. Para la elaboración de este trabajo se ha tomado como referencia una recopilación de trabajos previos de extracción, como es el caso de [26] en donde se presenta una propuesta para generar el extracto de un texto. Este trabajo principalmente se basa en el uso de una técnica denominada "expansión por corpus" que sirve para expandir cada término de un documento basándose en su sentido. Las oraciones expandidas son procesadas, después se usa una función de similitud para la obtención de oraciones más significativas que constituyen el extracto del texto. En los resultados expuestos en [27], se utilizan dos técnicas para la obtención del extracto de un texto y posteriormente se realizan una serie de consultas sobre los extractos para evaluar los resultados. La idea es innovadora, sin embargo, la cantidad de documentos utilizados es sumamente pequeña y adicionalmente la evaluación de los juicios es realizada únicamente por tres jueces. Los resultados presentados en ese trabajo son interesantes, valdría la pena verificar el comportamiento de los algoritmos con un corpus mucho más grande. Una aportación más se encuentra en [28], donde se realiza una comparación de dos métodos que determinan automáticamente el extracto de un texto; el objetivo de ese trabajo es hacer patente la importancia del título de un texto. Se obtienen extractos de un texto usando funciones de similitud entre todas las oraciones (método uno) y usando una función de similitud entre el título del texto y las oraciones restantes (método dos). Los resultados muestran que ambos resultados son semejantes, sin embargo, el método dos se encuentra en un orden de complejidad lineal a diferencia del primer método que se encuentra en uno cuadrático. Como se puede observar, existen diversos trabajos en generación automática de extractos y con resultados interesantes, que alientan a continuar investigando. Sin embargo hay que establecer primero, técnicas eficaces para la obtención de extractos que permitan migrar hacia la generación de resúmenes. En este sentido, se pretende ir en aumento con trabajos realizados. Hasta el momento se presenta un estudio activo de propuestas dentro de esta área. En nuestro caso se ha optado por proponer un método basado en la extracción de párrafos "significativos", a partir de la similitud de cada párrafo de un documento con lo que se ha denominado "el párrafo virtual" (PV), el cual a su vez es obtenido a través de una técnica llamada el punto de transición (PT).

El Párrafo Virtual

Uno de los propósitos del resumen es proporcionar los temas interesantes de un texto. Diversos métodos se han aplicado a través del tiempo con la finalidad de generar resúmenes automatizados. Un enfoque ha sido identificar las frases significativas dentro del mismo texto para realizar el proceso de generación del resumen. El párrafo virtual se puede construir a través de términos significativos del mismo documento extraídos mediante alguno de éstos tres métodos: el Punto de Transición, términos semejantes al Título, Palabras Claves, los cuales se explicarán a continuación:

El uso del Punto de Transición (Técnica del Punto de Transición)

El PT surge a partir de las observaciones de George Kinsley Zipf, quién formuló la ley de frecuencias de palabras de un texto (Ley de Zipf), donde se establece que las palabras con mayor frecuencia absoluta son las palabras cerradas, mientras que las menos frecuentes son aquellas que reflejan el estilo y riqueza del vocabulario y por último las que aparecen en la zona media de la función de distribución de frecuencias son las que representan los documentos [29]. 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 Punto de Transición es la siguiente:

$$PT = \frac{\sqrt{1+8*I_1}-1}{2} \qquad (3.1)$$

Donde I_1 representa el número de palabras que tienen frecuencia 1.

III.2. Método para la Generación de Extractos mediante el método del Punto de Transición

Se propone para cada documento:

- Preprocesamiento. Eliminar las palabras cerradas (artículos, preposiciones, signos de puntuación, etc.), y lo partición en párrafos.

- Obtención del vocabulario. Calcular la frecuencia de ocurrencia de cada término no repetido que posea el documento preprocesado.

- Generación del párrafo virtual. Aplicar la modalidad para documentos pequeños del PT, tomando el término con la frecuencia más baja que no se repite de los términos del vocabulario de cada documento.

- Determinación de párrafos significativos. Se utiliza la función de similitud de Jaccard ecuación 3.2 usada comúnmente en el modelo booleano de representación de información [30], para determinar el grado de similitud de dos elementos. En este caso se compara cada párrafo del documento con el párrafo virtual que se obtuvo a través de PT. Los extractos de cada documento se conforman mediante los tres párrafos más relevantes.

$$sim(D,q) = \frac{\#(D \cap q)}{\#(D \cup q))} \qquad (3.2)$$

Formalmente se puede formular el siguiente algoritmo:

Algoritmo

ENTRADA: Texto

SALIDA: Extracto

Paso 1: Dividir el Texto en Párrafos

Paso 2: Eliminar palabras cerradas

Paso 3: Obtener el vocabulario V de un D ordenado i.e.,

$$V_0 = [(t_1, f_1),...,(t_n, f_n)], \text{ con } f_i \geq f_1 - 1, \text{ entonces}$$

Paso 4: Obtención del párrafo virtual

$$Pt = f_i - 1, \text{ si } f_i = f_i + 1$$
$$Pv = t_j \mid j \leq N, (t_j f_j \in V)$$
Tomando un 25 % de $f_i(t_i, f_i) \in [Pt * 0.75, Pt * 0.25]$

Paso 5: Obtener la similitud del Pv y O_i

Tomar el Pv de V
Calcular la $sim(Pv, O_i) = (O_i \cap Pv)/(O_i \cup P_v)$

Solo se toman las 5 oraciones con mayor grado de similitud.

IV. Diseño del agente

Para el diseño del agente se utilizó UML, desarrollando los diagramas de casos de usos para capturar los requerimientos, los diagramas de clases, y los diagramas de secuencias. Es necesario destacar que se definieron todas las metas y tareas que deberá aplicar el agente móvil.

Los componentes principales de sistema se ilustran en la figura 5.

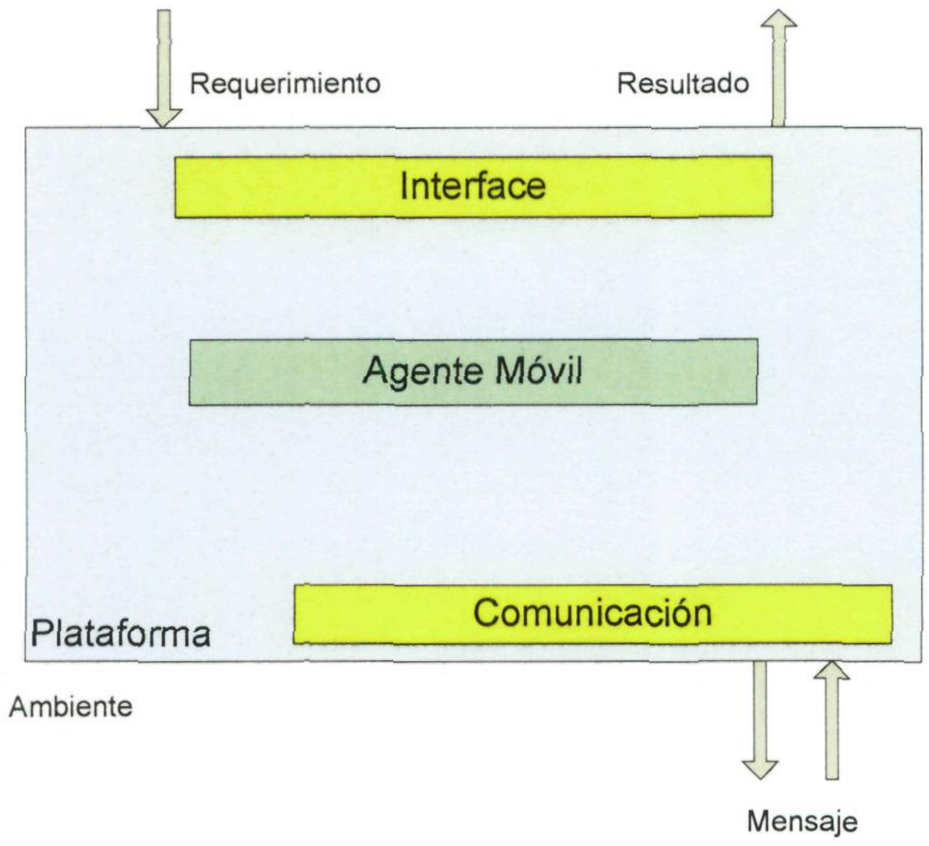


FIGURA 5. COMPONENTES DE LA ARQUITECTURA DEL SISTEMA

- Interface: Muestra información importante al usuario, y también recibe los requerimientos de este. También muestra 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 migrar a diferentes nodos y con ello satisfacer las preferencias de búsqueda de un usuario.
- Módulo de comunicación: Se encarga de enviar y recibir los mensajes entre los agentes mediante protocolos de transporte, Jade utiliza para ello RMI.

Casos de uso

Las funcionalidades del sistema para el control de los agentes que se construyó necesita una interacción grande del usuario, para realizar sus funciones, las tareas de clonación y migración son transparentes para el usuario, por lo que a continuación se mencionan las tareas principales de nuestro actor, el usuario:
El usuario necesita introducir la consulta que desea realizar, de una lista de posibles ubicaciones que necesita actualizar antes de dar su consulta selecciona uno de los host,

finalmente lanza un clon del agente a ese host. El puede seleccionar otro host pero debe mandar la misma consulta hasta que todos los clones que haya enviado estén de regreso.
Al regreso de los clones, el usuario puede accionar la visualización de los resultados, o puede revisar en una carpeta especial los resultados obtenidos de sus consultas.
Para poder realizar estas acciones, el usuario debe interactuar con la interfaz del agente móvil, que fue diseñada en modo gráfico para un mejor manejo, ver figura 6.

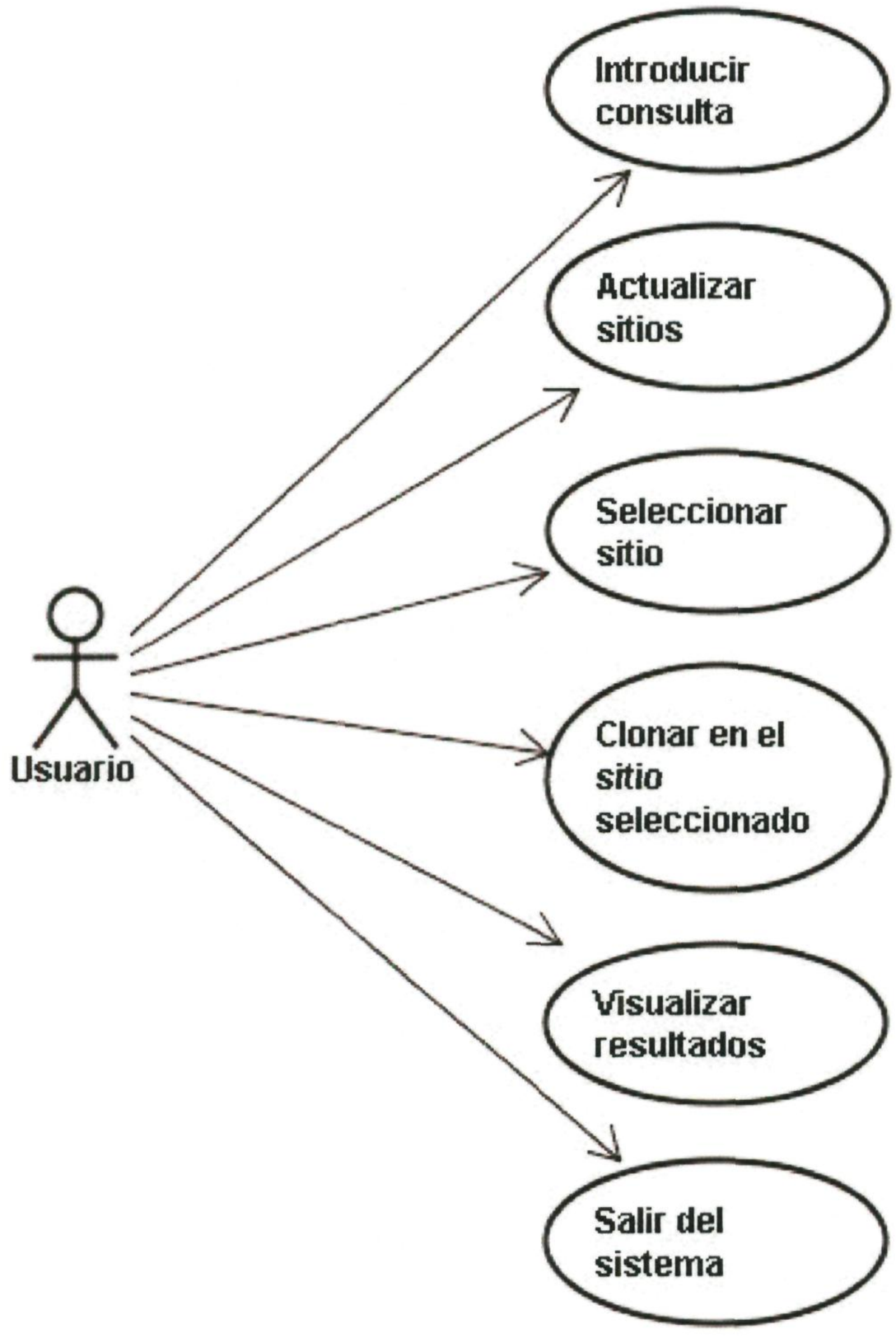


FIGURA 6. FUNCIONES PARA EL USUARIO

Con los casos de usos quedan capturados los requerimientos, a continuación se presentan las metas correspondientes del agente móvil.

Metas y Tareas del Agente

Al principio cuando se definía al agente, siempre se utilizó la palabra tarea, un agente siempre realiza tareas, casi de forma autónoma, de estas tareas logran metas. La funcionalidad del agente se determina mediante las tareas que sabe realizar, asignadas

directamente o como resultado de su participación. Mediante estos elementos se explica cómo se alcanzan las metas del agente.
Para diseñar el agente fue necesario identificar sus metas, así como las tareas asociadas a cada meta, ver tabla 7.

Metas del Agente Móvil	**Tareas**
1. Procesar la consulta	• Recibir y formatear consulta del usuario.
2. Clonarse en los nodos activos.	• Identificar nodos activos. • Migrar hacia los nodos activos. • Autentificarse en cada nodo. • Se obtiene el nombre del clon y la máquina a la que migró. • Se desactivan los clones una vez que se ha enviado la información al nodo origen.
3. Elaborar resúmenes	• Seleccionar las páginas html. • Preprocesar los documentos. • Elaborar resumen mediante PT. • Elaborar mensaje (nombre de clon, nombre máquina, nombre del archivo, título, resumen).
4. Enviar información recuperada al usuario	• Enviar mensaje con la información obtenida.
5. Desplegar la información al usuario	• Dar formato a la información recuperada de modo que sea de fácil lectura para el usuario.
6. Almacenar el historial de consultas	• Almacena datos relevantes de acuerdo a la consulta del usuario.

TABLA 7. METAS Y TAREAS DEL AGENTE MÓVIL

El siguiente caso de uso describe las metas del agente móvil

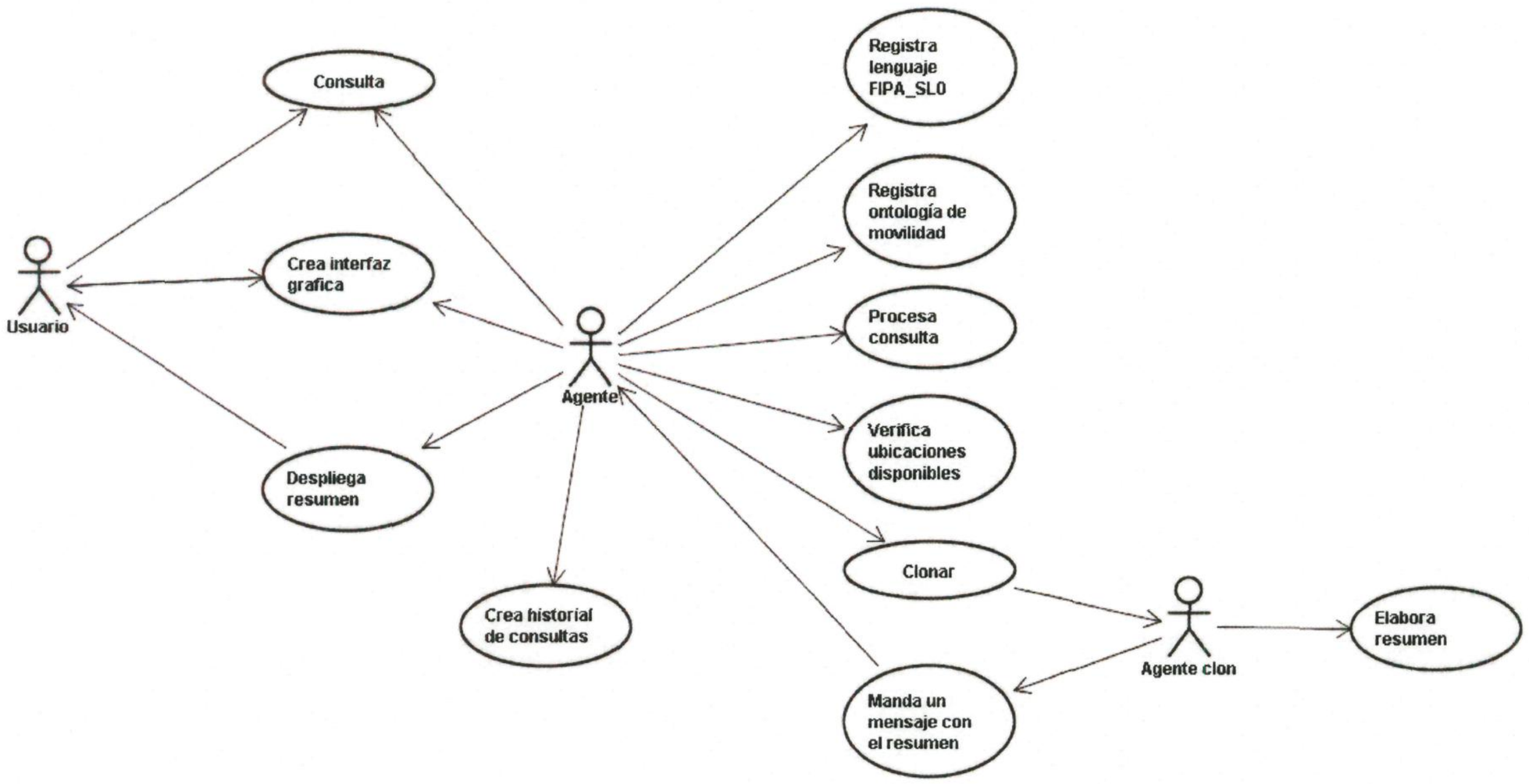


FIGURA 8. CASO DE USO DE LAS METAS DEL AGENTE MÓVIL

Una vez definidas todas las tareas y construidos los diagramas de casos de uso se está en condiciones de desarrollar el diagrama de clases, que se muestran en la figura 9.

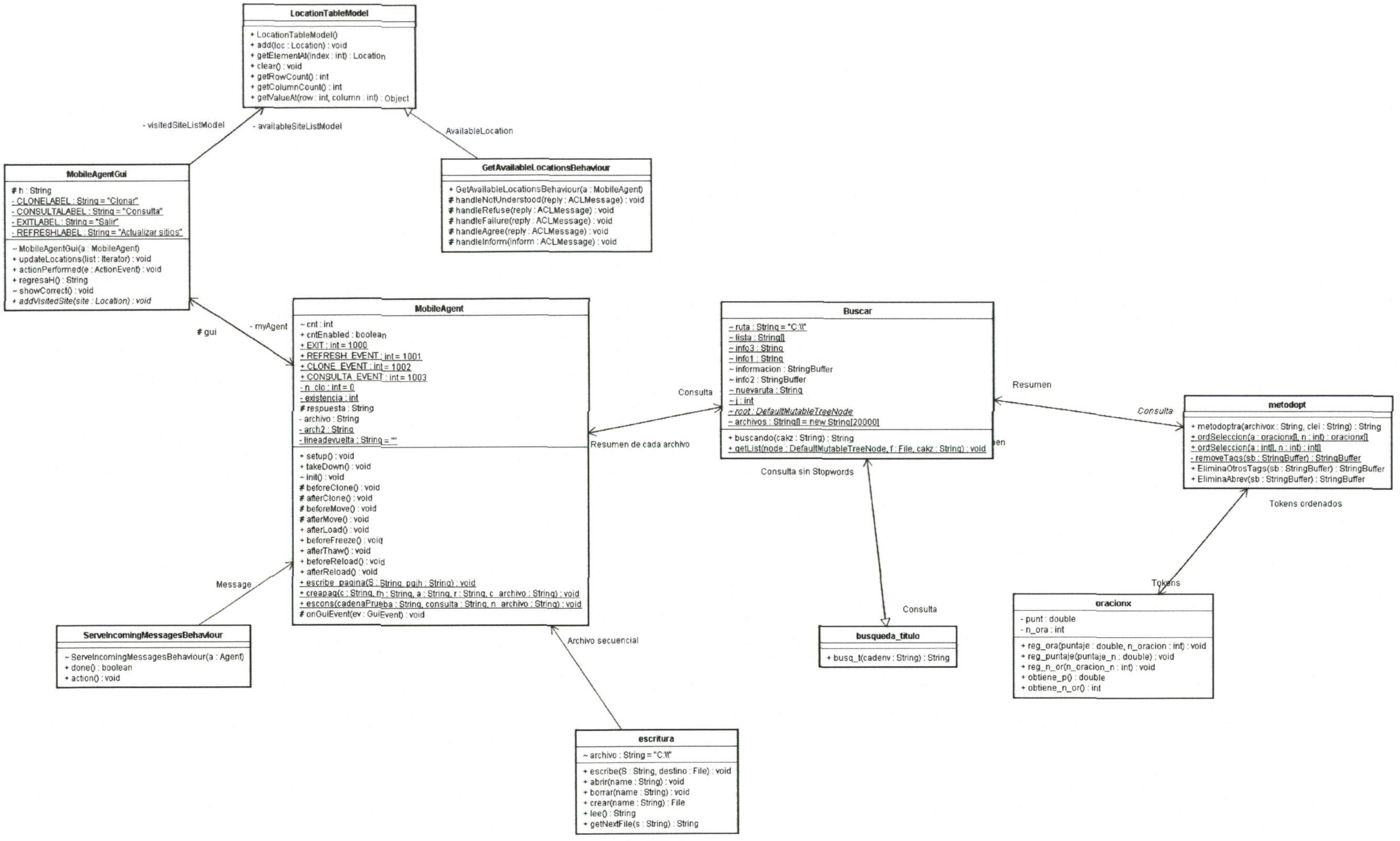
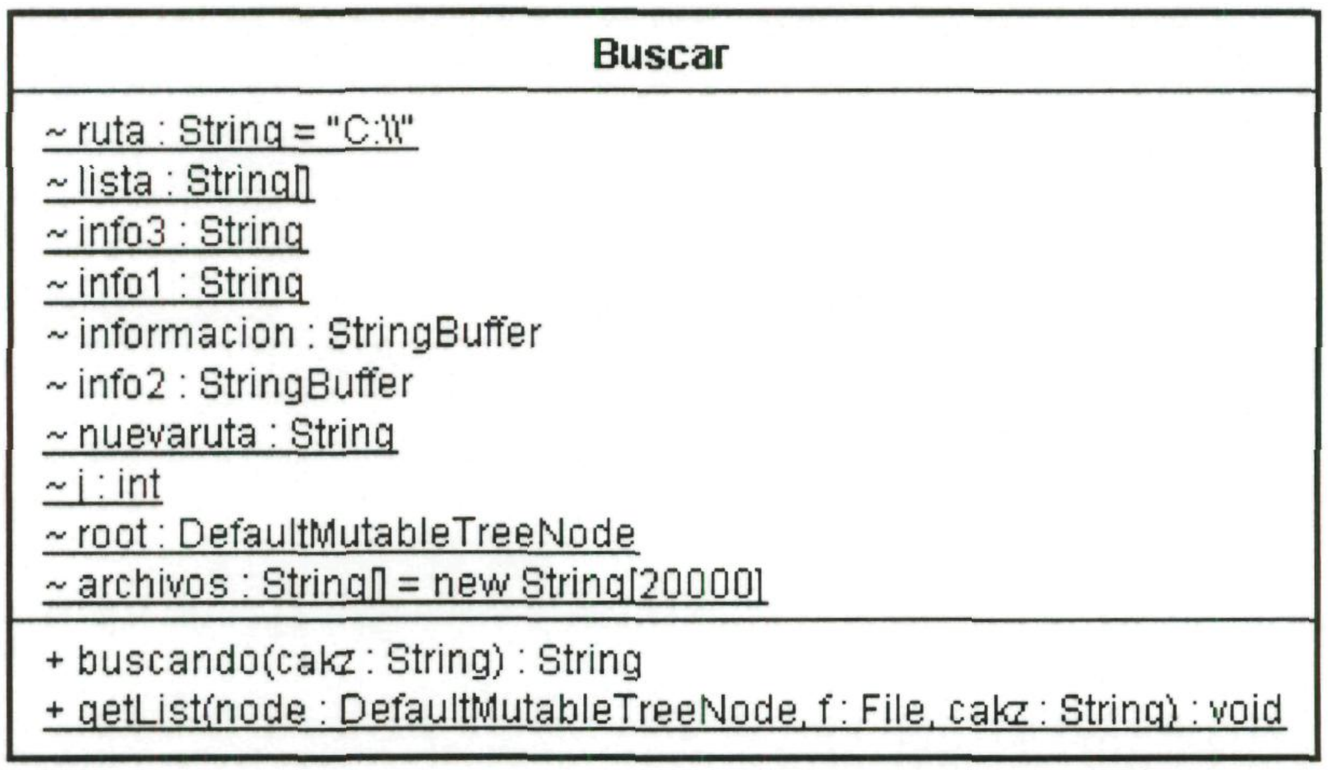
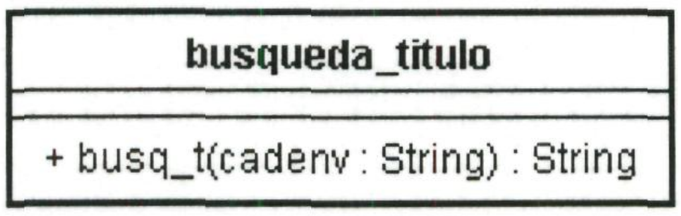


FIGURA 9. DIAGRAMA DE CLASES

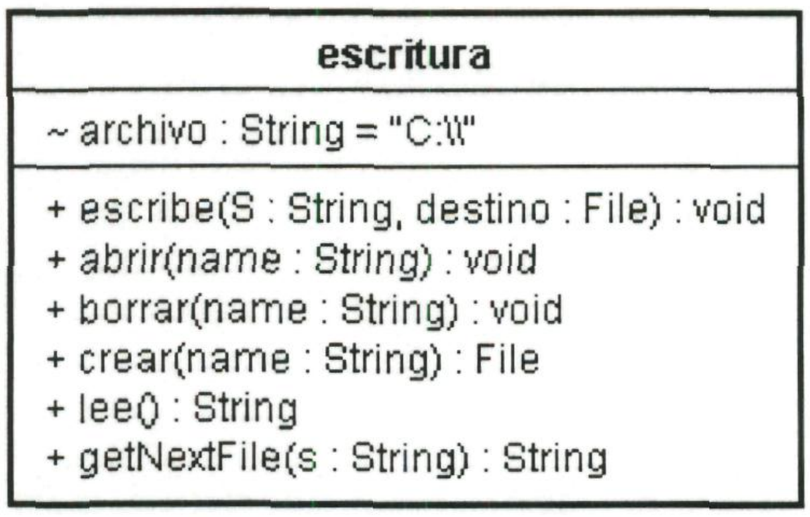
A continuación se dará una explicación general de cada una de las clases:
Clase buscar
La clase buscar es la que contiene los métodos necesarios para buscar todos los archivos html de los nodos en los que se realizará la consulta.



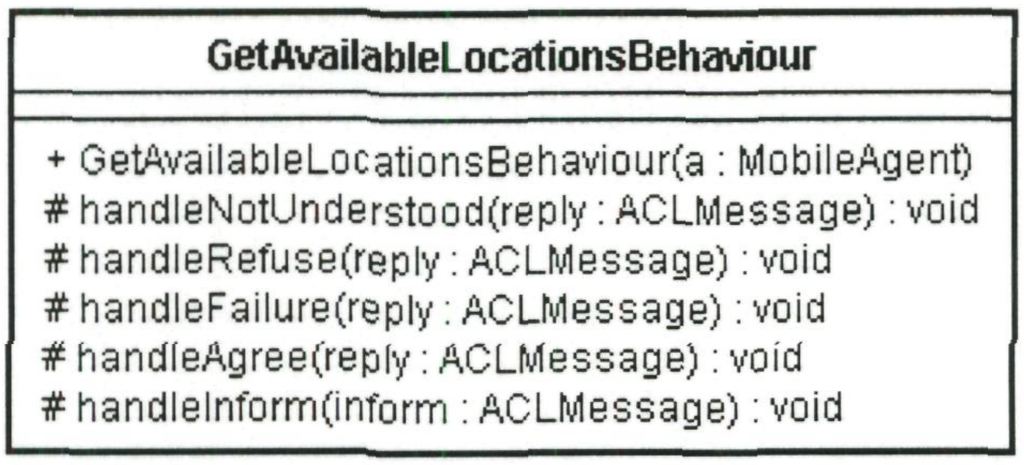
Clase búsqueda título
La clase búsqueda título descompone la consulta del usuario y elimina los stopwords de ella para hacer la búsqueda eficiente.



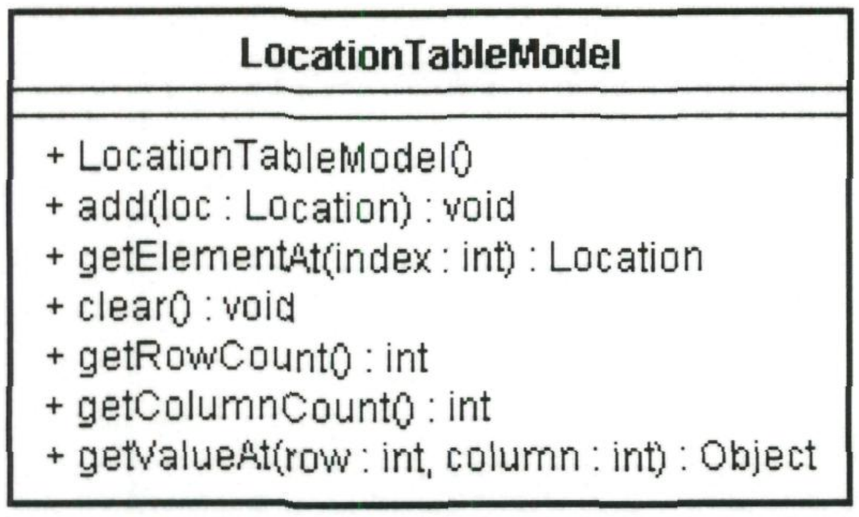
Clase escritura
La clase escritura tiene los métodos necesarios para hacer una secuencia de archivos cada vez que se realiza una consulta.



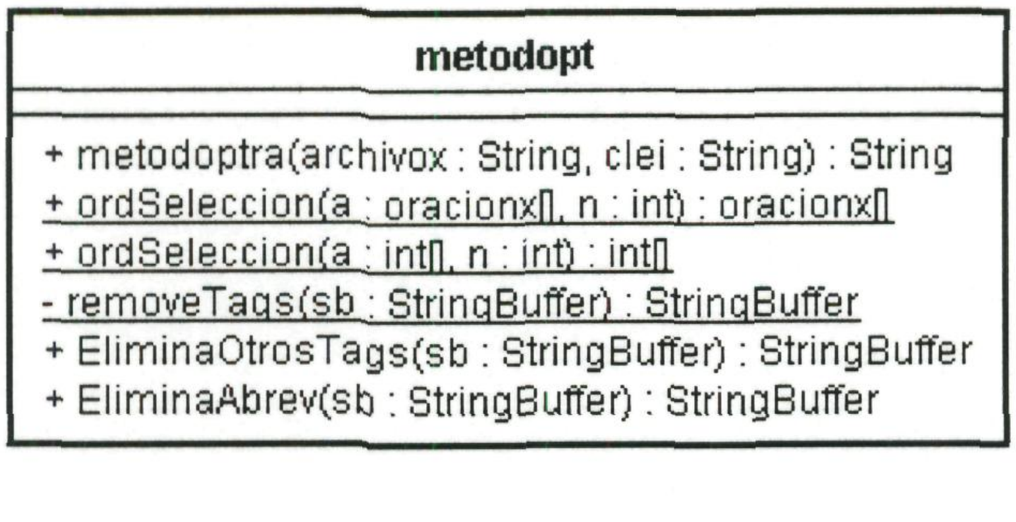
Clase GetAvailableLocationsBehaviour
La clase GetAvailableLocationsBehaviour solicita al AMS la lista de ubicaciones disponibles donde el agente se puede mover.



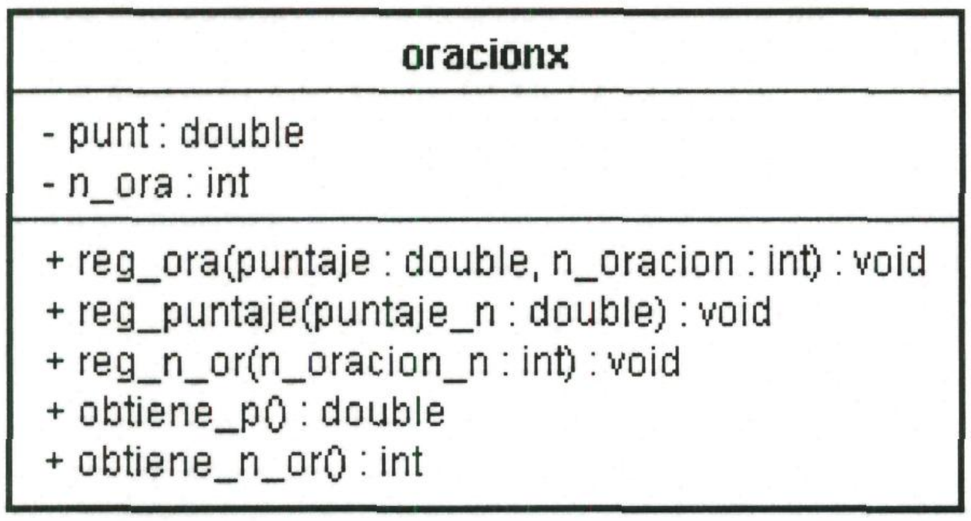
Clase LocationTableModel
La clase LocationTableModel prepara los datos para presentar las ubicaciones en forma de tabla.



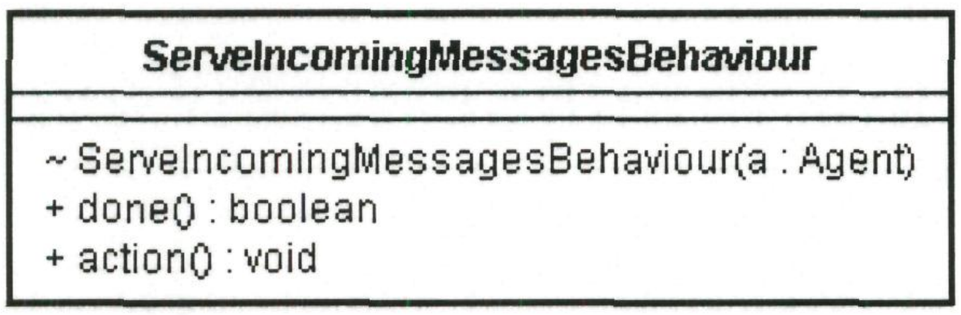
Clase metodopt
La clase metodopt contiene los métodos necesarios para obtener el punto de transición de cada documento.



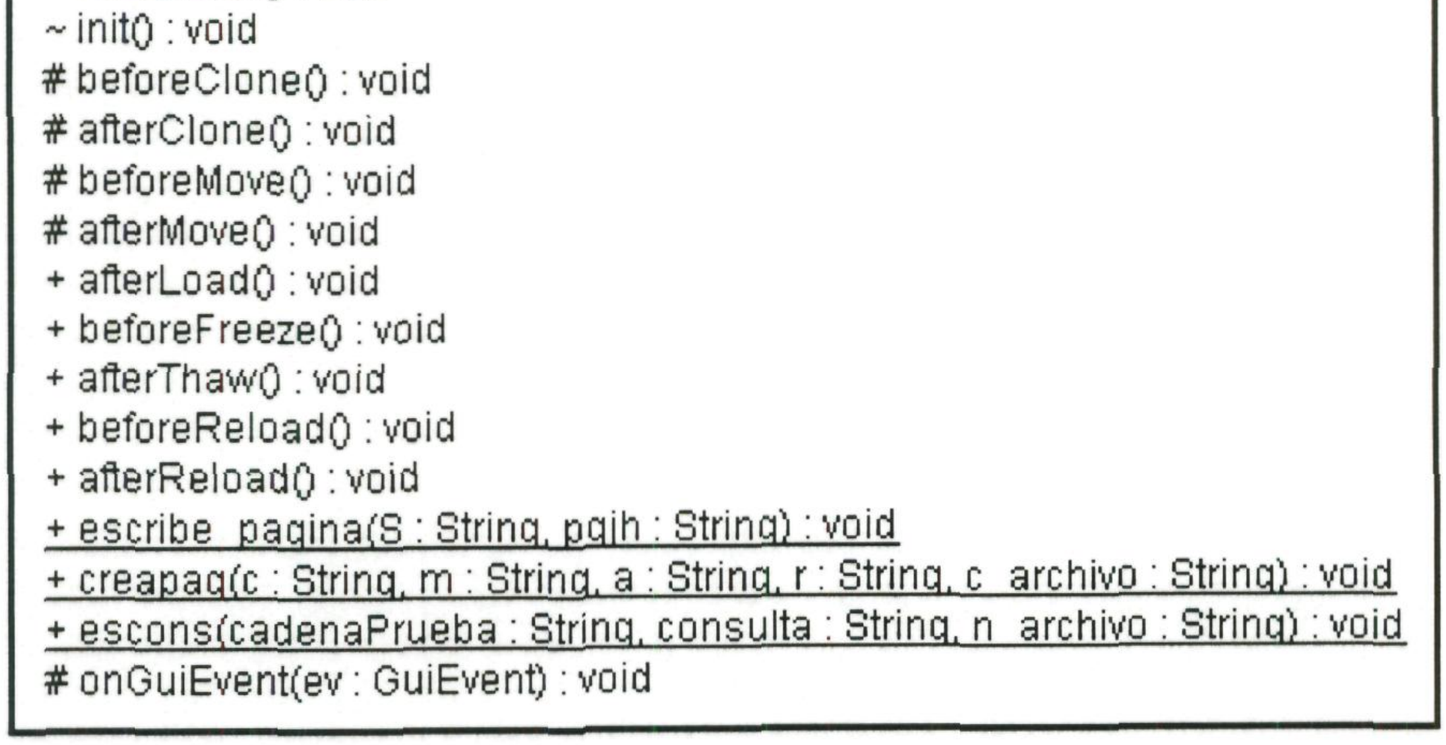
Clase oracionx
La clase oracionx contiene los métodos necesarios para manejar en forma de registro, los datos necesarios en el método del punto de transición.



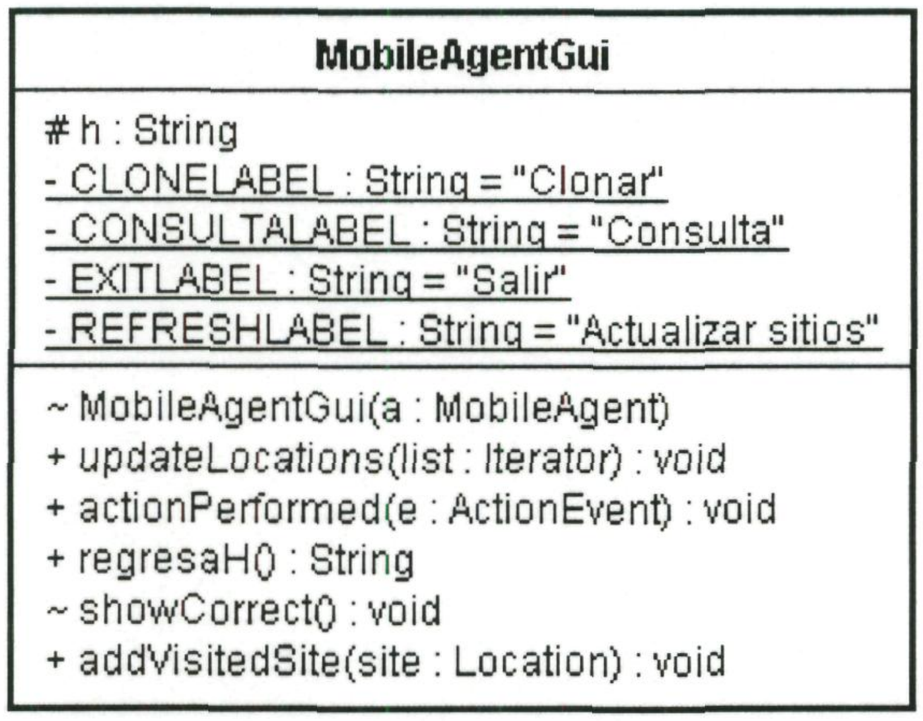
Clase ServeIncomingMessagesBehaviour
La clase ServeIncomingMessagesBehaviour contiene los métodos para manejar los mensajes del agente.



Clase MobileAgent
La clase MobileAgent contiene los métodos necesarios para las operaciones básicas del agente móvil



Clase MobileAgentGui
La clase MobileAgentGui contiene los métodos de la interface del agente móvil.



Diseño de los ficheros de almacenamiento

Para almacenar el historial de las consultas realizadas por un usuario, inicialmente se consideró manejar una base de datos, sin embargo no hay necesidad de ello, pues con la información recopilada sólo basta generar un archivo que contenga los datos importantes del proceso de búsqueda, a continuación se muestra la plantilla del fichero de almacenamiento.

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	
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. Por cada consulta que el usuario realice se crea un fichero que almacena los datos mostrados en la plantilla.

Diagramas de secuencia

Con los diagramas de clases definidos, se pueden generar los diagramas de secuencias. En los diagramas de secuencias se muestra el orden en que suceden las tareas, para llevar a cabo una tarea determinada. A continuación se muestran los diagramas de secuencia para las siguientes operaciones: afterClone y afterMove del método MobileAgent, buscando del método buscar y para la operación metodoptra del método metodopt.

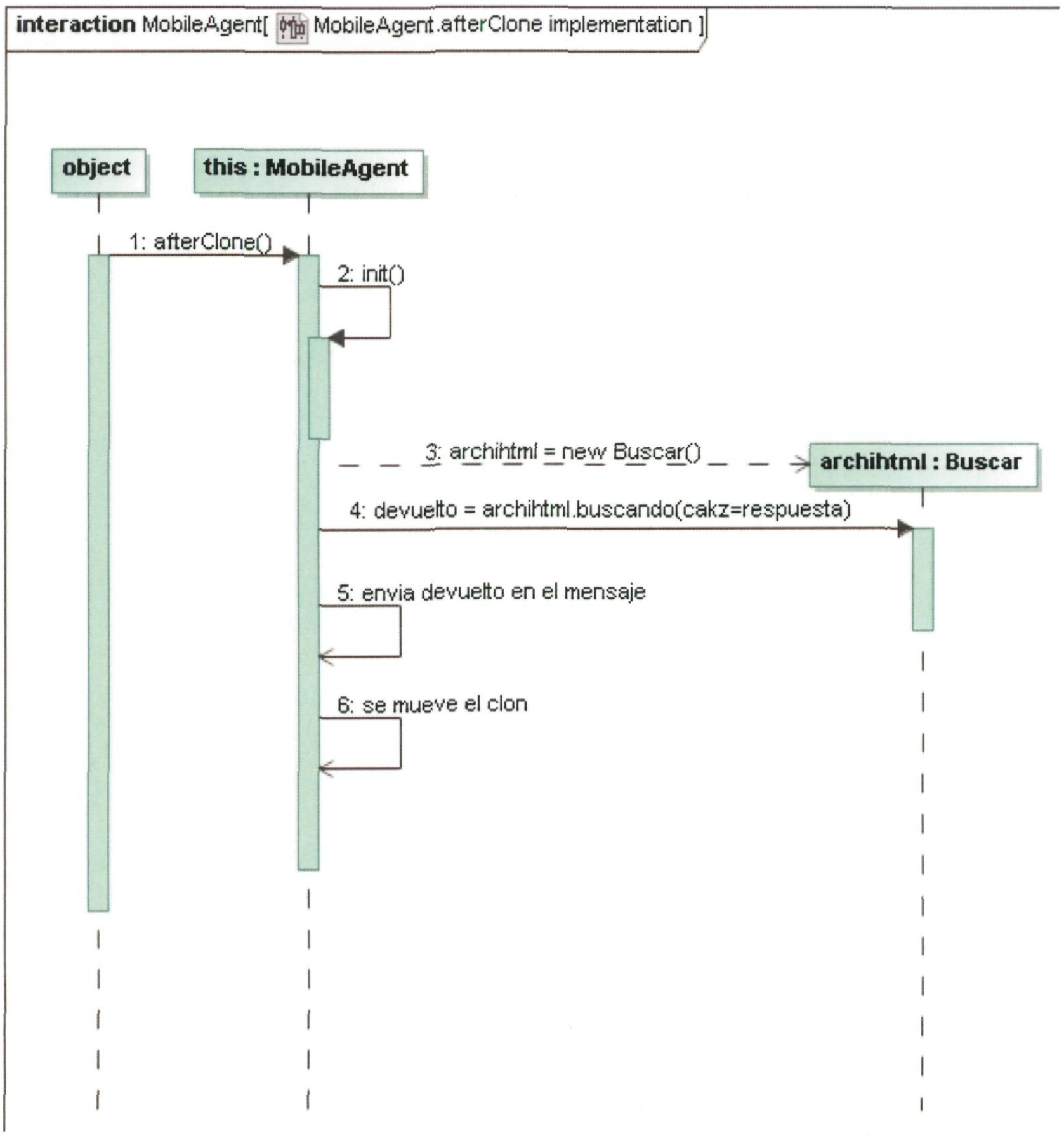


FIGURA 10. DIAGRAMA DE SECUENCIA PARA LA OPERACIÓN AFTERCLONE DEL MÉTODO MOBILEAGENT

FIGURA 11. DIAGRAMA DE SECUENCIA PARA LA OPERACIÓN AFTERMOVE DEL MÉTODO MOBILEAGENT

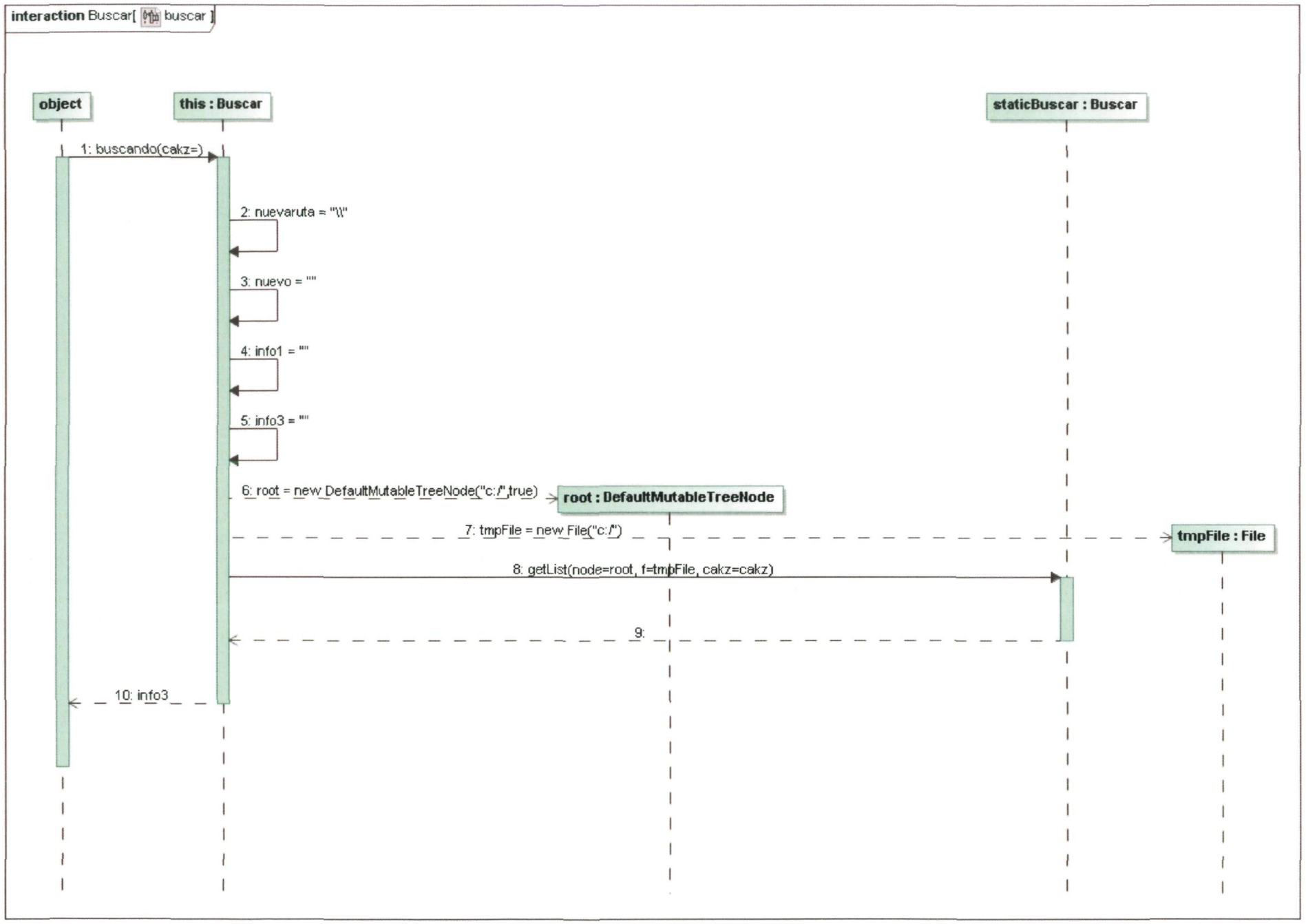


FIGURA 12. DIAGRAMA DE SECUENCIA PARA LA OPERACIÓN BUSCANDO DEL MÉTODO BUSCAR

FIGURA 13. DIAGRAMA DE SECUENCIA PARA LA OPERACIÓN BUSCANDO DEL MÉTODO METODOPTRA

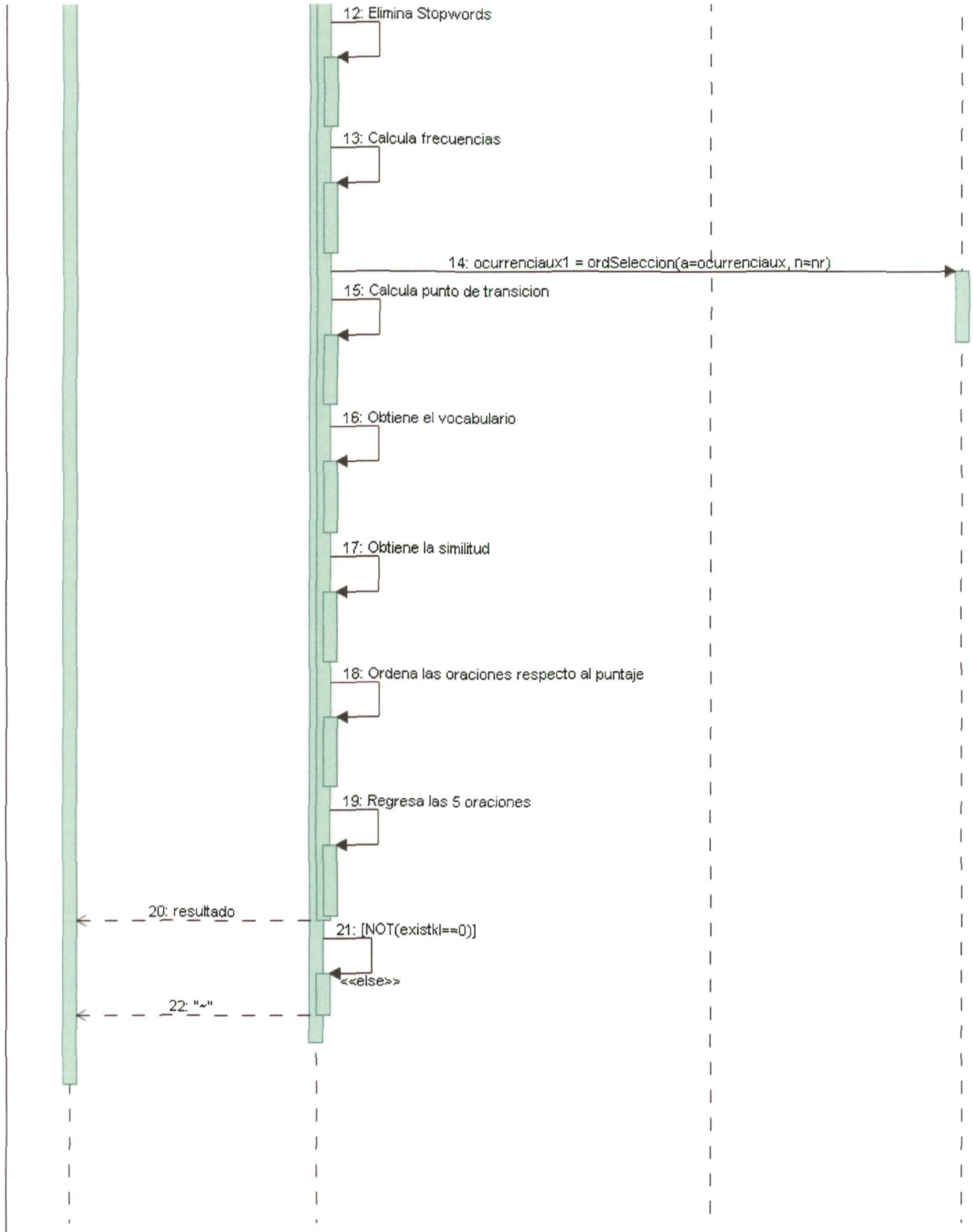


FIGURA 13. DIAGRAMA DE SECUENCIA PARA LA OPERACIÓN BUSCANDO DEL MÉTODO METODOPTRA

Una vez concluidos los diagramas de secuencia, se puede pasar a discutir la implementación realizada.

V. Implementación del Agente

El agente móvil como ya se ha dicho con anterioridad, ha sido desarrollado sobre JADE (Java Agent Development Framework. Esta plataforma proporciona las facilidades para el desarrollo del agente móvil, tales como el soporte donde implementar su modelo de ciclo de vida (creación, clonación y migración) y facilita la comunicación entre los agentes (o entre agentes y un usuario u otra entidad). El agente desarrollado tiene la siguiente característica, viaja a través de los distintos contenedores en forma de clon, buscando en la máquina los archivos para los cuales se cumpla la consulta realizada, el agente clon lleva consigo la consulta que realizará en los hosts por visitar. Considera además la posibilidad de no encontrar ningún documento que cumpla con la consulta especificada. La migración, por tanto es la esencia de los agentes móviles, y en este trabajo el enlace para lograr que un agente viaje de un host a otro se realiza por medio de un identificador.

Para que un agente móvil pueda transitar por una red, es necesario que la plataforma esté apoyada sobre una infraestructura de comunicación determinada, de tal forma que sea dicha infraestructura la que realice la transferencia del agente de una máquina a otra. Las infraestructuras más habituales que utilizan los sistemas de agentes móviles son: TCP/IP o plataformas de más alto nivel como RMI (Remote Method Invocation). JADE utiliza la librería estándar RMI como capa de transporte portable, además trabaja sobre RMI para registrar y encontrar nodos.

Se diseñó e implementó una clase que busca todos los documentos html almacenados en un host, una vez que los localiza, son abiertos para su lectura y se genera el resumen correspondiente de cada documento. Se tiene como nodo raíz un host, donde se albergarán los extractos de cada documento, a su vez este mismo host fungirá como servidor y lanzará diversos clones de los agentes a los distintos contenedores localizados en la red, ver figura 14. Posteriormente cada agente creado revisará las diferentes páginas. Cada vez que un agente clonado termina de revisar la página crea un mensaje con el nombre de la máquina, el archivo del cual se obtuvo el resumen, y el resumen de cada documento. Una vez que todos los objetos clonados han mandado la información obtenida a su nodo padre o servidor. Se crea una pagina HTML con la consulta, la máquina donde se encontró el archivo HTML, el archivo del cual se obtuvo el resumen y el resumen correspondiente.

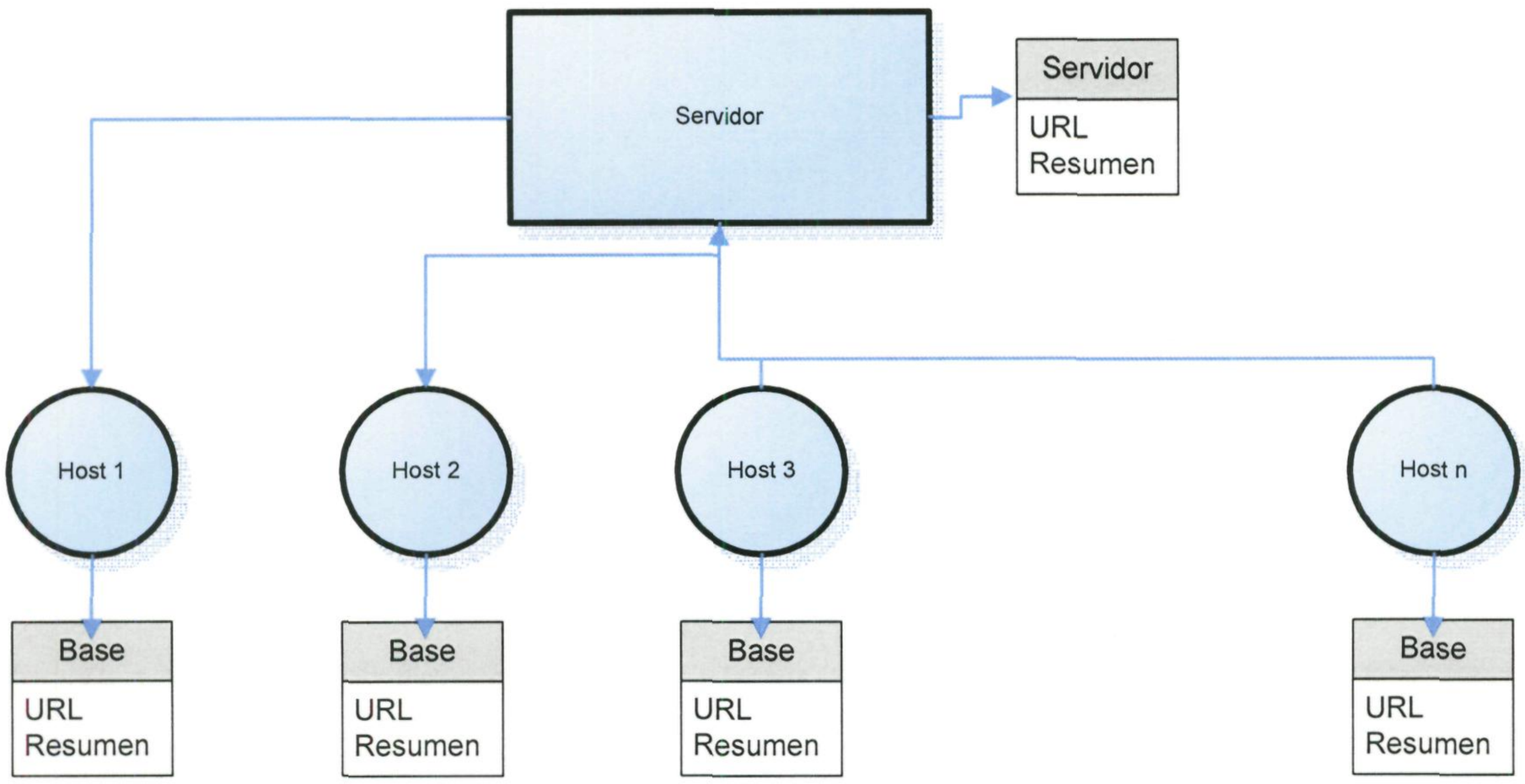


FIGURA 14. ESTRATEGIA DEL AGENTE

Durante la implementación en Jade, en la clase MobileAgent se definió el método afterClone(), y como parte de su implementación se hizo uso de la primitiva doMove().

```
protected void afterClone()
 {
  System.out.println(getLocalName()+" has cloned itself.");
  init();
  String leido, devuelto;
  devuelto="";

  Buscar archihtml = new Buscar();

  devuelto=archihtml.buscando(respuesta);
  String nclo="";
  nclo="<maquina>"+getLocalName()+"</maquina>\n\r";
  devuelto=nclo+devuelto;
  cfpx.setContent(devuelto);
  cfpx.setConversationId("clon");
  send(cfpx);
  ContainerID contene=new ContainerID();
  contene.setName("Main-Container");
  doMove(contene);
}
```

El método setup() de la clase Agent, es heredado dentro de la clase MobileAgent y define por su parte dos comportamientos básicos, el de búsqueda de los diferentes hosts, para mostrar la lista de sitios disponibles y el de lanzamiento de los mensajes, para que el agente pueda lanzar una acción en dependencia del mensaje recibido, como son: salir, mover o clonar entre otros:

```
public void setup()
 {
  getContentManager().registerLanguage(new SLCodec(),
FIPANames.ContentLanguage.FIPA_SL0);

  getContentManager().registerOntology(MobilityOntology.getInstance());

  gui = new MobileAgentGui(this);
  gui.setVisible(true);

  addBehaviour(new GetAvailableLocationsBehaviour(this));
  Behaviour b2 = new ServeIncomingMessagesBehaviour(this);
  addBehaviour(b2);
 }
```

El método action() es mas complejo, pues se involucra el concepto de paso de mensajes, y dependiendo del mensaje recibido se ejecuta alguna acción: salir, mover, clonar, etc.

```
public void action()
{

ACLMessage msg;
MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
msg = myAgent.receive(mt);
if (msg == null) {
block();
return;
}
else {
String replySentence = new String("");
String s = msg.getContent();

StringTokenizer st = new StringTokenizer(msg.getContent(), "
()\t\n\r\f");
String action = (st.nextToken()).toLowerCase();
if      (action.equals("exit"))
{
System.out.println("They requested me to exit (Sob!)");
replySentence = new String("\"OK exiting\"");
myAgent.doDelete();
}
else if (action.equals("move"))
{
String destination = st.nextToken();
System.out.println();
```

```
Location dest = new jade.core.ContainerID(destination, null);
System.out.println("They requested me to go to " + destination);
replySentence = new String("\"OK moving to " + destination+" \"");
((MobileAgent)myAgent).nextSite = dest;
myAgent.doMove(dest);
}
else if (action.equals("clone"))
{
String destination = st.nextToken();
System.out.println();
Location dest = new jade.core.ContainerID(destination, null);
System.out.println("They requested me to clone myself to " +
destination);
replySentence = new String("\"OK cloning to " + destination+" \"");
((MobileAgent)myAgent).nextSite = dest;
myAgent.doClone(dest,
"clone"+((MobileAgent)myAgent).cnt+"of"+myAgent.getName());
}
else if (action.equals("where-are-you"))
{
System.out.println();
Location current = myAgent.here();
System.out.println("Currently I am running on "+current.getName());
replySentence = current.getName();
}

ACLMessage replyMsg = msg.createReply();
replyMsg.setPerformative(ACLMessage.INFORM);
replyMsg.setContent(replySentence);
myAgent.send(replyMsg);
}

return;
}
```

Un agente JADE puede mandar mensajes del tipo jade.lang.acl.ACLMessage. Un mensaje perteneciente a esta clase puede ser transformado en un conjunto de bits que representan la estructura de mensajes FIPA el cual contiene los siguientes campos [31]:

performative: indica el tipo de acto comunicativo del mensaje.
sender: el emisor del mensaje.
receiver: el receptor del mensaje.
reply-to: el receptor de la réplica al mensaje.
content, language, enconding, ontology: el contenido del mensaje, el lenguaje en el que va representado, la codificación y la ontología a la que pertenecen los datos del mismo, respectivamente.
protocol: el protocolo de conversación subyacente entre los agentes implicados.
conversation-id: identificador de la conversación particular a la que pertenece el mensaje.
reply-with: identificador de mensaje que ha de llevar la réplica a este.
in-reply-to: identifica la réplica con respecto al mensaje original.

reply-by: dato de tiempo o fecha, dentro de la cual al agente emisor desearía recibir la réplica.

Inicialmente se crea un mensaje con la intención de hacer una petición simple (query-if) [32] a otro agente, a partir de la cual puede responder que:

(a) no entiende el mensaje (not-understood),
(b) rechaza realizar la petición (refuse),
(c) ha habido un fallo al intentar elaborar la respuesta (failure) ó
(d) la respuesta (inform).

Obviamente, un agente no sólo debe ser capaz de enviar mensajes. También debe poder recibirlos en una forma adecuada. Todo agente tendrá unos cometidos determinados que, junto con el estado en el que estén las posibles conversaciones que mantenga en un momento determinado, van a condicionar los tipos de mensajes que están interesado en recibir en un momento determinado. Para ello, en JADE se dispone de un mecanismo interesante basado en plantillas de mensajes. Básicamente, se definen unas plantillas que especifican un conjunto de mensajes que el agente está interesado en recibir. El siguiente fragmento de código lo ilustra con un ejemplo:

```
MessageTemplate t1 =
MessageTemplate.MatchPerformative(ACLMessage.QUERY_IF);
MessageTemplate t2 = MessageTemplate.and(t1,
MessageTemplate.MatchOntology("PC-ontology"));
MessageTemplate t3 = MessageTemplate.and(t2,
MessageTemplate.MatchLanguage("Jess"));
ACLMessage msg = blockingReceive(t3);
```

Este fragmento, que perfectamente podría aparecer en el cuerpo del método action() de un agente que eventualmente reciba el mensaje creado arriba, va creando plantillas que se van especializando sucesivamente hasta crear la más especifica t3. Ésta incluye a todos los mensajes que tienen como performativa QUERY IF, la ontología correspondiente es \PC-ontology" y el lenguaje que aparece en el contenido del mensaje es Jess. Un detalle interesante es la llamada Agent.blockingReceive(MessageTemplate), ésta causa que el flujo de ejecución del agente se detenga hasta que se reciba un mensaje que cumpla la plantilla pasada como argumento 3. Otro aspecto importante para el desarrollo de una agente es poder establecer mecanismos de coordinación, en particular la coordinación en JADE se sustenta, sobre todo, en dos elementos. El primero consiste en un servicio de directorio de agentes (Agent Directory Services) y un servicio de directorio de servicios (Services Directory Services). El segundo es el conjunto de los protocolos de coordinación que cada agente puede desempeñar.

Como se muestra en el siguiente ejemplo, se coloca el Id de conversación para establecer esta coordinación.

```
String nclo="";
nclo="<maquina>"+getLocalName()+"</maquina>\n\r";
devuelto=nclo+devuelto;
```

```
cfpx.setContent(devuelto);
cfpx.setConversationId("clon");
send(cfpx);
```

En el método action(),el agente recibe el mensaje y ejecuta la acción correspondiente que llega por medio del mensaje y la petición correspondiente:

```
ACLMessage msg;
MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
msg = myAgent.receive(mt);
if (msg == null)
      {
       block();
       return;
      }
else  { …

…
ACLMessage replyMsg = msg.createReply();
replyMsg.setPerformative(ACLMessage.INFORM);
replyMsg.setContent(replySentence);
myAgent.send(replyMsg);
```

V.1. Análisis de Resultados

A continuación se presenta el extracto obtenido para una página html aplicando el siguiente algoritmo:

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

1. Activar desde el contenedor principal a los contenedores remotos.
2. Para cada contenedor activo
    a. Mientras haya páginas html *D* del dominio de aplicación
        i. Eliminar palabras cerradas de la consulta Q
        ii. Buscar palabras de la consulta en el archivo
        iii. Si existe ii se pasa a iv, si no sale de este proceso
        iv. Eliminar Tags
        v. Cambia abreviaturas
        vi. Elimina signos de puntuación
        vii. Eliminar palabras cerradas
        viii. Obtener el vocabulario *V* de *D*
        ix. Dividir el documento en oraciones O_i
        x. Calcular PT
        xi. PV = [PT*.75, PT*1.25]
        xii. $PV = PV \cup Q$

b. Para cada O_i de D obtener
    i. $S_i=sim\ (O_i, PV)$ (Función de similitud de Jaccard)
c. Tomar las 5 oraciones con mayor puntaje de acuerdo a *Si:* $R = \{ Oi \mid Si \}$

3. Construir mensaje: [nombre del archivo, resumen, nombre de clon y nombre de la máquina].
4. Regresar al clon al contenedor principal con la información.
5. Crea fichero de almacenamiento con los resúmenes obtenidos.
6. Destruir la instancia del clon.

Documento sin Procesar

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0078)http://www.lania.mx/biblioteca/newsletters/1999-
otono-invierno/prog_movil.html -->
<HTML><HEAD><TITLE>PROGRAMACIÓN MÓVIL</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2800.1491" name=GENERATOR></HEAD>
<BODY bgColor=lightyellow>
<CENTER></CENTER>
<TABLE cellSpacing=8 cellPadding=6 width=768 align=left border=0>
  <TBODY>
  <TR>
    <TD width="100%">
      <TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
        <TBODY>
        <TR>
          <TD vAlign=baseline width=577><FONT
            face="Arial, Helvetica, sans-serif"><A
            href="http://www.lania.mx/index.html">Inicio</A> /</FONT> <FONT
            face="Arial, Helvetica, sans-serif"><A
            href="http://www.lania.mx/biblioteca/index.html">Biblioteca</A>
            /</FONT> <FONT face="Arial, Helvetica, sans-serif"><A

href="http://www.lania.mx/biblioteca/newsletters/index.html">Newsletters</A>
            / <A
            href="http://www.lania.mx/biblioteca/newsletters/1999-otono-
invierno/index.html">Año
            8, Vol. 29 y 30, Otoño-Invierno 1999</A></FONT></TD>
          <TD width=163>
            <P align=right><FONT face="Arial, Helvetica, sans-serif"><A
            href="http://www.lania.mx/index.html"><IMG height=36
            src="PROGRAMACIÓN MÓVIL_archivos/lania1.gif" width=158
border=0></A>
            </FONT></P></TD></TR>
        <TR>
          <TD vAlign=top colSpan=2>
            <HR align=left noShade>
          </TD></TR></TBODY></TABLE>
      <CENTER>
      <CENTER>
      <CENTER>
```

```
<CENTER>
<CENTER>
<P align=left>
<DIV align=left>
<CENTER>
<P align=left><FONT face="Arial, Helvetica, sans-serif"><BR></FONT>
<H2 align=center><FONT face=Arial>PROGRAMACIÓN MÓVIL</FONT></H2>
<H3 align=center><FONT face=Arial>Un nuevo paradigma para la
implantación
de aplicaciones itinerantes</FONT></H3>
<P align=center><B><FONT face=Arial>Víctor Germán Sánchez
Arias</FONT></B></P>
<P align=justify><FONT face=Arial>La gran difusión en el uso de redes
a
gran escala, como lo es Internet, está planteando al mismo tiempo y de
manera correlacionada, un nuevo dominio de aplicación y un nuevo
paradigma
de programación que, cada vez, se empieza a escuchar más y que tiene
que
ver con la movilidad. Últimamente se mencionan términos como código
móvil,
comunicación "inteligente", agentes móviles, migración de procesos,
aplicaciones itinerantes, etc. Pero no sólo se habla de conceptos,
también
hay ya herramientas disponibles que integran esa característica, como
por
ejemplo, Java. Pero, como a veces sucede en computación, de pronto
todo
mundo habla de un nuevo concepto o una nueva técnica sin saber
exactamente
qué significa.</FONT></P>
```

Continúa…

Documento PreProcesado

1. Se eliminan tags
2. Se eliminan stopwords y signos de puntuación. Se analizan las abreviaturas
3. Se segmenta en oraciones.

```
oracion 1:29 30  otoño-invierno 1999
      programación móvil
       nuevo paradigma implantación
      aplicaciones itinerantes
      víctor germán sánchez
      arias
       gran difusión redes
      gran escala  internet  está planteando
      manera correlacionada  nuevo dominio aplicación nuevo paradigma
      programación  vez  empieza escuchar más
      ver movilidad
oracion 2:últimamente mencionan términos código móvil
      comunicación  inteligente  agentes móviles  migración procesos
      aplicaciones itinerantes  etco sólo habla conceptos  también
```

```
      hay herramientas disponibles integran característica
      ejemplo  java   veces sucede computación  pronto
      mundo habla nuevo concepto o nueva técnica exactamente
      qué significa
       idea básica paradigma
      transmisión datos  código listo ejecutado
      algún servidor red analiza  idea nueva
      encontrar principios redes  iniciaba
      descentralización grandes computadoras época
      popularizaron rje  remote job entry   minicomputadoras
      específico  cuya función enviar lotes programas  código
      ejecutados gran computadora central propósito general
      posteriormente  recibir resultados  caso
      transmitían datos  enviaba código
oracion 3: ejemplo
      popular rsh unix  existe sistema operativo unix
      integró red tcp/ip cpte intérprete  sólo enviar
      ejecutar código estaciones unix conectadas
      red  además permite intercomunicar procesos distantes través
      mecanismo comunicación simple   pipes   envío/ejecución
      código distante  pipes  son mecanismos básicos programación
      procesos distribuidos red sistemas unix
oracion 4:
      ¿así  finalmente transmisión
      código concepto nuevo  podríamos concluir
      llamada programación móvil trata más moda
oracion 5:
      igualmente  sucedido
      paradigmas  concepto básico existía  había generalizado
      había analizado potencial aplicativo
```

Continúa...

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

Palabra	Frecuencia	Palabra	Frecuencia	Palabra	Frecuencia
código	13	movilidad	5	qué	4
red	11	idea	5	unix	4
aplicaciones	7	son	5	capacidad	4
servidor	7	servicios	5	esquema	4
nuevo	6	itinerantes	4	cliente	4
agentes	6	redes	4	así	4
o	6	está	4	ejecución	4
sitios	6	migración	4	lenguaje	4
recursos	6	sólo	4	móvil	3
programación	5	concepto	4	paradigma	3
gran	5	nueva	4	vez	3
más	5	técnica	4	móviles	3
				Frecuencia 1	273

TABLA 15. VOCABULARIO OBTENIDO

Los datos necesarios para obtener el PV se detallan en la tabla 16.

Total de términos en el documento con frecuencia 1	273
PT	11
Límite superior	14
Límite inferior	8
Términos PV	{código, red, inteligencia, artificial}

TABLA 16. EL PÁRRAFO VIRTUAL

Finalmente las oraciones obtenidas con las que se conforma el resumen aplicando la función de Jaccard son:

1. Otro ejemplo, es el popular rsh de UNIX, que existe desde que el sistema operativo UNIX se integró a una red TCP/IP CPte intérprete, no sólo se puede enviar y ejecutar código en cualquiera de las estaciones unix conectadas a una red, sino además permite intercomunicar los procesos distantes a través de un mecanismo de comunicación muy simple, los "pipes" (envío/ejecución de código distante y "pipes" son los mecanismos básicos de programación de procesos distribuidos en una red de sistemas unix).

2. Con este último esquema entramos en el terreno de la inteligencia artificial y a dicho tipo de entidades se les conoce como agentes móviles (entidades autónomas con una gran flexibilidad de movimiento y de cómputo).

3. En particular, uno de los recursos más interesantes que se pueden utilizar es el de procesamiento que, potencialmente, ofrece todos los sitios conectados a la red Poner a la disposición la capacidad de procesamiento de los sitios de una red no es tampoco una idea nueva, sino al contrario, se trata de una técnica muy usual; la arquitectura cliente/servidor es la base para distribuir la utilización de los recursos de una red, incluyendo, desde luego, el de procesamiento.

4. Así, mover y ejecutar código distante son los mecanismos fundamentales de este nuevo tipo de cómputo.

5. ¿Así, si finalmente la transmisión de código no es un concepto nuevo, entonces podríamos concluir que la llamada programación móvil se trata más que nada de una moda?.

Para analizar la calidad de los resúmenes obtenidos se ha analizado el grado de similitud, utilizando la función de Jaccard, entre el resumen brindado por la herramienta Copernic Summarizer [33] y el brindado por el agente, sobre 20 documentos de un CORPUS, conformado por documentos en español en el área de "Agentes". Se obtuvieron los siguientes resultados:

Documentos	**Grado de Similitud**
agentesaplicacionintro.txt	0.335878
agentesmovilescorba.txt	0.37931
agentesmovilesrincon.txt	0.242647
agintri.txt	0.659259
areasaplicaionagentesinteligentes.txt	0.283582
arquitecturaagentes.txt	0.219653
centroinvestigacionweb.txt	0.121569
cooperacionagentesautonomos1.txt	0.217687
cooperacionagentesautonomos2.txt	0.318841
elcomercioelectronicoagentes.txt	0.178423
eldesarrolloagentesinteligentes1.txt	0.302817
eldesarrolloagentesinteligentes2.txt	0.157576
eldesarrolloagentesinteligentes3.txt	0.320988
especificacionesteoriasagentes1.txt	0.214286
especificacionesteoriasagentes2.txt	0.197674

examenplataformas.txt	0.114035
implementacionequiposagentesdinamicos.txt	0.252874
internetintranetagentespractica1.txt	0.213592
internetintranetagentespractica2.txt	0.295337
introduccionconceptoagente.txt	0.0892857
Promedio	0.244451428

TABLA 17. RESULTADOS.

Analizando los resultados de la tabla 17, se aprecia que el grado de similitud fluctúa entre un 11 % y un 65%, resultados alentadores si se considera que el resumen ofertado por Copernico está conformado por las primeras oraciones de cada documento, y por otra parte la técnica que contempla el agente, divide al documento en términos de baja y alta frecuencia, y para la conformación del Párrafo Virtual se toma un 25% de términos alrededor del PT.

VI. Conclusiones y Recomendaciones

El trabajo desarrollado cumple con los objetivos planteados, se ha desarrollado un agente móvil capaz de migrar de host a host y extraer los términos más representativos de un determinado documento, confeccionando un resumen del mismo.

Se probó el comportamiento de la plataforma JADE para la migración de los agentes, resultado importante, pues con anterioridad se había trabajado con ProActive, que mostró serios problemas.

Una línea de investigación muy importante en Computación es la generación de extractos. El hecho de generar un buen extracto de un texto, redundará significativamente en la mejora de técnicas en muchas de las áreas del Procesamiento del Lenguaje Natural. En este trabajo de tesis se ha experimentado con la generación de extractos, mediante el uso del denominado Párrafo Virtual; un conjunto de términos caracterizados como importantes para un texto dado, en general el algoritmo ha brindado bueno resultados.

Por primera vez se probó el comportamiento del algoritmo sobre páginas HTML, con anterioridad se había trabajado con documentos con ciertas características. El formato de las páginas usadas ha sido UTF-8.

Como trabajo futuro se debe tematizar el documento original, con el objetivo de lograr recuperar con mayor facilidad a los términos de una misma familia.

Bibliografía

[1] Mª Dolores Vicente Luque , *Estudios de Información y Documentación de la UOC.* 20/07/2007. http://www.uoc.edu/web/esp/art/uoc/vicente0302/tfc/index1.html

[2] ABCdatos.com, *Definiciones de Robots de Búsqueda*. 2007. http://www.abcdatos.com/buscadores/robot.html

[3] Instituto Tecnológico de Ciudad Madero. *Estudio y Diseño de Agentes Móviles.* 2007

[4] Hipola, P. Y Vargas-Quesada, B. *Agentes inteligentes: definición y tipologías.* Los agentes de información. El profesional de la información, 1999, vol. 8. nº 4, p. 13-21

[5] Berrocal, J.L., et al. *Agentes inteligentes: recuperación autónoma de información en la web.* Revista Española de Documentación Científica, 2003, vol. 26, nº 1

[6] Ramón M. Gómez Labrador, *Agentes móviles y Corba.* Deptº de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Febrero de 1999, http://www.informatica.us.es/~ramon/tesis/CORBA/Seminario-MASIF/

[7] Inglada, V. J. Y Botti, V. *Agentes inteligentes: el siguiente paso en la Inteligencia Artificial.* Novatica, may-jun, 2000. p. 95-99

[8] Web, Universidad Nacional de Colombia. 2005. http://www.virtual.unal.edu.co/cursos/ingenieria/2001394/docs_curso/capitulo1/leccion1.3.htm

[9] Web, Lars Braubach, Alexander Pokahr, University of Hamburg, 2007, http://vsis-www.informatik.uni-hamburg.de/projects/jadex/links.php

[10] Giovanni Caire, *Jade Tutorial, Jade Programming for beginners.* 4 de Diciembre de 2003

[11] Fabio Bellifemine, Giovanni Rimessa, Tiziana Trucco, Giovanni Caire. *Jade programmer's guide.* 2 de Marzo de 2005

[12] Foundation for Intelligent Physical Agents. *FIPA abstract architecture specification. Technical report*. FIPA, February 2002

[13] Carlos Ángel Iglesias Fernández. *Fundamentos de los agentes inteligentes.* Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, 1997.

[14] Salazar-Martínez H Jiménez-Salazar H, Pinto-Avendaño D. *Information Retrieval Based on Text Extraction*. Ist. Indian International Conference on Artificial Intelligence, 2003.

[15] Web. *La cantidad de sitios en Internet sobrepasa los 100 millones.* http://www.linuxparatodos.net/geeklog/article.php?story=20061103093051 83

[16] Lawrence Sergey, Brin. *The anatomy of a large-scale hypertextual web search engine.* Computer Science Departament, 1998.

[17] Climent-Roca Salvador. *Sistemas de resumen automático de documentos.* http://www.uoc.es/humfil/digithum/digithum3/catala/Art_Climent_cat/Clim%ent/climent.html,2001.

[18] Inxight. *Inxight software.* http://www. inxight.com/, 2001.

[19] Hércules Dalianis Martin Hassel. *Resumidor.* http://www.nada.kth.se/~xinartin/swesum/Índex-eng.html, 2001.

[20] Web, *Resumidor*. http://extractor.iit.nrc.ca/on.line.demo.html, 2001.

[21] Universidad de Otawa. *The text summarization project.* http://www.site.uottawa.ca/tanka/ts.html, 2001.

[22] Dragomir Radev. *Text summarization.* http: //wwwl. es. columbia. edu/~radev/, 2001.

[23] Alberto Díaz José María Gómez Ignacio Acero, Mafias Alcojor. *Generación Automática de Resúmenes Personalizados.* Universidad Europea, Departamento de Inteligencia Artificial, 2000.

[24] TIDES. *Translingual information, detection, extraction and summarization (tides).* http://www.darpa.mil/ipto/research/tides/, 2001.

[25] Héctor Jiménez-Salazar Claudia Bueno-Tecpanecatl, David Pinto Avendaño.

[26] David Pinto-Avendaño Héctor Jiménez-Salazar, Hilario Salazar-Martínez. *Text Extraction A Corpus-Based Approach. XXX, aniversario del programa educativo de computación*. BUAP. 2003.

[27] Héctor Jiménez-Salazar. *Notas sobre el enfoque empírico en el procesamiento del lenguaje.* Facultad de Ciencias Físico Matemáticas, BUAP, 2000.

[28] Héctor Jiménez-Salazar Hilario Salazar-Martínez, David Pinto Avendaño. *Comparación de dos métodos que determinan automáticamente el extracto de un texto.* Taller de Tecnologías de lenguaje Humano, 2004.

[29] Rubén Urbizagástegui. *Las posibilidades de la Ley de Zipf en la indización Automática.* Reporte de la Universidad de California Riverside, 1999.

[30] Saltón Gerard. *Advanced information-retrieval models in Automatic Text processing.* 1989.

[31] Foundation for Intelligent Physical Agents. FIPA ACL message structure specification. Technical report. FIPA, August 2001.

[32] Foundation for Intelligent Physical Agents. FIPA query interaction protocol specification. Technical report. FIPA, August 2001.

[33] Web, *Copernic summarizer*, http://www.copernic.com/en/products/summarizer/index.html

Apéndice

Instalación de JADE

Describiremos en esta sección los pasos que hay que seguir para instalar la plataforma *JADE* de forma adecuada. Dicha instalación ha sido probada en *Windows 98*. El único requerimiento software es poseer la versión de Java *JDK 1.2*.

En primer lugar nos descargamos los ficheros `jadeBin.zip` y `jadeExamples.zip`. Ambos ficheros se pueden encontrar en la página de la asignatura. Para descomprimir estos ficheros se recomienda usar `'jar xvf'` en lugar de *Winzip*, ya que los desarrolladores de *JADE* han detectado que se pueden producir algunas incompatibilidades.

A continuación cambiamos la variable `CLASSPATH` para incluir los ficheros *.jar* en el subdirectorio *lib* y el directorio actual. Para *Windows 9x/NT* el comando sería:

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

Linux
Debe descargar el Java, ejecutar el instalador o instalar según su distribución Linux (por defecto se instala en /usr/bin/java).
Luego debe descargar y descomprimir Jade, el agente de conexión, y setear 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:\/ho
me/jade/lib/iiop.jar:/home/jade/lib/Base64.jar:/home/jade/lib/jadeTools.j
ar:\
```

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

```
java jade.Boot -gui
```

Veremos cómo nos aparece una consola (agente RMA) a partir de la cual podemos crear agentes y enviar mensajes entre ellos.

Dentro de la librería *jade/src/examples* podemos encontrar una serie de ejemplos escritos en Java, que son agentes que también se pueden ejecutar desde la línea de comandos. Para ejecutar unos de estos ejemplos nos tenemos que situar en el directorio *jade/src* y referirnos 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
```

Si además de lanzar dicho ejemplo queremos también lanzar el agente *RMA* añadiríamos al comando anterior la opción -gui:

```
java jade.Boot -gui -platform a1: examples. behaviours. ComplexBehaviour
Agent
```

Para una descripción más detallada de los distintos ejemplos de agentes y de cómo estos se arrancan se recomienda ver los ejemplos de JADE.

Documentación disponible

Al descargarnos los ficheros descritos en el apartado de Instalación de JADE, nos encontramos con la siguiente documentación:

- En el directorio *jade\doc* encontramos el fichero *index.html* que es la página principal de la documentación on-line que *JADE* proporciona. Dentro de esta documentación figuran una introducción a *JADE*, una guía del administrador y otra del programador y una documentación completa de la *API* con todas las clases y paquetes que nos podemos encontrar en la plataforma.
- El fichero *JADE Administrator's guide* es un manual con formato *pdf* en el que se detallan los requerimientos y la obtención del software, la ejecución de *JADE* desde la distribución binaria y desde el código fuente y el soporte para el envío y recepción de mensajes entre plataformas con *MTP*. Finaliza con una introducción a la interfaz gráfica de la que el administrador dispone para gestionar y monitorizar la actividad de los agentes.
- El fichero *JADE Programmer's guide* es un manual con formato *pdf* en el que se presenta una introducción a *JADE* y a sus principales características. También se detalla la plataforma que soporta los agentes, la clase *Agent*, los mensajes *ACL*, cómo se introducen tareas y comportamientos en los agentes de *JADE*, los distintos

protocolos existentes para la comunicación entre agentes, cómo se definen nuevos lenguajes y ontologías y el soporte a la movilidad de agentes.

- El fichero *Readme* que se encuentra en el directorio raíz de Jade describe muy por encima los siguientes aspectos: requerimientos del sistema, instalación y prueba, cómo se lanzan los ejemplos desde la línea de comandos, y el soporte *IIOP* para el intercambio de mensajes entre distintas plataformas.
- En *jade\src\examples* encontramos otro fichero *Readme* en el que se describen cada uno de sus subdirectorios con los ejemplos que contienen y para qué sirve.

Ejemplos de sistemas de agentes JADE

La plataforma *JADE* proporciona un conjunto de ejemplos utilizando distintas clases de la plataforma. En este apartado veremos como se ejecutan los distintos ejemplos, su funcionamiento y los distintos errores que generan.

En primer lugar hemos de compilar los ficheros *.java* para convertirlos en ficheros *.class*. Una vez compilados para ejecutar uno de estos ejemplos nos tenemos que situar en el directorio *jade/src* y referirnos 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
```

Si además de lanzar dicho ejemplo queremos también lanzar el agente *RMA* añadiríamos al comando anterior la opción *-gui*:

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

Si ahora lo que queremos es lanzar un ejemplo compuesto por dos programas escritos en Java, como pueden ser las clases *AgentReceiver* y *AgentSender* contenidas en el subdirectorio *'receivers'* del directorio *'examples'*, la instrucción a teclear en la línea de comandos sería (previamente hemos de situarnos en *jade/src*):

```
java jade.Boot -platform a1:examples.receivers.AgentReceiver
a2:examples.receivers.AgentSender
```

Al igual que en el ejemplo anterior, si además de lanzar dicho ejemplo queremos también lanzar el agente *RMA* añadiríamos al comando anterior la opción *-gui*:

```
java jade.Boot -gui -platform a1:examples.receivers.AgentReceiver
a2:examples.receivers.AgentSender
```

Para obtener más detalladamente la descripción de todos los ejemplos que hay en cada uno de los subdirectorios del directorio *src/examples* se recomienda leer el fichero *Readme* que se encuentra en *jade/src/examples*.

Ejemplo página

Inicio / Biblioteca / Newsletters / Año 8, Vol. 29 y 30, Otoño-Invierno 1999

PROGRAMACIÓN MÓVIL
Un nuevo paradigma para la implantación de aplicaciones itinerantes
Víctor Germán Sánchez Arias

La gran difusión en el uso de redes a gran escala, como lo es Internet, está planteando al mismo tiempo y de manera correlacionada, un nuevo dominio de aplicación y un nuevo paradigma de programación que, cada vez, se empieza a escuchar más y que tiene que ver con la movilidad. Últimamente se mencionan términos como código móvil, comunicación "inteligente", agentes móviles, migración de procesos, aplicaciones itinerantes, etc. Pero no sólo se habla de conceptos, también hay ya herramientas disponibles que integran esa característica, como por ejemplo, Java. Pero, como a veces sucede en computación, de pronto todo mundo habla de un nuevo concepto o una nueva técnica sin saber exactamente qué significa.

La idea básica de este paradigma es la transmisión no de datos, sino la de código listo para ser ejecutado en algún servidor de la red. Si se analiza bien, esta idea no es nada nueva, la podemos encontrar desde los principios de las redes, cuando se iniciaba la descentralización de las grandes computadoras. En esa época se popularizaron los RJE "remote job entry", minicomputadoras de uso específico, cuya función era enviar lotes de programas (código) para ser ejecutados en una gran computadora central de propósito general, y posteriormente, recibir los resultados. En este caso no se transmitían datos, se enviaba código. Otro ejemplo, es el popular *rsh* de UNIX, que existe desde que el sistema operativo UNIX se integró a una red TCP/IP. Con este intérprete, no sólo se puede enviar y ejecutar código en cualquiera de las estaciones unix conectadas a una red, sino además permite intercomunicar los procesos distantes a través de un mecanismo de comunicación muy simple, los "pipes" (envío/ejecución de código distante y "pipes" son los mecanismos básicos de programación de procesos distribuidos en una red de sistemas unix).

¿Así, si finalmente la transmisión de código no es un concepto nuevo, entonces podríamos concluir que la llamada programación móvil se trata más que nada de una moda?

Igualmente, como ha sucedido con otros paradigmas, aunque el concepto básico ya existía, no se había generalizado ni tampoco se había analizado todo su potencial aplicativo. Este potencial sólo se hizo patente, primero, cuando el uso de las redes se hizo muy popular, gracias a las interfaces abiertas que permitieron la interconexión de sitios heterogéneos, como es el caso de los "browsers"; y segundo, cuando la interconexión de estaciones se hizo a una gran escala, como es el caso actual de Internet. Con una red de esta dimensión, en principio se cuenta con la posibilidad de utilizar todos los recursos de la red, los cuales son prácticamente ilimitados, aún comparándolos con los que podría ofrecer la mayor supercomputadora del mundo. En particular, uno de los recursos más interesantes que se pueden utilizar es el de procesamiento que, potencialmente, ofrece todos los sitios conectados a la red.

Poner a la disposición la capacidad de procesamiento de los sitios de una red no es tampoco una idea nueva, sino al contrario, se trata de una técnica muy usual; la arquitectura cliente/servidor es la base para distribuir la utilización de los recursos de una red, incluyendo, desde luego, el de procesamiento. Bajo este esquema, el código denominado servicio, está prefijado en una estación que juega un papel específico -el servidor- y que está a la disposición de todos los sitios cliente. Bajo este esquema se puede distribuir el cómputo mediante la instalación de servicios en diferentes servidores de la red. Gracias a esta popular arquitectura, se han desarrollado intensamente los sistemas distribuidos que conocemos en la actualidad; sin embargo al crecer cada vez más las redes, se encontraron límites a este esquema.

Así, se empezó a sentir la necesidad de flexibilizar la ejecución distribuida, basada en servicios prefijados en un servidor. Una primera idea fue la siguiente, en lugar de que los servicios estuvieran fijos siempre en una estación, por qué no moverlos a cualquiera de las estaciones conectadas a la red. Así, por ejemplo, aparecieron los "browsers", códigos que residen en el servidor pero que son ejecutados en los clientes; con esta técnica se libera a los servidores, aligerando su carga y por lo tanto, su tiempo de respuesta. Posteriormente, apareció Java como un lenguaje orientado a objetos y de propósito general, que permite la distribución de código a clientes remotos. Otra idea fue, si es posible mover código del servidor al cliente, por qué no, del cliente al servidor; esta técnica, definida como evaluación remota, dio nacimiento al término de servidores elásticos. Con esta facilidad, el servidor no está sólo restringido a la ejecución de servicios generales prefijados, sino además, puede ejecutar servicios específicos requeridos por un cliente, quien envía el código correspondiente, pero no cuenta con los recursos que tiene el servidor. Así, una vez liberado el sentido del movimiento, por qué no liberar la iniciativa en la migración. De este modo, un programa con capacidad de migrar podría decidir por sí mismo una nueva migración si en el sitio al que llegó no encontró la información o los recursos necesarios para completar su misión. Para que esto sea factible, es necesario que el control de migración tenga una mayor flexibilidad y que esté a la disposición del programador de este tipo de aplicaciones, que algunos denominan

como aplicaciones itinerantes. El mecanismo de decisión de migrar puede ser prefijado en un lenguaje imperativo como C o bien podría flexibilizarse aún con técnicas de programación declarativa como la orientada a objetos o incluso, aún más, dando al lenguaje capacidades de procesar y almacenar conocimiento tal, que al viajar, le permita modificar, tanto su propio conocimiento como el del sitio visitado (si éste contara también con esa misma capacidad).

Con este último esquema entramos en el terreno de la inteligencia artificial y a dicho tipo de entidades se les conoce como agentes móviles (entidades autónomas con una gran flexibilidad de movimiento y de cómputo). Así, mover y ejecutar código distante son los mecanismos fundamentales de este nuevo tipo de cómputo. El concepto de movilidad plantea, a nivel de la implantación, problemas tales como: lograr una ejecución homogénea en procesadores heterogéneos; identificar a agentes viajantes desconocidos con los que se desea comunicar; proteger a los sitios receptores de agentes "mal intencionados"; proteger a los agentes viajantes de sitios "mal intencionados"; generalizar y controlar las funciones básicas para integrarlas en la semántica de un lenguaje orientado a la movilidad; y definir modelos que nos faciliten la especificación y diseño de aplicaciones itinerantes. Todos estos problemas, son temas de desarrollo e investigación que actualmente se están realizando en diversos centros y laboratorios de investigación. Sin embargo, aunque no se ha definido la movilidad, ya hay lenguajes que tienen ciertas características de movilidad, tales como Java y Telescript.

Con estos nuevos lenguajes se pueden implantar aplicaciones que requieran de enviar instructivos completos (programas) en lugar de un intercambio de datos o llamadas a ejecución distante, o el envío de expertos viajantes (agentes móviles) con capacidad para migrar, si requiere de otros recursos no existentes en el sitio visitado; nuevas aplicaciones que antes no se planteaban al pensar que el código era inamovible.

Partiendo de este análisis podemos afirmar que la programación sí es un nuevo paradigma, porque, aunque la movilidad de código ya existía, ésta no era explícita ni estaba generalizada y, por lo mismo, no se había previsto el potencial de su aplicación.

Código fuente de la página

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from
url=(0078)http://www.lania.mx/biblioteca/newsletters/1999-otono-
invierno/prog_movil.html -->
<HTML><HEAD><TITLE>PROGRAMACIÓN MÓVIL</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2800.1491" name=GENERATOR></HEAD>
<BODY bgColor=lightyellow>
<CENTER></CENTER>
<TABLE cellSpacing=8 cellPadding=6 width=768 align=left border=0>
  <TBODY>
  <TR>
    <TD width="100%">
      <TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
        <TBODY>
        <TR>
          <TD vAlign=baseline width=577><FONT
            face="Arial, Helvetica, sans-serif"><A
            href="http://www.lania.mx/index.html">Inicio</A> /</FONT>
<FONT
            face="Arial, Helvetica, sans-serif"><A
href="http://www.lania.mx/biblioteca/index.html">Biblioteca</A>
            /</FONT> <FONT face="Arial, Helvetica, sans-serif"><A

href="http://www.lania.mx/biblioteca/newsletters/index.html">Newsletters<
/A>
            / <A
            href="http://www.lania.mx/biblioteca/newsletters/1999-otono-
invierno/index.html">Año
            8, Vol. 29 y 30, Otoño-Invierno 1999</A></FONT></TD>
          <TD width=163>
            <P align=right><FONT face="Arial, Helvetica, sans-serif"><A
            href="http://www.lania.mx/index.html"><IMG height=36
            src="PROGRAMACIÓN MÓVIL_archivos/lania1.gif" width=158
border=0></A>
            </FONT></P></TD></TR>
        <TR>
          <TD vAlign=top colSpan=2>
            <HR align=left noShade>
      </TD></TR></TBODY></TABLE>
<CENTER>
<CENTER>
<CENTER>
<CENTER>
<CENTER>
<P align=left>
<DIV align=left>
<CENTER>
<P align=left><FONT face="Arial, Helvetica, sans-serif"><BR></FONT>
```

```
      <H2 align=center><FONT face=Arial>PROGRAMACIÓN MÓVIL</FONT></H2>
      <H3 align=center><FONT face=Arial>Un nuevo paradigma para la
implantación
      de aplicaciones itinerantes</FONT></H3>
      <P align=center><B><FONT face=Arial>Víctor Germán Sánchez
      Arias</FONT></B></P>
      <P align=justify><FONT face=Arial>La gran difusión en el uso de
redes a
      gran escala, como lo es Internet, está planteando al mismo tiempo y
de
      manera correlacionada, un nuevo dominio de aplicación y un nuevo
paradigma
      de programación que, cada vez, se empieza a escuchar más y que
tiene que
      ver con la movilidad. Últimamente se mencionan términos como código
móvil,
      comunicación "inteligente", agentes móviles, migración de procesos,
      aplicaciones itinerantes, etc. Pero no sólo se habla de conceptos,
también
      hay ya herramientas disponibles que integran esa característica,
como por
      ejemplo, Java. Pero, como a veces sucede en computación, de pronto
todo
      mundo habla de un nuevo concepto o una nueva técnica sin saber
exactamente
      qué significa.</FONT></P>
      <P align=justify><FONT face=Arial>La idea básica de este paradigma
es la
      transmisión no de datos, sino la de código listo para ser ejecutado
en
      algún servidor de la red. Si se analiza bien, esta idea no es nada
nueva,
      la podemos encontrar desde los principios de las redes, cuando se
iniciaba
      la descentralización de las grandes computadoras. En esa época se
      popularizaron los RJE "remote job entry", minicomputadoras de uso
      específico, cuya función era enviar lotes de programas (código)
para ser
      ejecutados en una gran computadora central de propósito general, y
      posteriormente, recibir los resultados. En este caso no se<B>
      </B>transmitían datos, se<B> </B>enviaba código. Otro ejemplo, es
el
      popular<I> rsh </I>de UNIX, que existe desde que el sistema
operativo UNIX
      se integró a una red TCP/IP. Con este intérprete, no sólo se puede
enviar
      y ejecutar código en cualquiera de las estaciones unix conectadas a
una
      red, sino además permite intercomunicar los procesos distantes a
través de
      un mecanismo de comunicación muy simple, los "pipes"
(envío/ejecución de
      código distante y "pipes" son los mecanismos básicos de
programación de
      procesos distribuidos en una red de sistemas unix).</FONT></P>
```

<P align=justify><FONT face=Arial>¿Así, si finalmente la transmisión de
código no es un concepto nuevo, entonces podríamos<B> </B>concluir que la
llamada programación móvil se trata más que nada de una moda?.</FONT></P>
<P align=justify><FONT face=Arial>Igualmente, como ha sucedido con otros
paradigmas, aunque el concepto básico ya existía, no se había generalizado
ni tampoco se había analizado todo su potencial aplicativo. Este potencial
sólo se hizo patente, primero, cuando el uso de las redes se hizo muy
popular, gracias a las interfaces abiertas que permitieron la
interconexión de sitios heterogéneos, como es el caso de los "browsers"; y
segundo, cuando la interconexión de estaciones se hizo a una gran escala,
como es el caso actual de Internet. Con una red de esta dimensión, en
principio se cuenta con la posibilidad de utilizar todos los recursos de
la red, los cuales son prácticamente ilimitados, aún comparándolos con los
que podría ofrecer la mayor supercomputadora del mundo. En particular, uno
de los recursos más interesantes que se pueden utilizar es el de
procesamiento que, potencialmente, ofrece todos los sitios conectados a la
red.</FONT></P>
<P align=justify><FONT face=Arial>Poner a la disposición la capacidad de
procesamiento de los sitios de una red no es tampoco una idea nueva, sino
al contrario, se trata de una técnica muy usual; la arquitectura
cliente/servidor es la base para distribuir la utilización de los recursos
de una red, incluyendo, desde luego, el de procesamiento. Bajo este
esquema, el código denominado servicio, está prefijado en una estación que
juega un papel específico -el servidor- y que está a la disposición de
todos los sitios cliente. Bajo este esquema se puede distribuir el cómputo
mediante la instalación de servicios en diferentes servidores de la red.
Gracias a esta popular arquitectura, se han desarrollado intensamente los
sistemas distribuidos que conocemos en la actualidad; sin embargo al
crecer cada vez más las redes, se encontraron límites a este
esquema.</FONT></P>
<P align=justify><FONT face=Arial>Así, se empezó a sentir la necesidad de

flexibilizar la ejecución distribuida, basada en servicios prefijados en un servidor. Una primera idea fue la siguiente, en lugar de que los servicios estuvieran fijos siempre en una estación, por qué no moverlos a cualquiera de las estaciones conectadas a la red. Así, por ejemplo, aparecieron los "browsers" , códigos que residen en el servidor pero que son ejecutados en los clientes; con esta técnica se libera a los servidores, aligerando su carga y por lo tanto, su tiempo de respuesta.

Posteriormente, apareció Java como un lenguaje orientado a objetos y de propósito general, que permite la distribución de código a clientes remotos. Otra idea fue, si es posible mover código del servidor al cliente, por qué no, del cliente al servidor; esta técnica, definida como evaluación remota, dio nacimiento al término de servidores elásticos. Con esta facilidad, el servidor no está sólo restringido a la ejecución de servicios generales prefijados, sino además, puede ejecutar servicios específicos requeridos por un cliente, quien envía el código correspondiente, pero no cuenta con los recursos que tiene el servidor.

Así, una vez liberado el sentido del movimiento, por qué no liberar la iniciativa en la migración. De este modo, un programa con capacidad de migrar podría decidir por sí mismo una nueva migración si en el sitio al que llegó no encontró la información o los recursos necesarios para completar su misión. Para que esto sea factible, es necesario que el control de migración tenga una mayor flexibilidad y que esté a la disposición del programador de este tipo de aplicaciones, que algunos denominan como aplicaciones itinerantes. El mecanismo de decisión de migrar puede ser prefijado en un lenguaje imperativo como C o bien podría flexibilizarse aún con técnicas de programación declarativa como la orientada a objetos o incluso, aún más, dando al lenguaje capacidades de procesar y almacenar conocimiento tal, que al viajar, le permita modificar, tanto su propio conocimiento como el del sitio visitado (si éste contara también con esa misma capacidad). </FONT></P>

<P align=justify><FONT face=Arial>Con este último esquema entramos en el terreno de la inteligencia artificial y a dicho tipo de entidades se les conoce como agentes móviles (entidades autónomas con una gran flexibilidad

de movimiento y de cómputo). Así, mover y ejecutar código distante son los
mecanismos fundamentales de este nuevo tipo de cómputo. El concepto de
movilidad plantea, a nivel de la implantación, problemas tales como:
lograr una ejecución homogénea en procesadores heterogéneos; identificar a
agentes viajantes desconocidos con los que se desea comunicar; proteger a
los sitios receptores de agentes "mal intencionados"; proteger a los
agentes viajantes de sitios "mal intencionados"; generalizar y controlar
las funciones básicas para integrarlas en la semántica de un lenguaje
orientado a la movilidad; y definir modelos que nos faciliten la
especificación y diseño de aplicaciones itinerantes. Todos estos
problemas, son temas de desarrollo e investigación que actualmente se
están realizando en diversos centros y laboratorios de investigación. Sin
embargo, aunque no se ha definido la movilidad, ya hay lenguajes que
tienen ciertas características de movilidad, tales como Java y
Telescript.</FONT></P>
<P align=justify><FONT face=Arial>Con estos nuevos lenguajes se pueden
implantar aplicaciones que requieran de enviar instructivos completos
(programas) en lugar de un intercambio de datos o llamadas a ejecución
distante, o el envío de expertos viajantes (agentes móviles) con capacidad
para migrar, si requiere de otros recursos no existentes en el sitio
visitado; nuevas aplicaciones que antes no se planteaban al pensar que el
código era inamovible.</FONT></P>
<P align=justify><FONT face=Arial>Partiendo de este análisis podemos
afirmar que la programación sí es un nuevo paradigma, porque, aunque la
movilidad de código ya existía, ésta no era explícita ni estaba
generalizada y, por lo mismo, no se había previsto el potencial de su
aplicación.</FONT></P>
<P align=left><FONT
face="Arial, Helvetica, sans-serif">
</FONT></CENTER>
<CENTER>
<CENTER>
<P align=left></CENTER></CENTER>
<P>
<CENTER></CENTER></DIV></CENTER></CENTER>
<CENTER></CENTER>

```
<P></CENTER></CENTER></CENTER>
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
  <TBODY>
  <TR>
    <TD vAlign=top colSpan=2>
      <P>
      <CENTER>
      <CENTER></CENTER></CENTER>
      <CENTER></CENTER>
            <P>
            <CENTER>
            <DIV align=left></DIV></CENTER>
            <HR>

            <P></P>
            <P align=center><FONT face="Arial, Helvetica, sans-
serif"><B><FONT
            color=#000099 size=1><A href="mailto:webmaster@lania.mx"><IMG
            height=21 src="PROGRAMACIÓN MÓVIL_archivos/arroba1.gif"
width=22
            border=0><BR>comentarios y sugerencias</A></FONT></B></FONT>
        </P></TD></TR></TBODY></TABLE></P></TD></TR></TBODY></TABLE>
<P>
<H1> </H1></BODY></HTML>
```

Glosario

Termino	Significado
ACL	Es un lenguaje de comunicación de agentes que permite la interoperación entre agentes autónomos distribuidos
Agente	Es un componente de software que es autónomo, proactivo y social
Agente débil	Sistema de hardware o un sistema de cómputo basado en software que contiene las propiedades de autonomía, habilidad social, reactividad y proactividad
Agente Dummy	Permite componer y mandar mensajes ACL así como mantener una lista con todos los mensajes ACL mandados y recibidos
Agente fuerte	Sistema computacional que, además de las propiedades mencionadas en el agente débil, es conceptualizado e implementado usando conceptos que son usualmente aplicados a humanos
Agente RMA	Permite controlar el ciclo de vida de todos los agentes residentes en la plataforma
Agente Sniffer	Permite seleccionar cualquier agente o grupo de agentes de la plataforma, de manera que controla el intercambio de mensajes entre dichos agentes
Agentes móviles	Son programas de software capaces de viajar por redes de computadoras, como Internet, de interactuar con hosts, pedir información a nombre de un usuario y regresar a su lugar de origen una vez que ha realizado las tareas especificadas por un usuario
AMS	Proporciona el servicio de denominación y representa la autoridad en la plataforma
Comportamiento	Es una abstracción usada para modelar el flujo de ejecución de un fragmento de código

Comportamientos en paralelo	Se refieren a un código que se ejecuta en paralelo
Comportamientos simples	Se refieren a un código que se ejecuta secuencialmente
Contenedor	Instancia de ejecución en JADE
Coordinación	Propiedad de un agente para no interferir en las actividades de otro
DF	Proporciona el servicio de las páginas amarillas que es el medio por el que un agente puede encontrar a otros agentes que proporcionan los servicios que él exige para lograr sus metas
FIPA	Es un estándar para los aspectos de relacionados con la tecnología de agentes y sistemas multi-agentes, esta enfocado al soporte de la interoperabilidad de agentes desarrollados sobre marcos de trabajo en los cuales se soporten transportes heterogéneos.
Interacciones básicas	Se refiere a la asimilación por parte del agente receptor de la interacción, de un efecto racional deseado por el agente que inicia la interacción
Interoperabilidad	Condición necesaria para que los usuarios tengan un acceso completo a la información disponible
JADE	Plataforma diseñada para el desarrollo de agentes
JAVA	Tecnología desarrollada por Sun Microsystems para aplicaciones software independiente de la plataforma
Main container	Es un contenedor único principal

Mensajes asíncronos	Es un modelo de comunicación entre entidades heterogéneas que no saben nada de los otros
Movilidad de código y de estado de ejecución	Es el proceso por el cual un agente puede dejar de correr sobre un Computador, emigrando sobre diferentes computadores remotos (sin la necesidad de tener el código del agente instalado en ese computador), y reiniciar su ejecución del punto que fue interrumpido
Negociación	Mecanismos de convencimiento para adquirir un bien
Páginas amarillas	Es un servicio que permite a los agentes publicar uno o más servicios que ellos proporcionan para que otros agentes los puedan encontrar y consecutivamente pueden aprovecharlos
Plataforma	Conjunto de todos los contenedores
Proactividad	Es el proceso por el cual los agentes pueden programarse para comenzar la ejecución de acciones sin la intervención humana, solo en base a un objetivo y cambios de estado
Protocolo	Es un patrón que se usa para llevar por unos cauces concretos una conversación
Protocolos de negociación	Esta forma de interacción entre agentes
Sistema abierto	Son aquellos sistemas que proporcionan alguna combinación de interoperabilidad, portabilidad y uso de estándares abiertos
Versatilidad	Es una característica que permite a desarrolladores de aplicaciones reusar el mismo código de la aplicación para un PC, un PDA o un teléfono con Java
XML	Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos.