HERRAMIENTA INFORMÁTICA BASADA EN ONTOLOGÍAS PARA LA CLASIFICACIÓN AUTOMÁTICA DE INFORMACIÓN DE SEGURIDAD

HECTOR FERNANDO VARGAS MONTOYA
Código 200128302010

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
ESCUELA DE INGENIERÍA
UNIVERSIDAD EAFIT
MEDELLÍN
2.005

HERRAMIENTA INFORMÁTICA BASADA EN ONTOLOGÍAS PARA LA CLASIFICACIÓN AUTOMÁTICA DE INFORMACIÓN DE SEGURIDAD

HECTOR FERNANDO VARGAS MONTOYA

Proyecto de grado para optar al título de
Ingeniero de Sistemas

Asesor

Juan Guillermo Lalinde Pulido
Doctor Ingeniero en Telecomunicaciones
Jefe del departamento de Ingeniería de Sistemas

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
ESCUELA DE INGENIERÍA
UNIVERSIDAD EAFIT
MEDELLÍN
2.005

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Medellín, Noviembre de 2005

A mi familia: Mi esposa y mis hijos por su infinita paciencia durante el transcurso de la carrera y de este proyecto, brindándome su apoyo, tiempo y amor.

AGRADECIMIENTOS

El autor expresa sus agradecimientos a:

Juan Guillermo Lalinde Pulido, Doctor Ingeniero de Sistemas, por brindarme la oportunidad de participar en este proyecto, por el conocimiento y experiencia para el desarrollo del mismo.

Y a todas aquellas personas que de una u otra forma me dieron información que apoyaron la realización del proyecto.

TABLA DE CONTENIDO

Lista de Figuras

Pág.

1. INTRODUCCIÓN

La tecnología y la computación avanzan a pasos agigantados. Paralelamente los ataques en seguridad informática, exitosos o no, a organizaciones se hacen cada vez más constantes [[1]]. Los programas adquiridos para reducir el riesgo de ataques exitosos a las organizaciones deben ser actualizados rápidamente al momento que se detecte una nueva vulnerabilidad. Pero, el conocer esta nueva vulnerabilidad, en ocasiones es demasiado tardío y depende de la recolección a tiempo y procesamiento de la información de los canales que divulgan los nuevos eventos en seguridad.

Por otra parte, las nuevas tecnologías ofrecen la posibilidad de crear almacenes muy grandes de información. Sin embargo disponer de cantidades abundantes de datos sin el soporte tecnológico adecuado puede generar desinformación. Este problema es especialmente significativo cuando el buen funcionamiento de la organización depende de la calidad de la información.

La necesidad de tener instalados dispositivos de seguridad tanto internos como externos, crece a medida que las organizaciones se hacen más grandes o se vuelven más vulnerables en el ámbito informático. Para fomentar más la disponibilidad de la información, es necesario tener un alto desempeño en los equipos y programas capaces de asegurar y reducir el riesgo de perdida de información.

Uno de los problemas en el campo de la seguridad informática es cómo manejar los grandes volúmenes de información que se generan permanentemente. Para tener un repositorio de información sobre vulnerabilidades que sea útil a la organización se debe contar con una herramienta que sea capaz de recopilar, analizar y clasificar automáticamente la información, transformándola en

conocimiento interrelacionado y eficaz. Adicionalmente, en el campo de la seguridad informática, se requiere un tiempo de respuesta adecuado a las diferentes notificaciones, dado que de esto depende la exposición de los sistemas informáticos.

Este trabajo de grado, titulado *HERRAMIENTA INFORMÁTICA BASADA EN ONTOLOGÍAS PARA LA CLASIFICACIÓN AUTOMATICA DE INFORMACIÓN DE SEGURIDAD*, es una prueba de concepto que es posible construir una herramienta para la clasificación automática de la información relacionada con vulnerabilidades de seguridad en herramientas software.

En la primera parte, se da a conocer el problema y las necesidades específicas a las que se desean dar solución con este proyecto.

En la segunda parte, se tiene el marco de referencia, los conceptos teóricos y base conceptual, necesarios para la consecución de la solución. Se referencia temas relacionados directa o indirectamente con el proyecto, tecnologías y software involucrado y como conceptualización principal, se tiene la investigación sobre Ontologías y algo sobre recuperación de información.

Como parte final y teniendo el marco de referencia anterior, se propone una solución al problema, donde se describe todo el desarrollo elaborado para tal fin.

Como insumo adicional (y parte importante para el usuario final y futuros desarrollos), se tiene algunas consideraciones de inconvenientes presentados en el transcurso del proyecto.

2. GLOSARIO

Ontología [15]: Define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento.

Conocimiento interrelacionado: Unificación, colaboración o relación directa entre conocimientos dentro de un dominio específico.

Ataques: Acción hostil efectuado por medios físicos o lógicos hacia una infraestructura informática, utilizando la fuerza o el conocimiento en programas software.

Vulnerabilidad: Es una "debilidad" de algún sistema, que es la posibilidad de un ataque (cualquiera que sea la probabilidad).

Repositorio único: Es un contenedor donde residen o puede residir información diversa pero centralizada, que puede ser utilizada por diferentes sistemas o personas, teniendo un punto único de control.

Clasificación automática de información: Proceso por el cuál se hace una clasificación de información sin la intervención del hombre.

Herramientas para la construcción de Ontologías: Piezas de software capaces de diseñar, modelar, construir o actualizar ontologías.

Atacante: Persona o programa malicioso que ejecuta un ataque a uno o varios puntos de la infraestructura informática.

Taxonomía: En su sentido más general, la taxonomía (del griego taxis, "ordenamiento", y nomos, "norma" o "regla") es la ciencia de la clasificación.

Metalenguajes: Medio para describir formalmente un lenguaje, para el caso de sistemas de información, un lenguaje de codificación etiquetado.

Web semántica: Es una extensión de la actual Web que permitirá encontrar, compartir y combinar la información más fácilmente. Es una iniciativa para permitir que la información que reside en la red sea accesible y "comprensible" no sólo por los humanos sino también por las máquinas

IDE: Integrated development environment – es un entorno integrado de desarrollo de software, ej: Eclipse, Java Beans, J-Builder, etc..

***Open Source* :** Cualidad de algunos softwares de incluir el código fuente en la distribución del programa, permitiendo la posibilidad de modificarlo y redistribuirlo..

Ofimática: Nombre que se suele dar a la informática aplicada a la oficina, herramientas colaborativas tales como: proceso de textos, hojas de cálculo, etc.

OWL (*Web Ontology Language*): Proporciona un lenguaje para la definición de Ontologías estructuradas, basadas en la Web, que ofrece una integración e interoperabilidad de datos más rica entre comunidades descriptivas.

Framework [2]**:** Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Un FrameWork representa una Arquitectura de Software que modela las relaciones generales de las entidades del dominio.

EMF: Es un framework de modelado y generación de código, para construir herramientas y otras aplicaciones basadas en un modelo de datos estructurado.

***Buffer*:** Un buffer es una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

RDF [30]**:** Es un modelo de meta-datos para los recursos y las relaciones que se puedan establecer entre ellos, que ha sido desarrollada por W3C. Aporta una semántica básica para este modelo de datos que puede representarse mediante XML.

W3C [33]: El W3C (*World Wide Web Consortium*) fue creado para guiar la Web hacia su potencial máximo mediante el desarrollo de protocolos comunes que promuevan su evolución y garanticen su interactividad.

XML: Su objetivo principal era conseguir una página Web más semántica, separando la estructura del contenido y permitiendo el desarrollo de vocabularios modulares, compatibles con cierta unidad y simplicidad del lenguaje.

Motores de Inferencia: Es un programa ubicado entre el usuario y una base de conocimiento, y su objetivo es extraer conclusiones a partir de los datos simbólicos que están almacenados en las bases de hechos y de conocimiento.

Java: Es una plataforma de software desarrollada por Sun Microsystems [3], de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

Plugins: Es un programa de computador que interactúa con otro programa para aportarle una función o utilidad específica, Este programa adicional es ejecutado por la aplicación principal.

3. DEFINICIÓN DEL PROBLEMA

Las nuevas tecnologías ofrecen la posibilidad de crear almacenes muy grandes de información, sin embargo, disponer de cantidades ingentes de datos puede provocar en ocasiones, una preocupante desinformación y una diferencia grande entre la tecnología emergente y la implantada, en particular, en América latina [4].

La tecnología y la computación avanzan a pasos agigantados. Paralelamente los ataques, exitosos o no, a organizaciones se hacen cada vez más constantes [5]. La necesidad de tener instalados dispositivos de seguridad tanto internos como externos, crece a medida que las organizaciones se hacen más grandes o se vuelven más vulnerables en el ámbito informático, para garantizar la disponibilidad de la información, es necesario tener un alto desempeño en los equipos y programas capaces de asegurar y reducir el riesgo de perdida de información.

Los programas adquiridos para reducir el riesgo de ataques exitosos a las organizaciones, deben ser actualizados rápidamente al momento que se genere una nueva vulnerabilidad, pero, el conocer esta nueva vulnerabilidad, en ocasiones es demasiado tardío y depende de los canales, controlados por organizaciones confiables, por los cuales se divulgan estos nuevos eventos en seguridad (tal es el caso de organizaciones como CERT, grupos de seguridad, empresas productoras de software o hardware y las casas de antivirus) [6].

Para obtener esta información (de manera confiable y lo más exacto posible), es necesario consultar periódicamente los sitios (en Internet) de tales organizaciones y/o estar suscritos a las listas de correo que utilizan para su divulgación. Cada organización proporciona información de acuerdo al foco del negocio que manejen u ofrezcan (antivirus, software, hardware, etc.), lo cual implica que la información está muy dispersa y se requiere de talento humano que esté constantemente

consultando las fuentes de datos, analizando la información y tomando los correctivos necesarios con base en la información recibida.

Dado que es necesario tener un control de las vulnerabilidades –o tener este riesgo lo más bajo posible –, nace la necesidad de crear una herramienta que, basada en ontologías[7] y en gestión del conocimiento aplicado a la seguridad informática, sea capaz de realizar la recopilación, análisis de la información y transformarla en conocimiento interrelacionado y eficaz. Una vez esta información está almacenada, puede ser recuperada reduciendo el tiempo de respuesta a la información de vulnerabilidades de seguridad de manera que las acciones correctivas puedan ser materializadas al interior de una organización de manera más rápida al no depender exclusivamente de personas para realizar las acciones de búsqueda y clasificación.

En esta prueba de concepto la actividad de clasificación, al igual que cuando es realizada por seres humanos, no se espera que tenga una eficiencia del 100% en la clasificación, sino que clasifique adecuadamente al menos el 75% de la información procesada.

El uso de ontologías en el desarrollo de Sistemas de Información (SI) permite establecer correspondencia y relaciones entre la forma como diferentes aplicaciones denominan las diferentes entidades de información en un dominio del conocimiento específico, creando un *lenguaje* común estandarizado.

4. OBJETIVOS

4.1. GENERAL

Implementar una herramienta Informática basada en ontologías para la clasificación automática de Información, recibida en forma de correo electrónico, sobre vulnerabilidades en herramientas software.

4.2. ESPECÍFICOS

- Analizar herramientas para la construcción de ontologías.
- Analizar lenguajes para la construcción de ontologías.
- Verificar y clasificar las entidades que proporcional información a cerca de las vulnerabilidades que van surgiendo.
- Construir una ontología sobre vulnerabilidades en herramientas software.
- Construir la herramienta para la clasificación automática de Información

4.3. ALCANCE

El alcance del proyecto estará enfocado a clasificar automáticamente al menos el 75% de las vulnerabilidades de herramientas software que se reciban mediante correo electrónico. Al ser una prueba de concepto, el alcance excluye intencionalmente muchos elementos que son necesarios para poder hablar de un producto de software. Esto se debe a que lo que se pretende es mostrar que el concepto es realizable.

5. MARCO TEÓRICO

En esta sección, se dan a conocer las diferentes investigaciones referentes a los conceptos teóricos necesarios para el desarrollo de la solución al problema.

5.1. CONCEPTOS SOBRE SEGURIDAD INFORMÁTICA

5.1.1. **¿Qué es Seguridad Informática?**

Se define Informática, como el conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores [8]. Ahora, seguridad informática, es la protección exclusiva de este tratamiento automático de la información, enfocando primordialmente en las herramientas, tecnologías y servicios.

“La seguridad informática busca la protección contra los riesgos ligados a la informática. Los riesgos son en función de varios elementos:

- Las amenazas que pesan sobre los activos (datos) a proteger
- Las vulnerabilidades de estos activos
- Su sensibilidad, la cual es la conjunción de diferentes factores:
    - la confidencialidad,
    - la integridad
    - la disponibilidad o accesibilidad

    - Revisión anticipada de la información” [9]

La seguridad informática se ocupa, en asegurar los recursos de una infraestructura informática determinada, de tal forma que se dé un uso adecuado tal cual fue definido.

5.1.2. **¿Qué es una Vulnerabilidad de Seguridad?**

Podemos definir que una vulnerabilidad de seguridad es una “debilidad” de algún sistema, que es la posibilidad de un ataque (cualquiera que sea la probabilidad). Para que un sistema sea comprometido debe tener una vulnerabilidad y debe existir una forma de explotar la vulnerabilidad. A la forma de explotar una vulnerabilidad la denominamos amenaza. Cuando un tercero trata de utilizar una amenaza para explotar una vulnerabilidad se tiene un ataque. Si el ataque es exitoso se produce un incidente.

Las vulnerabilidades como tal, una vez encontrada, puede ser divulgadas o no dependiendo de la persona, organización (productoras de software) o ente que la descubrió (y probablemente la sometió a pruebas de éxito). Si la vulnerabilidad es conocida pero no divulgada, el riesgo a que esta expuesto todo sistema informático, debe asumirse [10].

La divulgación de las vulnerabilidades de seguridad está en cabeza, principalmente, de las organizaciones productoras de software, donde éstas reconocen que sus sistemas poseen debilidades ante un atacante. Para la divulgación, se incluyen siguientes pasos:

- **Descubrimiento:** Existe una posible vulnerabilidad, descubierta por un ente cualquiera.
- **Notificación:** El descubridor informa al fabricante.
- **Investigación:** El fabricante indaga, prueba y verifica, puede o no solicitar apoyo del descubridor.
- **Resolución:** El fabricante acuerda una corrección y los plazos de publicación.
- **Publicación:** Fabricante o sus canales de información, publican la vulnerabilidad y su posible su corrección

Las vulnerabilidades pueden tener asociados agujeros de seguridad, que es un fallo en un programa que permite mediante su explotación, violar la seguridad de un sistema informático.

5.1.3. **Organismos más importantes que divulgan las vulnerabilidades.**

Dentro de las organizaciones mas relevantes, se encuentran las casas de antivirus, casas de desarrollo de software y sistemas operativos, algunas de ellas son:

- Casas de antivirus: Norton Antivirus, McAfee security, Sophos, Panda Sofware, etc.
- Casas de Software y Herramientas: Microsoft Corporation, Sun Microsystem, IBM, Novell, Symantec, Oracle Corporation, Mozilla Project, Linux Project (GNU), ISS, Cisco System, VSAntivirus, entre otras.

Adicional a esto, existen entidades (canales) que divulgan vulnerabilidades a través de su página Web o sus listas de correo. Algunos de ellos son:

- Hispasec Sistemas: Es un laboratorio especializado en Seguridad y Tecnologías de la Información. E-mail unaaldia-admin@hispasec.com
- SANS Instituto: Organización especializada en la formación técnica sobre seguridad informática.
- ACIS: Asociación Colombiana de Ingenieros de sistemas. e-mail segurinfo@acis.org.co , segurinfo-bounces@acis.org.co
- ETEK: Empresa líder en soluciones integrales de seguridad en informática.
- RedIRIS: La red académica y de investigación nacional española. E-mail cert@rediris.es
- CERT: Es un equipo de respuesta de emergencias a incidentes de seguridad en los Estados Unidos. E-mail cert@cert.org
- CriptoRed: Red temática iberoamericana de criptografía y seguridad de la información.
- BUGTRAQ: listserv@lists.securityfocus.com
- Best of Security: best-of-security-request@suburbia.net
- Linux Security: linux-security-request@redhat.com
- Linux Alert: linux-alert-request@redhat.com
- Computer Privacy Digest: comp-privacy-request@uwm.edu
- WWW Security: www-security-request@nsmx.rutgers.edu

5.1.4. **Principales vulnerabilidades de seguridad para el año 2004-2005.**

Para el año 2004, la entidad SANS Institute [11] conjuntamente con el FBI, publicaron las 20 vulnerabilidades mas importantes de Internet, esta publicación es el resultado arrojado por docenas de expertos en la materia, y en el cual se

muestra 2 listas: los diez servicios vulnerables más comúnmente explotados de Windows y los diez elementos más comúnmente explotados en los ambientes UNIX y Linux.

SANS Institute actualmente es la fuente más grande y de mayor confianza para el entrenamiento y la certificación de la seguridad de la información en el mundo, también desarrolla, convierte y mantiene sin ningún costo, la colección más grande de documentos de investigación sobre varios aspectos de la seguridad de la información.

Las siguientes vulnerabilidades son el resultado de estas investigaciones:

Vulnerabilidades para los sistemas Windows:

- W1. Servidores Web y servicios asociados.
- W2. Servicio Workstation.
- W3. Servicios de acceso remoto de Windows.
- W4. Microsoft SQL Server.
- W5. Autenticación de Windows.
- W6. Navegadores Web.
- W7. Aplicaciones de compartición de archivos.
- W8. Subsistema LSAS.
- W9. Clientes de correo electrónico.
- W10. Programas de Mensajería instantánea.

Vulnerabilidades para los sistemas Unix - Linux:

- U1. BIND: Sistema de nombres de dominio.
- U2. Servidor Web
- U3. Autenticación

- U4. Sistema de control de versiones.
- U5. Servicio de transporte de mail.
- U6. Protocolo SNMP: Simple Network Management Protocol.
- U7. SSL: Open Security Sockets Layer.
- U8. Mala Configuracion de servicios de red: NIS/NFS
- U9. Bases de datos.
- U10. Kernel: Núcleo del sistema operativo.

5.1.5. **Como controlar las vulnerabilidades *[12].***

Una vez conocida la vulnerabilidad, las entidades que la divulgaron, informan adicionalmente las posibles soluciones que se deben implementar para corregir la vulnerabilidad.

En el caso de entidades como Microsoft, se envía un parche de seguridad, que de acuerdo a su criticidad debe ser implementado de inmediato. En el caso de servicios o configuraciones, se hacen las recomendaciones que los expertos en el tema indican.

5.1.6. **Consecuencias de no controlar las vulnerabilidades a tiempo.**

En el caso específico de la seguridad informática, tener información apoya en gran parte a los oficiales de seguridad. Dentro de sus labores está la de discriminar entre la inmensa cantidad de vulnerabilidades que se reportan diariamente para centrar su atención en aquellas que afectan la seguridad de la organización.

Teniendo el conocimiento sobre las vulnerabilidades, se puede reducir el riesgo de de ataques a la infraestructura tecnológica de una organización.

Esta reducción del riesgo se ve reflejada en reducción de costos en administración y operación, dado que cuando no se tiene información al día de las vulnerabilidades de seguridad estas se pueden materializar en los sistemas de cómputo y las consecuencias son desastrosas y las perdidas son muchas (económicamente, perdida de imagen, perdida de confianza en los sistemas).

5.2. ONTOLOGÍAS

5.2.1. **¿Qué es una Ontología?**

Una de las definiciones que se tiene sobre Ontología, es que es una especificación formal explícita de cómo representar los objetos, los conceptos y otras entidades que se asumen para existir en un cierto campo de interés y las relaciones que sostienen entre ellas [13].

La construcción de ontologías en la informática tiene su origen en los sistemas basados en el conocimiento y en las tecnologías de la inteligencia artificial que utilizan la lógica para representar conocimiento. Sin embargo, su estandarización es un campo de acción relativamente reciente. No se debe confundir la noción de ontología en sistemas de información con el concepto filosófico.

Los sistemas de recuperación de información[14] basados en ontologías son genéricos en el sentido de que el conocimiento es un parámetro de su funcionamiento**.**

La definición de Ontología que se manejará en este proyecto de grado y de manera más precisa [15], fue extraído de Documento de requisitos de OWL [16]:

“Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos,

y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento […]. Las ontologías incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, que son útiles para los ordenadores [...]. Codifican el conocimiento de un dominio y también el conocimiento que extiende los dominios. En este sentido, hacen el conocimiento reutilizable".

5.2.2. **Características y Clasificación de Ontologías**

De acuerdo al Van Heist [17], las ontologías pueden clasificarse en dos dimensiones:

- La cantidad y el tipo de estructura de la conceptualización
- El tema de la conceptualización.

Con respecto a la primera dimensión, se distinguen tres (3) categorías:

- *Ontologías terminológicas:* Especifican los términos que son usados para representar el conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un campo determinado.
- *Ontologías de información*: Especifican la estructura de registro de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información. Los esquemas de la base de datos (Motores) son un ejemplo de esta clase de ontologías.
- *El conocimiento que modela la ontología*: Especifican conceptualizaciones del conocimiento. Contienen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen.

Para la segunda dimensión, se distinguen 4 categorías:

- *Ontologías del Uso:* contienen todas las definiciones que son necesarias modelar el conocimiento requerido para un uso particular.
- *Ontologías del Dominio:* expresan las conceptualizaciones que son específicas para los dominios particulares.
- *Ontologías genéricas:* son similares a los ontologías del dominio, pero los conceptos que definen se consideran ser genéricos a través de muchos campos.
- *Ontologías de la representación:* Las conceptualizaciones que hace unos formalismos de la representación del conocimiento, Se piensan para ser neutrales con respecto a entidades del mundo. Un ejemplo de esta categoría es ***Ontolingua [18].***

5.2.3. **Usos de las ontologías**

Las ontologías pueden aplicarse en:

- *Sistemas de Información:* El uso de ontologías explícitas en el desarrollo y uso de sistemas de información lleva a los que son llamados Sistemas de Información basados en ontologías.
    - Repositorios para la organización de conocimientos e información.
    - Herramienta para la adquisición de información.
    - Herramienta de referencia en la construcción de sistemas basados en el conocimiento.
    - Para permitir la reutilización del conocimiento ya existente.
    - Como base para la construcción de lenguajes de representación del conocimiento.
- *Comunicación:* En modelos normativos, crea la semántica de un sistema y el modelo para extenderlo y transformarlo entre diferentes contextos. Permite la comunicación entre sistemas de información.
- *Interoperabilidad:* usa ontologías como una inter-lengua.

Adicional a esto, las ontologías pueden ser utilizadas para:

- Reutilización: Una ontologías como codificación de un dominio, puede ser reutilizada y/o compartida.
- Búsquedas: Los términos de las ontologías pueden usarse como meta-datos para indexar documentos.
- Librerías digitales.
- Especificaciones: Pueden guiar el proceso de especificaciones de un sistema.
- Mantenimiento: Las ontologías pueden incluirse como parte de la documentación, reduciendo costos de mantenimiento.
- Adquisición del conocimiento: Un sistema basado en conocimiento, puede construirse más fácil y fiable, si se parte de una ontología.

Donde algunos escenarios de aplicación pueden ser:

- Autoría neutral: construcción de ontologías para su posterior uso en varios sistemas. La idea es construir un artefacto en un lenguaje dado y traducirlo a los correspondientes formatos de varios sistemas.
- Ontología como especificación: Se trata de construir una ontología con el vocabulario y los requerimientos de una o varias aplicaciones, y utilizarla para implementar distintos sistemas.
- Acceso común a la información: Consiste en utilizar ontologías para permitir que distintos agentes (humanos o software) accedan a fuentes de información heterogéneas
- Búsquedas basadas en Ontologías: Se trata de utilizar una ontología para ayudar en la identificación de los recursos que el usuario busca, esto mejora la eficiencia del proceso de búsqueda.
- Comercio electrónico: Se busca facilitar el intercambio de información entre los participantes de operaciones comerciales.

- Portales. Los portales son sitios de entrada a la web que agrupan servicios de uso común tales como directorios, correo, noticias, forums, etc. Los portales sirven para hacer más eficiente la comunicación de información dentro de la organización.
- Como repositorios para la organización de conocimientos e información, tanto de tipo corporativo como científico.
- Como herramienta para la adquisición de información, en situaciones en la que un equipo de trabajo la utiliza como soporte común para la organización del dominio.

5.2.4. **Partes y conceptos de la Ontología**

Es necesario tener un conocimiento mas profundo de la conformación de una ontología, los siguientes elementos descritos son propios de una ontología:

a. **Un dominio**: Contexto específico. Es el área del conocimiento que se pretende hacer explicita mediante la construcción de la ontología.
b. **Clases:** Descripción formal explicita de los conceptos de un dominio, éstas contienen individuos.
c. **Subclase:** Es una especialización del concepto definido por la clase padre.
d. **Propiedades**: Son las características, atributos o relaciones que existen entre individuos o entre clases. En algunos estándares para definición de ontologías se denominan *slot*s.
e. **Restricciones:** son atributo de las propiedades. Pueden ser cardinalidad, funcionalidad, operabilidad, tipo, etc.

El desarrollo de una Ontología requiere definir un dominio de conocimiento y las clases que conforman el dominio, organizar esas clases en una taxonomía, definir las propiedades de cada clase y las restricciones de los valores (en el caso que las posea), y por último, asignar valores las propiedades para crear instancias.

En la siguiente gráfica se representa algunos de estos conceptos [19]:

Clases: Person, Pet y Country.
Individuos por clase:
Person: Matthew y Gemma
Pet: Fluffy y Fido
Country: Italy, England, USA
Relaciones: HasSibling, LivesInCountry y HasPet

5.2.5. **Construcción de una ontología: Consideraciones**

Para la construcción de una ontología, es necesario tener en cuenta un proceso de diseño, adicional a esto, se debe cuestionar por algunos aspectos relevantes que acoten la idea de modelamiento que se desea realizar.

La construcción de una ontología depende en gran manera del contexto en que se construye, además hay que tener en cuenta que una ontología especifica una conceptualización, una forma de ver el mundo, por lo cual una ontología incorpora

un punto de vista, definiciones para referirse a un dominio (a través de un vocabulario) y éstas dependen del lenguaje que se usa para describirlas.

Algunos autores expresan que, para el diseño de una ontología, es necesario tener en cuenta los siguientes aspectos:

- *Claridad:* Los términos de una ontología debe comunicar efectivamente el significado del dominio en que esta. Las definiciones deben ser objetivas y comentadas en lenguaje natural[20].

- Consistencia: Las inferencias realizadas sobre las definiciones, deben ser consistentes, es decir que no deben generar contradicciones.
- *Extendible*: Debe anticipar usos y permitir extensiones y especializaciones monotónicas.
- *Sesgo de codificación mínimo (Minimal encoding bias):* El nivel del conocimiento especificado, no debe depender de la codificación particular al nivel de símbolo.
- *Mínimo compromiso ontológico:* Debe hacer la menor cantidad posible de "pretensiones" acerca del mundo modelado.

De igual manera, es conveniente tener resuelta la frontera hacia donde queremos ir o representar con la Ontología, para ello, es importante responder:

a. *Qué dominio de conocimiento se incluirá?*
b. *Cuál es el contexto en que se construirá (para que se va a usar)?*
c. *Cuál es el alcance?*
d. *Qué herramienta y lenguaje para la construcción de ontologías se utilizará?*
e. *Cómo se obtiene el conocimiento sobre el dominio escogido?*
f. *Cómo identificar y seleccionar las fuentes potenciales de información?*
g. *Cuál es el público objetivo?*
h. *Cómo se mantendrá actualizada la ontología*?

Teniendo ya un alcance definido, podemos adentrarnos a sugerencias que acotan de una manera más eficaz, la construcción de ésta:

- Definiciones: Identificación de los conceptos y relaciones claves en el dominio de interés, producción de definiciones no ambigüas de conceptos y de sus relaciones e identificación de términos para referirse a estos conceptos y relaciones.
- Codificación: representación explícita de la conceptualización en un lenguaje formal.
- Lenguaje de representación adecuado. Codificación de este lenguaje, verificación de la posible integración de ontologías existentes.
- Evaluación: Se considera que la ontología construida va a ser reutilizada por lo tanto debe seguir unos principios básicos, como la abstracción, la modularización y la jerarquización.
- Documentación: Debe de hacerse de forma paralela a los puntos anteriores.

5.2.6. **Aplicaciones reales de las Ontologías:**

En la actualidad, existen grandes aplicaciones y desarrollos de software apoyados en Ontologías, algunos de los más relevantes son:

- *Ontolingua:* Ontolingua proporciona un ambiente de colaboración distribuido para mirar, para crear, para corregir, para modificar, y para utilizar ontologías, es un conjunto de herramientas, escritas en Common Lisp, para el análisis y traducción de ontologías.
- *WordNet*: WordNet es una base de datos léxico-conceptual del inglés estructurada en forma de red semántica -es decir, compuesta de unidades léxicas y relaciones entre ellas-, que pretende ser un modelo del

conocimiento léxico-conceptual de los hablantes de inglés. Organiza verbos, adverbios, sustantivos como conjuntos de sinónimos (Synset) donde cada uno representa un concepto léxico. Cada Synset se relaciona con otros por medio de relaciones semánticas.

- *KA*$^{2:}$ Knowledge Acquisition Community - Tiene como objetivo el razonamiento basado en el conocimiento en la Web, en comparación con la recuperación de datos más generales. Desarrollado por la Universidad de Karlsruhe, se trata de una Ontología para la comunidad investigadora internacional y heterogénea, permite el acceso semántico a fuentes de información on-line de la comunidad.
- *SUMO:* Standard Upper Merged Ontology - Proporciona una fundación para los ontologías de nivel medio y del dominio, y su propósito es promover interoperabilidad de los datos, la recuperación de datos, la inferencia automatizada, y el procesamiento del lenguaje natural. Se trata de una Ontología universal a la que puedan engancharse todas las ontologías de diferentes dominios.
- *CyC:* Cycorp fue fundado en 1994 para investigar, para desarrollar, y para comercializar la inteligencia artificial. La visión de Cycorp es crear la primera inteligencia artificial verdadera del mundo, teniendo sentido común y la capacidad de razonar con ella.
- *Frame Ontology:* Define los términos del las convenciones de captura usados en sistemas de representación de conocimiento. El objetivo de esta ontología es permitir a gente usando diversos sistemas de representación para compartir ontologías que se organizan para representar conocimiento centrado en objetos (*object centered*).
- *GALEN:* Ontología para la medicina (http://www.opengalen.org/)
- *WebOnto:* WebOnto fue desarrollado por el Knowledge Media Institute de la Universidad Abierta (Open University). Se diseñó para soportar búsquedas colectivas y creación y edición de ontologías. Proporciona una interfaz de

modificación directa presentando en pantalla las expresiones ontológicas y también una herramienta de debate sobre ontologías llamado Tadzebao.

5.2.7. **Motores de Inferencia**

Un motor de inferencia es un programa ubicado entre el usuario y una base de conocimiento, y su objetivo es extraer conclusiones a partir de los datos simbólicos que están almacenados en las bases de hechos y de conocimiento. Dependen en gran medida de la representación elegida.

Existen varios algoritmos de búsqueda a lo largo de las reglas para inferir conclusiones a partir de los hechos y las reglas. Todos los algoritmos son del tipo "pattern-matching", van disparando reglas a medida que se cumplen las condiciones. Se pueden diferenciar dos mecanismos de inferencia:

- Encadenamiento hacia delante: se extraen conclusiones a partir del cumplimiento de las condiciones de ciertas reglas que, a su vez, provocarán el cumplimiento de las condiciones en otras reglas hasta que no se cumplan en ninguna de ellas.
- Encadenamiento hacia atrás: se suponen ciertas las conclusiones de una regla y, como consecuencia, se van disparando aquellas reglas que provocarían la regla original. El proceso acaba si no se cumplen las condiciones de las reglas o si se cumplen para todas las reglas

Es importante destacar que la base de conocimientos y el motor de inferencia son sistemas independientes, por ello, ambos pueden correr y modelarse independientemente.

Los motores de inferencia permiten combinar conocimientos conocidos para elaborar nuevos conocimientos, deduciendo estos nuevos conocimientos del conocimiento ya especificado.

Un sistema experto, es un programa que infiere conclusiones a partir del conocimiento almacenado, utilizando como insumos dos tipos de elementos: Los datos (hechos y evidencias) y el conocimiento (el conjunto de reglas almacenadas en la base de conocimientos).
Para hallar estas conclusiones, los motores de inferencia utilizan dos estrategias de inferencia: encadenamiento de reglas y encadenamiento de reglas orientadas a un objeto, obteniendo conclusiones simples y compuestas respectivamente.

Las reglas de inferencia más comunes utilizadas por los motores son la Modus ponens y Modus Tollens. El rendimiento del motor de inferencia depende del conjunto de reglas en su base de conocimiento. Hay ocasiones en que el motor de inferencia puede concluir utilizando un cierto conjunto de reglas, pero no puede utilizando otro, aunque sean lógicamente equivalentes.

El encadenamiento de reglas es una de las estrategias de inferencia más utilizadas para obtener conclusiones. Esta estrategia puede utilizarse cuando las premisas de ciertas reglas coinciden con las conclusiones de otras. Cuando se encadenan las reglas, los hechos pueden utilizarse para dar lugar a nuevos hechos, esto se repite sucesivamente hasta que no puedan obtenerse mas conclusiones.

Los motores de inferencia realizan un control de coherencia. Este control consiste en no tener hechos inconsistentes y evitar dentro de la base de conocimiento cualquier tipo de conocimiento contradictorio. Además, este control de coherencia debe hacerse controlando la coherencia de las reglas y la de los hechos, Por ello, cualquier valor no factible debe ser eliminado de la lista de valores posibles, de su

correspondiente objeto para eliminar la posibilidad de que el motor de inferencia pueda obtener conclusiones inconsistentes.

5.3. WEB SEMÁNTICA

5.3.1. ¿Qué es la Web Semántica?

La Web Semántica es una extensión de la actual Web que permitirá encontrar, compartir y combinar la información más fácilmente. Es una iniciativa liderada por Tim Berners-Lee [21] para permitir que la información que reside en la red sea accesible y "comprensible" no sólo por los humanos sino también por las máquinas.

La Web semántica tiene como objetivo crear un medio universal para el intercambio de información basado en representaciones del significado de los recursos de la Web, de una manera inteligible para las máquinas. Con ello se pretende ampliar la interoperabilidad entre los sistemas informáticos y reducir la mediación de operadores humanos en los procesos inteligentes de flujo de información [22].

La Web semántica es un área fuerte nacida en la inteligencia artificial y las tecnologías Web, que propone nuevas técnicas y paradigmas para la representación del conocimiento que faciliten la localización, compartición e integración de recursos a través de la Web. Es una gran combinación de información enlazada y codificada en tal forma que puede ser fácilmente procesada por máquinas al involucrar reglas de procesamiento semántico. La manera de realizar el procesamiento se lleva a cabo con las distintas etiquetas que se encuentran dentro del código XML, las cuales poseen un contexto determinado para poder procesarlas [23].

La Web semántica mantiene los principios que han hecho un éxito de la Web actual, como son los principios de descentralización, compartición, compatibilidad, y la apertura al crecimiento.

Según Berners Lee: "La idea de la Red es crear un espacio de información en el que la gente puede comunicarse de manera muy precisa: compartiendo sus conocimientos" [24]. La visión de la Web semántica modificaría la forma en que se presenta la información en la Web para facilitar su procesamiento por parte de las computadoras, en cierta forma se habla de un reordenamiento, de ordenar el caos.

5.3.2. **¿En qué aplica?**

La comunicación prácticamente con todo el mundo en cualquier momento y a bajo coste es posible hoy en día. Podemos realizar transacciones económicas a través de Internet. Tenemos acceso a millones de recursos, independientemente de nuestra situación geográfica e idioma. Todos estos factores han contribuido al éxito de la Web. Sin embargo, al mismo tiempo, estos factores que han propiciado el éxito de la Web, también han originado sus principales problemas: sobrecarga de información y heterogeneidad de fuentes de información con el consiguiente problema de interoperabilidad.

La Web Semántica ayuda a resolver estos dos importantes problemas permitiendo a los usuarios delegar tareas en software. Gracias a la semántica en la Web, el software es capaz de procesar su contenido, razonar con éste, combinarlo y realizar deducciones lógicas para resolver problemas cotidianos automáticamente [***25***].

5.3.3. **Ventajas de la Web Semántica.**

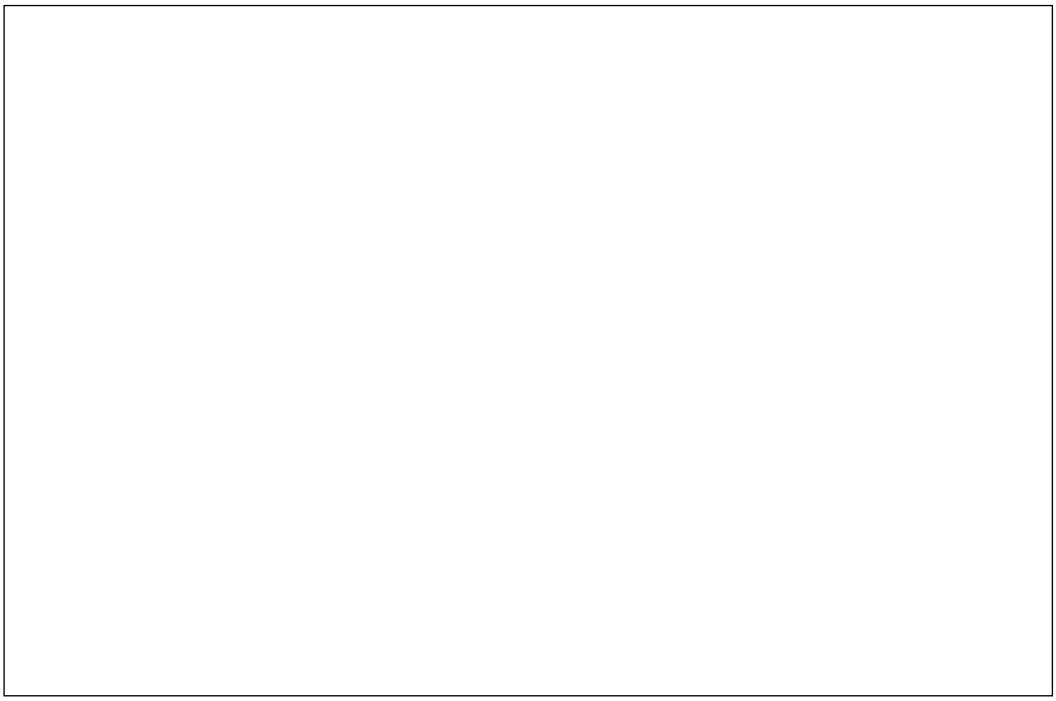
Una de las ventajas de la Web semántica es el desarrollo de aplicaciones con esquemas de datos comunes, fomento de las transacciones entre empresas por

comercio electrónico y búsqueda de información con inferencias, además, se pueden construir ontologías, obteniendo integración de información de la plataforma o del idioma.

También permitirán al usuario que, cuando use palabras con varios significados, especifique la categoría de las palabras que usa para buscar, así se evitará la abundancia de resultados inútiles que obtenemos al hacer cualquier búsqueda en la Web.

5.3.4. **Componentes de la Web Semántica**

Los principales componentes de la Web Semántica son los metalenguajes y estándares de representación. La nueva red semántica ha sido estructurada por niveles, estableciendo una jerarquía de abstracción y unas dependencias entre los distintos niveles, la siguiente estructura ilustra los componentes que hacen parte de la Web Semántica (Figura 2: arquitectura) [26]:



Estos niveles son definidos en 5 grandes grupos:

Nivel de Recursos: En el primer nivel se incluye la identificación de recursos Web, estableciendo así la capital importancia que tiene definir el conjunto de recursos distribuidos por la red.

Nivel Sintáctico: En este nivel se soluciona el problema de cómo definir distintos lenguajes de etiquetados para añadir contenido semántico a las páginas Web.

Nivel de Descripción de Recursos: RDF (Resource Description Framework) es una recomendación de W3C diseñada para estandarizar la definición y el uso de las descripciones de metadatos de los recursos Web.

Nivel de Ontologías: Las ontologías son la piedra angular de la propuesta de Berner-Lee. Para que esto se pueda llevar a cabo, el conocimiento no sólo debe ser legible por una máquina, sino que debe ser consensuado y reutilizable. OWL es oficialmente (según W3C) la forma más apropiada de compartir conocimiento en la web.

Resto de niveles: El nivel de la lógica pretende dar flexibilidad a la arquitectura para realizar consultas e inferir conocimiento a partir de las ontologías de la capa anterior.

5.3.4.1. XML

Fue desarrollado por el World Wide Web Consortium (W3C), Su objetivo principal era conseguir una página Web más semántica, separando la estructura del contenido y permitiendo el desarrollo de vocabularios modulares, compatibles con cierta unidad y simplicidad del lenguaje.

XML es un formato de texto simple, muy flexible derivado de SGML (ISO 8879). Diseñado originalmente para resolver los desafíos de la gran publicación electrónica, XML también está desempeñando un papel cada vez más importante en un amplio intercambio datos sobre la Web.

La idea general detrás de XML es que pueda ser generado y leído fácilmente por una máquina, procurando crear un estándar que sea independiente de los idiomas, lenguajes de programación, plataformas y que no presenten mayor complejidad, de manera que pueda ser utilizado como método para el intercambio de información estructurada independiente de las aplicaciones y las plataformas que se utilizan para crearla y manipularla.

5.3.4.2. **SHOE** *[27]*

SHOE, es una extensión al HTML y proporciona una manera de incorporar conocimiento semántico legible por la máquina en HTML u otros documentos de la Web. SHOW se puede también utilizar con los documentos de XML, con las siguientes especificaciones:

- Un mecanismo jerárquico de la clasificación para documentos del HTML o las subdivisiones de los documentos del HTML.
- Un mecanismo para especificar relaciones entre los elementos clasificados y otras clases clasificadas del elemento o específicas de los datos (números, fechas, etc.)
- Una manera simple de especificar los ontologías que contienen las reglas que definen clasificaciones válidas, relaciones, y reglas deducidas.

El objetivo es facilitar búsquedas y anotaciones de documentos, es recopilar la información verdaderamente significativa sobre las páginas y los documentos de la

Web, permitiendo una mejor búsqueda a través de un mecanismo con un conocimiento central.

5.3.4.3. OIL

OIL (Ontology Inference Layer) es una oferta para una representación basado en Web e inferencia para las ontologías, que combina el uso de modelos primitivos desde lenguajes basados en frames con una semántica formal y provee un servicio de razonador por descripciones lógicas [28].

OIL presenta un acercamiento a un lenguaje para ontologías. Cada capa adicional agrega funcionalidad y complejidad a la capa anterior. Esto hace que los agentes (los seres humanos o las máquinas) que puede procesar solamente una capa más baja pueden sin embargo entender parcialmente los ontologías que se expresan en cualesquiera de las capas más altas.

5.3.4.4. XOL

XOL es un lenguaje para el intercambio de ontologías, es diseñado para proporcionar un formato que puede intercambiar definiciones de ontologías entre sistemas.

La definición de ontología que es diseñada por XOL, incluye la codificación de ambos esquemas de información (meta-datos) [29].

5.3.4.5. DAML+OIL

DAML (DARPA Agent Markup Language) tiene como esfuerzo global desarrollar un lenguaje y una herramienta para facilitar los conceptos de la Web semántica, es un lenguaje creado por DARPA como un lenguaje de inferencia y ontología basado en RDF.

DAML+OIL fusión de DAML (DARPA Agent Markup Language) y OIL (Ontology Inference Layer) que amplía RDF(S) con primitivas de lógica de descripciones,

proporciona un sistema rico en construcciones para la creación de ontologías y un beneficio en los frames, de modo que sea legible y comprensible por la maquina [30].

5.3.4.6. **RDF** *[31]*

RDF (Resource Description Framework) Es un modelo de meta-datos para los recursos y las relaciones que se puedan establecer entre ellos, que ha sido desarrollada por el World Wide Web Consortium (W3C). Aporta una semántica básica para este modelo de datos que puede representarse mediante XML.

Este modelo se basa en la idea de convertir las declaraciones de los recursos en expresiones con la forma sujeto-predicado-objeto (conocidas en términos RDF como tripletas). El sujeto es el recurso, es decir aquello que se está describiendo. El predicado es la propiedad o relación que se desea establecer acerca del recurso. Por último, el objeto es el valor de la propiedad o el otro recurso con el que se establece la relación. La combinación de RDF con otras herramientas como RDF Schema y OWL permite añadir significado a las páginas, y es una de las tecnologías esenciales de la Web semántica [32].

5.3.4.7. **OWL** *[33]*

OWL (Web Ontology Language) es un lenguaje de Ontologías *Web.* Añade más vocabulario para describir propiedades y clases: tales como relaciones entre clases, igualdad, tipologías de propiedades más complejas, caracterización de propiedades (por ejemplo simetría) o clases enumeradas.

OWL es una propuesta del W3C a partir de DAML+OIL para responder a las necesidades de la Web semántica. OWL proporciona un lenguaje para la definición de Ontologías estructuradas, basadas en la Web, que ofrece una integración e interoperabilidad de datos más rica entre comunidades descriptivas.

Es el desarrollo mas reciente en lenguajes estándares para ontologías, construido W3C, facilita el mayor interoperabilidad de la máquina con los contenidos de la Web y esta apoyado por esquemas de XML, RDF, y RDF (Rdf-s) proporcionando vocabulario adicional junto con una semántica formal.

OWL proporciona tres sub-lenguajes cada vez más expresivos, diseñados para el uso de las comunidades específicas de ejecutores y de usuarios:

- *OWL Lite:* Apoya a esos usuarios que necesitan sobre todo una jerarquía de clasificación y apremios simples. OWL *lite* está construido de tal forma que toda sentencia pueda ser resuelta en tiempo finito. OWL-Lite es el más sencillo de los tres. Cubre las necesidades de clasificar conceptos jerárquicamente y con restricciones sencillas, permitiendo sólo cardinalidad 0 ó 1.
- *OWL DL:* Fue diseñado para aquellos usuarios que requiriesen la máxima expresividad del lenguaje pero teniendo en cuenta la eficiencia computacional.
- *OWL Full:* es el más completo de los tres sublenguajes, permitiendo la misma expresividad que OWL DL y con toda la capacidad sintáctica que ofrece RDF, sin embargo añade una gran carga de procesamiento.

5.3.5. **W3C (World Wide Web Consortium)** ***[34]***

El W3C fue creado para guiar la Web hacia su potencial máximo mediante el desarrollo de protocolos comunes que promuevan su evolución y garanticen su interactividad. Se trata de un consorcio de la industria internacional gestionado conjuntamente por el Laboratorio de Ciencia Computacional e Inteligencia Artificial del MIT (MIT CSAIL) en EEUU, el Consorcio Europeo para la Investigación en Informática y Matemáticas (ERCIM) con sede en Francia y la Universidad Keio en

Japón. Los Servicios proporcionados por el Consorcio incluyen un almacén de información sobre el World Wide Web para desarrolladores y usuarios, y varios prototipos y aplicaciones de ejemplo para demostrar el uso de estas nuevas tecnologías. Hasta la fecha, cerca de 400 organizaciones son miembros del Consorcio. [35]

5.4. INTELIGENCIA ARTIFICIAL

Se define la inteligencia artificial como aquella inteligencia exhibida por artefactos creados por humanos (es decir, artificial). A menudo se aplica hipotéticamente a los computadores. El nombre también se usa para referirse al campo de la investigación científica que intenta acercarse a la creación de tales sistemas [36].

Algunos autores sostienen que la IA incluye características humanas tales como el aprendizaje, la adaptación, el razonamiento, la auto-corrección, el mejoramiento implícito, y la percepción de modelar el mundo.

Algunas técnicas y campos de la IA son:

- Aprendizaje automático.
- Ingeniería del conocimiento.
- Lógica difusa
- Redes neuronales artificiales.
- Sistemas reactivos.
- Sistemas multi-agente.
- Sistemas basados en reglas.
- Sistemas expertos.
- Redes Bayesianas
- Técnicas de representación del conocimiento (Redes semánticas y Frames)

La IA tiene las siguientes disciplinas:

5.4.1. **Lenguaje Natural**

Se utiliza el término lenguaje natural para referirse principalmente al lenguaje humano. El término se utiliza en contraposición a los lenguajes formales.

La lengua natural evoluciona enmarcada por una cultura de hablantes nativos que utilizan dicha lengua con una finalidad comunicativa. De esta forma, se distingue entre lenguas tales como el chino mandarín, el español y el inglés, las cuales son lenguas naturales; y el esperanto o interlingua, a las cuales se les denomina lenguas planificadas [37].

5.4.2. **Procesamiento de Lenguaje Natural (PLN)**

Estudia los problemas inherentes al procesamiento y manipulación de lenguajes naturales, sin embargo no suele plantear el entendimiento de lenguajes naturales

Son elementos de software que tienen la capacidad de sobrellevar una conversación con un ser humano de manera interactiva. Implican el uso de técnicas de la computación semántica.

Las principales tareas del PLN comprenden:

- Síntesis del discurso
- Análisis del lenguaje
- Comprensión del lenguaje
- Reconocimiento del habla
- Generación automática del lenguaje.
- Traducción automática
- Respuesta a preguntas
- Recuperación de la información
- Extracción de la información

Existen algunas dificultades con respecto al procesamiento del lenguaje natural, dentro de estas están:

Separación entre palabras, ambigüedad semántica, ambigüedad sintáctica, recepción imperfecta de datos, entre otras.

5.5. HERRAMIENTAS SOFTWARE Y LENGUAJES DE PROGRAMACIÓN

5.5.1. **Java**

Java es una plataforma de software desarrollada por Sun Microsystems [38], de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

La plataforma java consta de las siguientes partes: Una maquina virtual, un lenguaje de programación y la biblioteca estándar para el lenguaje.

La máquina virtual de Java (JVM) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

Java es independiente de la plataforma de desarrollo, Java permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.

Java ha sido diseñado de modo de eliminar las complejidades de otros lenguajes como C y C++. Java combina flexibilidad, robustez y legibilidad gracias a una mezcla de chequeo de tipos durante la compilación y durante la ejecución.

El principal objetivo de los diseñadores de Java, y dado el gran crecimiento de las redes en los últimos años, fue el de desarrollar un lenguaje cuyas aplicaciones una vez compiladas pudiesen ser inmediatamente ejecutables en cualquier máquina y sobre cualquier sistema operativo. La gran ventaja de la máquina virtual java es aportar portabilidad al lenguaje, para ello, se han creado diferentes máquinas virtuales java para diferentes arquitecturas y así un programa .class escrito en un Windows puede ser interpretado en un entorno Linux.

5.5.2. **Eclipse** ***[39]***

Es una comunidad *Open Source* que se centran en proveer una plataforma de desarrollo extensible y un framework de aplicación para la construcción de software.

El eclipse proporciona las herramientas extensibles y los framework que soportan el ciclo de vida del desarrollo del software, incluyendo la ayuda para modelar, los ambientes de desarrollo para Java, C/C++ y otros, pruebas y funcionamiento, inteligencia de negocio, aplicaciones fuertes para los cliente y desarrollo personalizado.

Eclipse es un IDE [40] de software, basado en java, que permite la creación, manipulación y desarrollo de programas para computadores; Soporta otros programas (a través de plugins [41]).

5.5.3. **UIMA (Unstructured Information Management Architecture)** ***[42]***

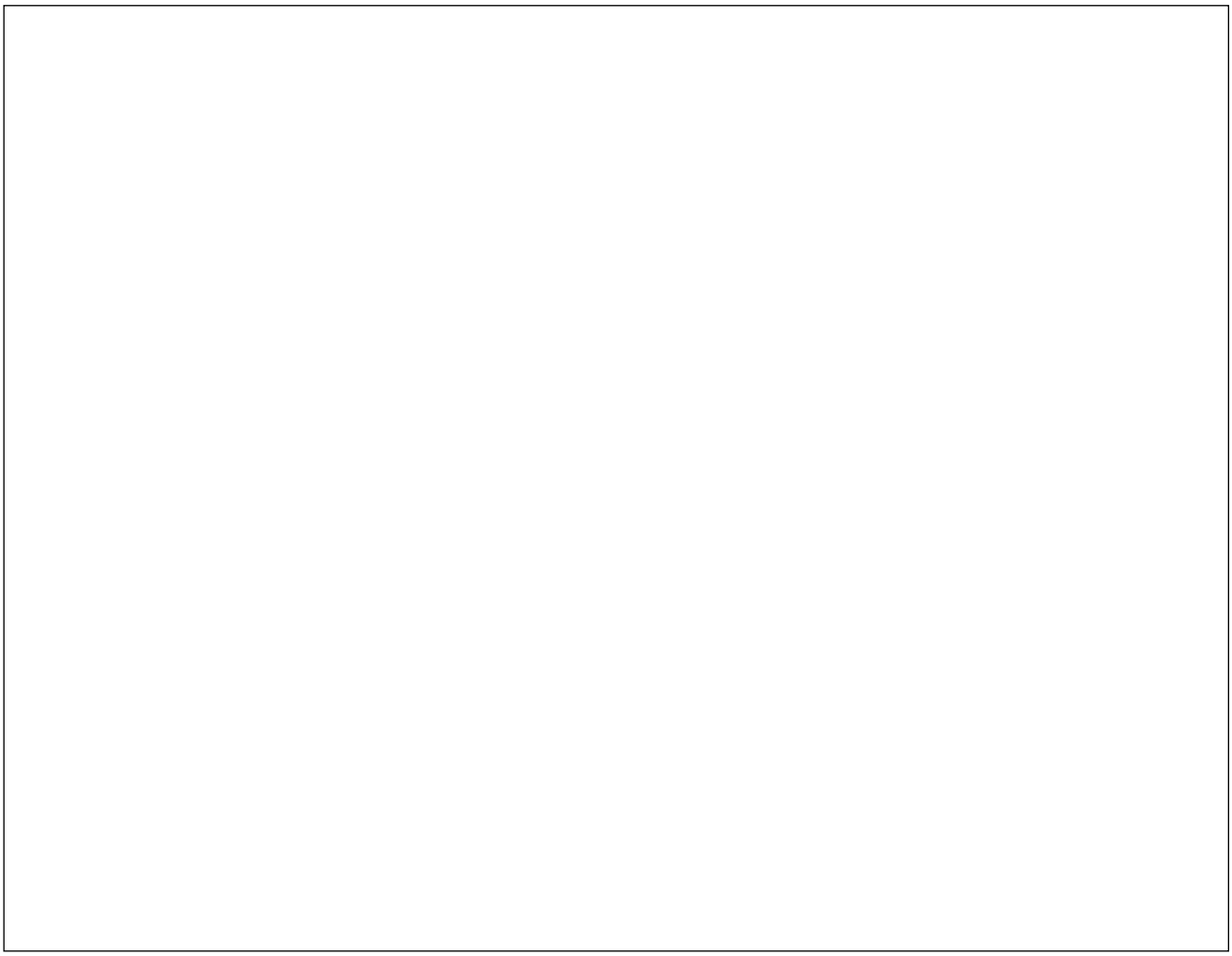
Es una plataforma abierta, fortalece la industria, escalable y **extensible** para el manejo de información no estructurada desde combinaciones de análisis semántico y búsquedas de componentes.

Creada por IBM [43] liberó UIMA y está disponible como software open source para proporcionar una fundamentación común para la industria y la academia. Colabora y acelera el desarrollo mundial de las tecnologías críticas para descubrir el conocimiento vital presente en las fuentes de crecimiento más rápidas de la información hoy.

IBM apuesta por la búsqueda basada en conceptos, en lugar de las tradicionales palabras clave, recuperar los datos en las redes corporativas. Esta es una tecnología que se puede utilizar para construir aplicaciones capaces de analizar textos, documentos y otros medios, para entender contenidos latentes, relaciones y hechos.

UIMA es una arquitectura y framework que ayuda a construir en puente desde la información no estructurada hasta el conocimiento estructurado. Es una plataforma de integración escalable, para componer motores de análisis e integrar sus resultados en sistemas back-end para el tratamiento de la información.

El siguiente gráfico representa la arquitectura de UIMA:



UIMA es una arquitectura en la cual los bloques de construcción básicos, llamados Analisys Engines (AEs), se componen para analizar un documento, deducir y para registrar cualidades descriptivas sobre el documento en su totalidad, y/o sobre regiones en éste. Los resultados del análisis representan típicamente metadatos sobre el contenido del documento.

UIMA apoya el análisis de diversas modalidades incluyendo el texto, el audio y el vídeo. Los resultados del análisis incluyen diversas declaraciones sobre el contenido de un documento. UIMA proporciona un tipo de componente básico previsto para contener los algoritmos de análisis con lo que funcionan los AE's.

5.5.4. **Visual Basic y Microsoft Outlook 2003.**

Visual Basic

Es un lenguaje de programación desarrollado por Microsoft Corporation [44]. Visual Basic es un lenguaje visual que desciende del lenguaje de programación BASIC. Su primera versión fue presentada en 1991 con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y en cierta medida también la programación misma.

Este lenguaje es orientado a eventos, utilizado principalmente en el World Wide Web para realizar consultas a bases de datos de Microsoft como Fox Pro, SQL-Server, etc.

Este lenguaje puede ser utilizado desde herramientas colaborativas de Microsoft Office, tales como Word, Excel o Outlook 2000 ó 2003. Para estas herramientas colaborativas, existen 2 derivaciones del lenguaje: *Visual Script (VBScript)* y *Visual Basic for Application (VBA)*.

VBScript es el lenguaje predeterminado para Active Server Pages (ASP) y VBA permite codificar módulos a veces llamados macros. Estas macros son grupos de instrucciones que tienen un desarrollo secuencial y son usadas para economizar tareas; una macro no es más que un conjunto de expresiones (instrucciones) tales como "borrar archivo", "añadir registro", etc., y que se almacenan en una ubicación especial.

Especialmente a partir de la versión 6 del lenguaje, se permite la interacción y generación de objetos remotos que puedan ser invocados desde páginas de scripts.

Algunas ventajas de Visual Basic son que es un lenguaje simple y por tanto fácil de aprender, su mayor facilidad radica en el dibujado de formularios, mediante el arrastre de controles y la sintaxis está cercana al lenguaje humano, permite el tratamiento de mensajes de Windows e integración nativa con las herramientas colaborativas.

Algunas desventajas es que es propietario de Microsoft, por tanto nadie que no sea del equipo de desarrollo de esta compañía decide la evolución del lenguaje. Sólo existe un compilador e IDE, llamado igual que el lenguaje, sólo genera ejecutables para Windows, la sintaxis es bastante inflexible y los ejecutables generados son relativamente lentos.

Microsoft Outlook 2003

Es un programa de agenda ofimática y cliente de email de Microsoft, y forma parte de la suite Microsoft Office.

Puede ser utilizado como aplicación independiente, o junto con Microsoft Exchange Server para dar servicio a múltiples usuarios, Outlook 2003 proporciona una solución integrada para administrar y organizar mensajes de correo electrónico, programas, tareas, notas, contactos y demás información. Outlook 2003 ofrece innovaciones que puede utilizar para administrar las comunicaciones, organizar el trabajo y trabajar mejor con otros usuarios [45].

Como se vio, en Microsoft Outlook 2003 se tiene la posibilidad de crear macros, la grabación de macros permite registrar una secuencia de acciones de modo que se pueda reproducir más tarde.

5.5.5. **Qué son y para qué se utilizan los plugins.**

Un plugin (o plug-in) es un programa de computador que interactúa con otro programa para aportarle una función o utilidad específica, Este programa

adicional es ejecutado por la aplicación principal. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos, obtener funcionales de programación adicionales, etc.

Se utilizan como una forma de expandir programas de forma modular, de forma que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes ni complicar el desarrollo del programa principal.

Algunas de las ventajas que traen los plugin son, además de extender las prestaciones del programa, que conceden capacidades extra para quien las necesite, sin que haga falta saturar de opciones los menús del programa, la mayoría son de uso libre (gratis), pueden activarse o desactivarse de acuerdo a la necesidad, permite actualizaciones sin daño ni perjuicio en las versiones anteriores.

5.5.6. **Herramientas para la construcción de Ontologías.**

Dentro de las herramientas para la construcción de ontologías, se tiene una gama amplia, de acuerdo a la necesidad, algunas herramientas se pueden instalar independientemente y otras pueden ser parte de un plugin de una herramienta base.

Las siguientes herramientas tienen como objetivo ayudar a la creación de las ontologías:

- **Chimaera** [46]**:** Es un software que soporta a usuarios en la creación y mantenimiento distribuido de ontologías en la Web.

  Dos funciones importantes que apoya esta herramienta son la combinación de ontologías múltiples y el diagnostico de ontologías individuales o múltiples.

Proporcionan un entorno distribuido y colaborativo para la creación, edición, modificación, navegación y utilización de ontologías mediante la Web.

- **Protégé** [47]**:**

Es una herramienta libre y *open source* para editar ontologías. Está basado en Java, es extensible y provee fundamentación para personalizar aplicaciones basadas en conocimiento.

Es una herramienta que permite al diseñador construir una ontología de un dominio y diseñar formularios personalizados para la adquisición de conocimiento. Así mismo la herramienta es una plataforma que puede ser ampliada con widgets gráficos para tablas, diagramas, u otro tipo de componentes para acceder a aplicaciones embebidas. Protégé además incorpora una librería que otras aplicaciones pueden utilizar para acceder y visualizar conocimiento base.

Protégé soporta Frames, XML Schemas, RDF(s) y OWL, tiene la posibilidad de adicionarles plugins para diferentes funcionalidades.

- **OntoEdit** [48]

Es una herramienta que permite la inspección, navegación, codificación y modificación de ontologías, soportando de esta forma el mantenimiento y el desarrollo de las mismas. En OntoEdit las ontologías se modelan con independencia de un lenguaje concreto de presentación, utilizando una interfaz gráfica de usuario para representar distintas vistas sobre estructuras conceptuales (conceptos, jerarquía de conceptos, relaciones, axiomas) en vez de la codificación de dichas estructuras directamente en ASCII. El modelo conceptual de una ontología es almacenado internamente utilizando un potente modelo ontológico, el cual puede ser extrapolado a diferentes lenguajes de representación.

- **KAON** [49]

KAON (The Karlsruhe Ontology and Semantic Web Tool Suite) construye un RDF y provee herramientas especializadas para el desarrollo, la ingenieria, el descubrimiento, la gerencia y la presentación de ontologías y meta-datos.

Un objetivo importante de KAON es el razonamiento escalable y eficiente de las ontologías. KAON proporciona 2 niveles de usuario: IO-Modeler y KAON PORTA, el resto de la suite, es para desarrollo.

OI-Modeler: Es el redactor de la ontología y proporciona la ayuda para la creación y mantenimiento de las ontologías. Las ontologías se pueden corregir simultáneamente por varios usuarios (soporta desarrollo concurrente). Otra característica de KAON es su ayuda para consultar las ontologías usando el lenguaje de interrogación de KAON. La navegación en una ontología se proporciona con metáforas basadas en gráficos y árboles.

OI-Modeler también proporciona una plataforma abierta y extensible para convivir con otros módulos de software para ontologías. Por lo tanto los usuarios pueden tener acceso a un registro de las ontologías de KAON y a los algoritmos vía OI-Modeler.

KAON PORTA: Proporciona un marco simple para navegar y buscar ontologías con browsers de la Web. Mientras que KAON porta se ha desarrollado sobre todo como un frame, espera ser adaptado a los propósitos individuales cambiando la disposición (y posiblemente un cierto código), es completamente funcional como una herramienta independiente.

- **OIL-ED** [50]

Es una herramienta para la edición de ontologías utilizando el lenguaje de especificación DAML+OIL. La idea inicial de OILEd fue la de proporcionar al usuario un editor muy simple para demostrar el uso del lenguaje OIL. Esta herramienta está en desarrollo, y todavía no permite la edición de ontologías de gran tamaño, así como operaciones de migración e interacción entre las mismas.

Permite que el usuario construya ontologías usando OIL. La intención detrás de esta herramienta es proporcionar un sistema de redaccción simple, freeware, el cual demuestre el uso y estimule el interés en OIL. OilEd no se piensa como ambiente completo de desarrollo de ontologías – no soportarán activamente el desarrollo de ontologías grandes, la migración y la integración de las ontologías, de versiones, argumentos y de muchas otras actividades que están implicados en la construcción de una ontología.

- **Plugins para Eclipse** [51]

Como se vio en el ítem 5.5.2 (Eclipse), este IDE puede convivir con programas adicionales que permiten la creación y/o manipulación de desarrollos de otras plataformas- En el caso de ontologías, existen una gama amplia de plugins capaces de desarrollar, modificar o actualizar ontologías dentro de eclipse.

Algunos de estos plugins son:

- Plugins Protégé 2000: Se puede obtener documentación en http://informatics.mayo.edu/LexGrid/index.php?page=protege
- DL-workbench ontological editor: http://projects.opencascade.org/dl-workbench/readme.html
- SWeDE: http://owl-eclipse.projects.semwebcentral.org/userguide.html
- OWL-S Editor: Esta herramienta puede ser instalada independiente o a través de eclipse: http://owlseditor.semwebcentral.org/download.shtml
- OWLS2PDDL
- VisioOWL
- CoBrA Demo Toolkit
- JSemWed

5.5.7. **Motores de inferencia** *[52]*

Algunos de los programas que suplen esta necesidad, son los siguientes:

- RACER [53]: Se integra nativo con Protégé y Eclipse. .
- CWM
- Euler
- SWISH
- MetaLog
- SwiProlog
- ConsVISor
- SWRL Validator

5.5.8. **Jena** [*54*]

Jena es un framework para la construcción de aplicaciones para la Web Semántica. Este provee un ambiente de programación para RDF, RDFS y OWL, incluyendo motores de inferencia basado en roles.

Jena es *open source*. El framework incluye:

- Un API para RDF
- Lectura y escritura de RDF en RDF/XML, N3 y N-Triples
- Un API OWL
- Almacenamiento en memoria y persistente
- RDQL – un lenguaje de consulta para RDF

Con JENA se pueden desarrollar programas java, capaces de leer, escribir o modificar una ontologías.

5.6. RECUPERACIÓN DE INFORMACIÓN (INFORMATION RETRIEVAL)

5.6.1. **Sistemas manejadores de contenidos (CMS)**

Los CMS (Content Management System) es un software que funciona para la creación y administración de contenido, actualmente se usa para denominar a los sistema que sirven para administrar el contenido de páginas Web. Permite asociar la información que se encuentra en artículos y que por el nombre del documento no es posible relacionarla con un tema o campo de conocimiento específico.

"Un CMS es una herramienta que provee facilidades de diseño, integración y adición de contenidos de manera automática a un sitio en la red (Internet, Extranet o Intranet). Su filosofía se basa en utilizar plantillas predeterminadas para los diseños gráficos, utilizar bases de datos para almacenar los contenidos y presentar una interfaz vía Web que permita interactuar con el CMS (tanto para administración como para la adición del contenido) sin la necesidad de estar presentes en el servidor o tener conocimientos de lenguajes de programación para visualización en browsers (html, php, xml, etc.)." [55]

Por tanto los CMS son herramientas que permiten publicar, diseñar y administrar contenidos en Internet sin necesitar la tarea de subir archivos vía FTP como hay que hacer con los editores web como Dreamweaver, FrontPage, etc.

5.6.2. **Gestión del Conocimiento**

La Gestión del Conocimiento busca agilizar y facilitar todos estos procesos. Supone un paso más en el tratamiento de la información, un salto cualitativo, que transforma la información aislada en conocimiento interrelacionado, eficaz.
The Gartner Group [56] define la Gestión del Conocimiento, como "la **identificación, gestión y difusión de la totalidad de los activos informativos de una empresa**, incluyendo las bases de datos, los documentos, las políticas y los procedimientos, al igual que el conocimiento experto y la experiencia poseída

por los trabajadores individuales (...). Persigue lograr el acceso rápido y cómodo de todo tipo de informaciones y enfatizar la comunicación interpersonal en lugar de la simple captura y almacenamiento de conocimiento".

5.6.3. **Conceptos básicos sobre la recuperación de la información.**

La recuperación de documentos tiene como objetivo satisfacer la necesidad de información de un usuario, normalmente expresada en lenguaje natural [57]. La recuperación de información tiene que ver con la representación, almacenamiento, organización y acceso a los ítems de información.

Es la ciencia de la búsqueda de información en documentos, búsqueda de los mismos documentos, la búsqueda de metadatos que describan documentos, o, también, la búsqueda en bases de datos, ya sea a través de Internet, intranet, para textos, imágenes, sonido o datos de otras características.

5.6.4. **Búsquedas clásicas de recuperación de información**

Algunos de los modelos para la recuperación de información, son los siguientes:

- *Búsqueda de contenidos Multimedia*: Búsqueda de archivos de vídeo, audio, imágenes u otros formatos cualesquiera a partir de la localización de expresiones que pudieran estar en campos de texto o en las proximidades del enlace al archivo.
- *Buscador difuso:* Localiza documentos o registros en bases de datos similares a la expresión de consulta.
- *Buscador Semántico*: Expande las consultas usando sinónimos de las palabras empleadas para expresar una búsqueda.
- *Buscador Multilingüe:* Permite expresar la búsqueda en un idioma y localizar todos los documentos relevantes en cualquier idioma.

- *Sistemas de Auto respuesta*: Tratan de localizar, no un documento, sino el párrafo concreto que responde a una consulta realizada por un usuario.

5.6.5. **Operaciones sobre los textos** [58].

Se puede definir a un lenguaje de consulta como un conjunto de órdenes, operadores y estructuras que, organizados según unas normas lógicas, permiten la consulta de fuentes y recursos de información electrónica

Los operadores son los encargados de expresar las relaciones que mantienen entre sí los términos que definen (más adecuado sería decir que pueden definir) las necesidades informativas del usuario. Pueden distinguirse diferentes tipos de operadores, que se analizan a continuación.

- *Operadores lógicos (o booleanos):* El principio que rige la utilización de este tipo de operadores es que las relaciones entre conceptos pueden expresarse como relaciones entre conjuntos. Las ecuaciones de búsqueda pueden transformarse en ecuaciones matemáticas, que ejecutan operaciones sobre los conjuntos, lo que da como resultado otro conjunto.
  Existen 3 operadores básicos: suma/unión (generalmente identificado como O/OR), el operador producto/intersección (identificado como Y/AND), y el operador resta/negación (identificado como NO/NOT).

- *Operadores posicionales:* Toman como punto de partida la consideración del valor del término dentro del contexto, es decir, que la posición de ese término en relación con otros, o dentro del propio registro, es significativa para valorar su pertinencia a los objetivos buscados. Estos pueden ser de 2 tipos:

Posicionales absolutos: Son aquellos que permiten buscar un término en un lugar dado del documento o registro.

Posicionales relativos: También llamados de proximidad, se trata de operadores que permiten establecer la posición de un término respecto a otro dado.

- *Operadores de comparación:* Especifican el rango de búsqueda, fijando unos límites para la misma. Estos límites pueden ser tanto numéricos como alfabéticos, correspondiendo los operadores a formas del tipo "mayor que", "menor o igual que".

- *Operadores de truncamiento:* Para facilitar la búsqueda de este tipo se han introducido operadores de truncamiento, a los que también se llama máscaras. Se trata de operadores (normalmente símbolos como *, $), cuya presencia puede sustituir a un carácter o a un conjunto de caracteres, situados a la izquierda, dentro o a la derecha del término en cuestión.

En los actuales sistemas de recuperación de información es posible encontrar todos estos tipos de operadores, que pueden combinarse entre sí, permitiendo crear ecuaciones complejas que reflejan con bastante precisión los conceptos y sus relaciones.

5.7. ELEMENTOS DEL MARCO TEÓRICO APLICADOS AL PROYECTO

Para la herramienta informática basada en ontologías para la clasificación automática de información de seguridad se tiene como insumo los reportes de vulnerabilidades de seguridad que son recibidas a través de correos electrónicos.

Una vez recibido y filtrado de acuerdo con la fuente de información de un correo (cualquiera que sea), se debe analizar el contenido de éste y verificar si está contenido dentro de la ontología (que es la referencia de clasificación y búsqueda), si el correo analizado cumple con las especificaciones o clasificaciones definidas por la ontología, este correo será clasificado como un reporte de una vulnerabilidad para herramientas software.

La relación y utilización de los elementos del marco teórico, se concentra en varios componentes, todos basados en métodos para la recuperación de la información.

Dentro de la inteligencia artificial están contenidos elementos como la recuperación de la información, el procesamiento del lenguaje natural, sistemas basados en reglas y la gestión del conocimiento, cada uno de estos elementos son retomados y aplicados en diferentes partes de la herramienta: En la Ontología (que se construye con la herramienta Protégé), tenemos la recuperación de la información que se hace a través de Jena, se aplican conceptos de los sistemas basados en reglas, que se hacen a través de los diferentes conectores lógicos de las propiedades (relaciones entre clases o individuos) dentro de la ontología, que son reglas de inferencia validados por el motor de inferencia *Racer* [53], capas de sacar conclusiones a partir de estas reglas.

De la misma forma UIMA como integrador de toda la solución, retoma técnicas para la recuperación de la información desde los archivos fuentes y realizar las respectivas comparaciones con la clasificación dada por la ontología.

Además, para toda la herramienta, tenemos el procesamiento de la información obtenido de fuentes en español (que es nuestro lenguaje natural).

Como se tiene la recuperación y comparación de información desde archivos textos, se realizan algunas operaciones con dichos textos, tal es el caso de la búsqueda taxonómica dentro de la ontología, la búsqueda será exitosa, cuando al menos se tengas 2 coincidencias desde la fuente con la clasificación definida en la ontología.

Por otro lado, algunos de los componentes de la Web semántica, tales como OWL (tipo de esquema escogido para representar la Ontología) y XML utilizado en los descriptores de UIMA, además, se hace la integración con Eclipse y Java, para crear las diferentes clases que interactúan con la Ontología y las fuentes de información.

Las diferentes vulnerabilidades reportadas por Sans Institute, son acopladas y configuradas como clases, sub-clases e individuos en la ontología y las fuentes de información son pre-filtradas a través de las macros en Outlook.

Por último y apoyado en la gestión de conocimiento, la herramienta para la clasificación de la información, nos permite transformar la información aislada en conocimiento interrelacionado y eficaz.

6. SOLUCIÓN

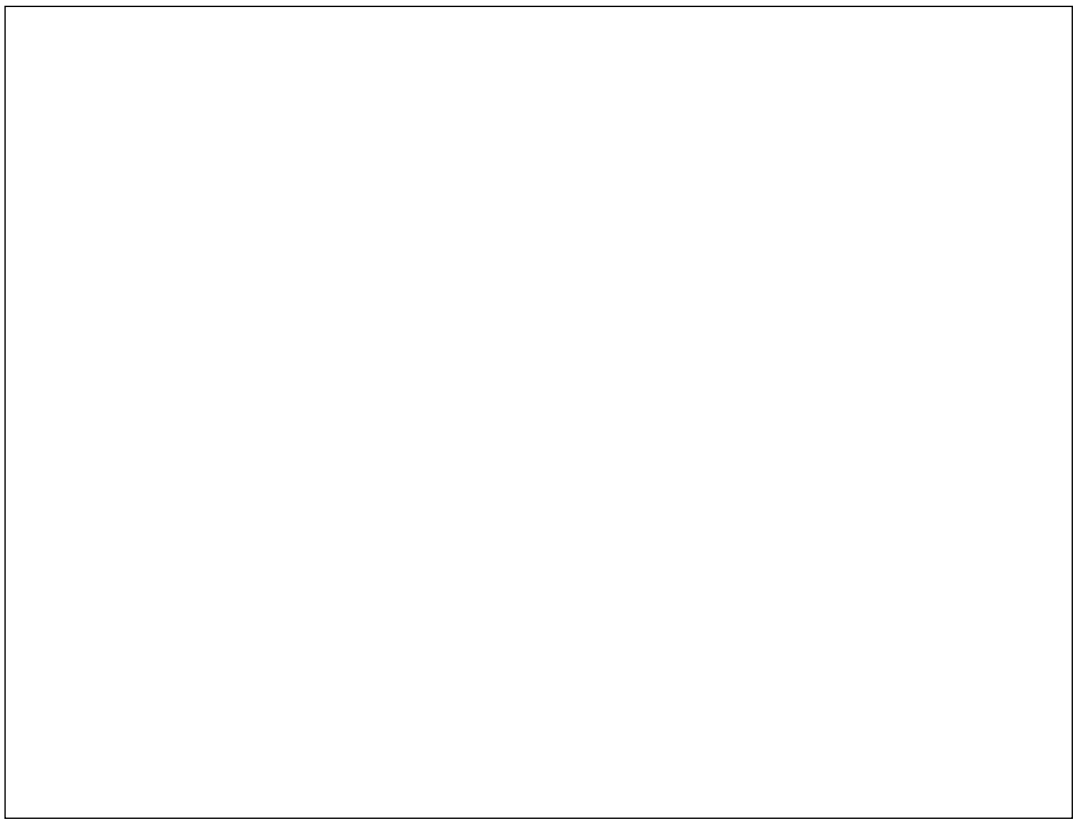
6.1. Análisis de la solución.

Se plantea la solución de división por partes, se requiere una pieza de software que lea, filtre y los guarde en disco duro los correos electrónicos.

Una vez guardados, otra pieza de software debe leer en esta ubicación los archivos guardados y procesar su contenido (se verifica palabra por palabra), por cada palabra que encuentre, invocará un programa que es capaz de abrir y leer una ontología y buscar en ésta, la palabra encontrada en el correo.

Si la palabra es encontrada en la ontología (de acuerdo con el criterio de búsqueda), el archivo será marcado y pasado a una carpeta de archivos procesados exitosamente, en caso contrario, en la carpeta no exitosos.

En el siguiente diagrama, se ilustra la arquitectura de la solución:



6.2. Requisitos de la solución.

Para dar solución el problema, es necesario las siguientes piezas de software:

- Outlook 2002 o 2003 y macros para este cliente.
- UIMA
- IDE Eclipse (Versión 3.1.0) con los siguientes plugins: Plugins UIMA, Jena, Protégé, EMF
- Java: j2sdk1.4.1_01.
- Equipo PC con las siguientes características:
    - Sistema operativo Windows 2000 o superior.
    - 256 Mb memoria mínimo (preferiblemente 512 Mb)
    - Procesador 800 Mhz o superior.
    - 1 Gb disco duro disponible para las instalaciones

- Capacidad en disco duro para el resguardo de los archivos de correos que llegarán.

- RACER: Versión 1.7.23 - Motor de Inferencia para la ontología.
- Protégé 3.0 con los siguientes plugins: ontoviz, owlviz, jambala.

6.3. Planteamiento de la solución.

6.3.1. **División del problema.**

El problema fue atacado desde varios frentes, apoyado en los objetivos específicos. Se realizaron las siguientes acciones, que continuación se describen:

6.3.2. **Clasificación y utilización de herramientas software para el desarrollo.**

Dentro de la búsqueda de herramientas para la construcción de la solución, se definieron las siguientes:

- **Outlook 2003**: Provee una gama amplia de macros, capaces de procesar en forma nativa los correos que son visualizados través del cliente Outlook (y su servidor exchange). Desde las macros, podemos referenciar cualquier tipo de acción sobre los correos (entrantes o salientes) dando una funcionalidad grande a esta herramienta colaborativa.

  Con esto, nos evitamos tener desarrollos externos al cliente de correo, que es la herramienta utilizada para la visualización.

- **Protégé:** Herramienta para la creación de Ontologías, se seleccionó esta herramienta por varias razones:
  - Software libre basado en Java
  - Provee edición de Ontologías a diferentes niveles.
  - Se integra nativo con RACER.

- Herramienta más dinámica para la construcción de ontologías.
- Permite la creación de ontologías OWL o RDF.
- Incluye una gama amplia de ejemplos, foros de discusiones y ayudas múltiples (Desde la herramienta, tutoriales, guías de usuario, shortcut y FAQ) [59].
- Permite el re-uso de otras ontologías.
- Permite importar ontologías en diferentes formatos (RDF, OWL, Pins, etc.)

- **UIMA:** Esta herramienta, creada por IBM, permite el procesamiento de información no estructurada y se ajusta a la necesidad de procesamiento de los diferentes correos. Tiene como valor agregado que lo que se haga en esa plataforma se puede integrar con cualquier otro producto basado en la misma. En los proyectos de la Universidad se está tratando de mantener UIMA como estándar para que los mecanismos de búsqueda y clasificación que se desarrollen se puedan utilizar sin mayores tropiezos.

- **Eclipse:** IDE para java, que a través de los plugins, podemos interactuar con UIMA, Protégé (en el caso que se requiera modificar las ontologías desde este IDE) y con RACER. A través del plugin JENA podemos interactuar con las diferentes ontologías desde un mismo programa en java.

6.3.3. **Clasificación de fuentes de información.**

Las organizaciones que proveen información en línea a través de listas de correo electrónico, nos dan la fuente más poderosa de información referente a las diferentes vulnerabilidades de herramientas software y basado en el informe oficial de SANS institute sobre vulnerabilidades de seguridad, inicialmente tendremos las siguientes fuentes (listas de correo):

- Hispasec Sistemas: E-mail unaaldia-admin@hispasec.com
- ACIS: E-mail segurinfo@acis.org.co , segurinfo-bounces@acis.org.co
- CERT: E-mail cert@cert.org

6.3.4. **Construcción de la ontología.**

Desde Protégé (previamente Instalado)[60] construimos la ontología, teniendo en cuenta los siguientes parámetros:

a. *Qué dominio de conocimiento se incluirá?*

Nuestro dominio de conocimiento son las vulnerabilidades de seguridad presentadas en herramientas software.

b. *Cuál es el contexto en que se construirá (para qué se va a usar)?*

La búsqueda y filtrado de contenidos sobre las vulnerabilidades en herramientas software, nos dará la posibilidad de actuar de manera eficiente y rápida ante un ataque eminente (o una epidemia de virus) hacia nuestra infraestructura informática. Con la gran incertidumbre que se maneja a nivel de información que puede ser relevante, es muy conveniente tener, de forma centralizada y filtrada, la información.

c. *Cuál es el alcance?*

El alcance inicial será para herramientas software, pero podría extenderse a cualquier dominio de conocimiento.
Además, el alcance estará dado también sobre la información que se tenga en idioma español.

d. *Cómo se obtiene el conocimiento sobre el dominio escogido?*

Se tiene la base de referencia las listas de seguridad publicadas por Sans Institute (para el periodo 2004-2005).

e. *Cómo identificar y seleccionar las fuentes potenciales de información?*

Algunas de las listas de correo más relevantes a nivel Colombia y en español.

f. *Cuál es el público objetivo?*

Todos aquellos administradores, funcionarios de soporte técnico o mesa de ayuda, que dan soporte y/o mantenimiento a la infraestructura informática de las organizaciones. También a los desarrolladores en las diferentes tecnologías.

g. *Cómo se mantendrá actualizada la ontología*?

Se pretende dar una actualización cada trimestre, dado que Sans Institute se toma este tiempo para generar sus actualizaciones.

Una vez delimitado el alcance de la ontología, es necesario describir las clases, sub-clases, propiedades, relaciones o restricciones e individuos dentro de la Ontología.
Para nuestro caso tenemos:

***Clases**:* Se han definido como clases principales:

- Sistemas Operativos
- Vulnerabilidades

Sub-clases: La siguientes son las sub-clases contenidas dentro de las clases principales, además, se relacionan sub-clases dentro de las sub-clases (Esta clasificación esta basada de cuerdo a la información obtenida de Sans Institute)

- Sistemas Operativos
    - Sistemas_Microsoft: Clase de todas las versiones de sistemas operativos de la empresa Microsoft[61].
    - Sistemas_Unix: Clase que representa los sistemas operativos basados en Unix.
- Vulnerabilidades
    - Cliente_Correo: Clase que contiene los individuos que representan las piezas de software tal que son vulnerabilidades ocasionadas en los clientes del Correo electrónico.
    - En_Autenticacion: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades dadas en la autenticación de procesos/usuarios.
    - En_Bases_De_Datos: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades en bases de datos.
    - En_Comparticion_Archivos: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades en aplicaciones para la compartición de archivos.
    - En_Control_Versiones: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades en software para el control de versiones (CVS)

- En_Kernel: Clase que contiene las vulnerabilidades dadas en el Kernel.
- En_Mensajeria_Instantanea: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades dadas en aplicaciones de mensajería instantanea.
- En_Protocolos: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades dadas en protocolos de comunicación.
- En_Servicio_Transporte_Mail: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades dadas en los servicios para el transporte de correos electrónicos.
- En_Servicios: Clase que contiene las vulnerabilidades dadas en diferentes servicios.
- En_Servidor_Correo: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades dadas en los servidores de correo.
- Web_Browser: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades dadas en las interfaces para navegación.
- Web_Server_Services: Clase que contiene los individuos que representan las piezas de software que son vulnerabilidades dadas en los servicios de los servidores Web.

Fue necesario definir 2 super-clases conteniendo varias de estas sub-clases, están son:

- **En_Mail**: Clase que contiene las vulnerabilidades dadas en el correo electrónico, contendría las sub-clases: Cliente_Correo, En_Servicio_Transporte_Mail y En_Servidor_Correo
- **En_Web:** Clase que contiene las vulnerabilidades orientadas a la Web, y Contendría las Sub-clases: Web_Browser y Web_Server_Services

El árbol de clases (desde protégé) es:

Propiedades:

Las propiedades definidas son:

- *Poseen*: Propiedad que indica la conexión o relación que existe entre los sistemas Operativos y las Vulnerabilidades.
- *ES_DE_UN:* Propiedad que describe la relación entre las vulnerabilidades y los sistemas operativos. Las vulnerabilidades ES_DE_UN tipo de S.O
- *Tiene_S.O-U*: Propiedad que relaciona el tipo de vulnerabilidad presentada en los sistemas Operativos Unix-Linux.
- *Tiene_S.O-W*: Propiedad que relaciona el tipo de vulnerabilidad presentada en los sistemas Operativos Microsoft.

Dominio y rango: A las restricciones les podemos delimitar su campo de acción, para ello podemos utilizar el dominio que es una clase al que pertenece y el rango donde se moverán (clases de acción.). Para nuestro proyecto de grado, se definió lo siguiente:

Restricción	**Dominio (Clase)**	**Rango (Clase)**
Posee	Vulnerabilidades	Sistema_Operativo
ES_DE_UN	Sistema_Operativo	Vulnerabilidades
Tiene_S.O-U	Sistemas_unix	
Tiene_S.O-W	Sistemas_Microsoft	

***Individuos**:* Se han definido por cada clase una serie de individuos propios de la misma.

Es importante también considerar, que los individuos pueden tener similitud con otros (sinónimos) podemos describir esto a través de la opción en Protégé ***SameAs*** con el objeto de describir otra forma del mismo individuo. Algunos de estos sinónimos que se definieron para individuos son:

Individuo	Sinónimo	Individuo	Sinómino (s)
Linux	Unix	DBMS	MSSQL,

			PostgreSQL, Oracle, MySQL
P2P	Peer-to-Peer	Courier-MTA	MTA
MTA	SendMail, Postfix, Qmail	AIM	AOL
Messenger	MSN	IE	Internet_Explorer
iPlane	SunOne		

Por cada sub-clase tenemos:

- Sistemas_Operativos
  - Sistemas_Microsoft: Windows
  - Sistemas_Unix: Unix y Linux
- Vulnerabilidades
  - Cliente_Correo: Outlook y OutlookExpress
  - En_Autenticacion: Autenticación y Contraseñas
  - En_Bases_De_Datos: DBMS, MSSQL, MySQL, Oracle, PostgreSQL y SQL
  - En_Comparticion_Archivos: Aplicaciones para compartir archivos a traves de las tecnologías peer-to-peer: E-Donkey, GNUTella,Kazaa, Napter, P2P, Peer-to-Peer
  - En_Control_Versiones: CVS (Concurrent Versions System)
  - En_Kernel: Kernel
  - En_Mensajeria_Instantanea: AIM,AOL,Messeger,MSN,Yahoo
  - En_Protocolos: http, SMTP, SNMP, SSL
  - En_Servicio_Transporte_Mail: Courier-MTA, MTA, PostFix, Qmail, SendMail.
  - En_Servicios: BIND, LSAS, NETBIOS, NFS, NIS, RAS, RPC
  - En_Servidor_Correo: DoS, Exchange

- Web_Browser: Firefox,Internet_Explorer, Mozilla, Netscape, Opera.
- Web_Server_Services: Apache, IIS, iPlanet, SunOne

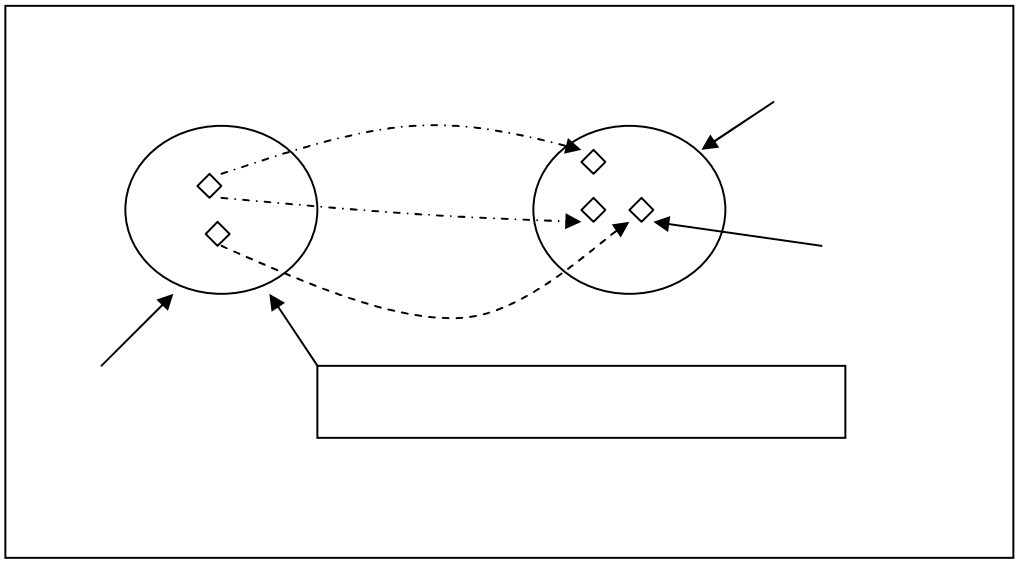
Las clases pueden tener varias características, entre ellas, la disyunción, que es la relación que se establece entre dos o más elementos, uno de los cuales excluye a los demás. Para las siguientes clases podemos tener esta característica:

Cliente_Correo *es Disjunta de* Servidor_Correo
En_Protocolos *es Disjunta de* En_Servicios
Web_Browsers *es Disjunta de* Web_Server_Services

Restricciones: Existen 3 tipos de restricciones:

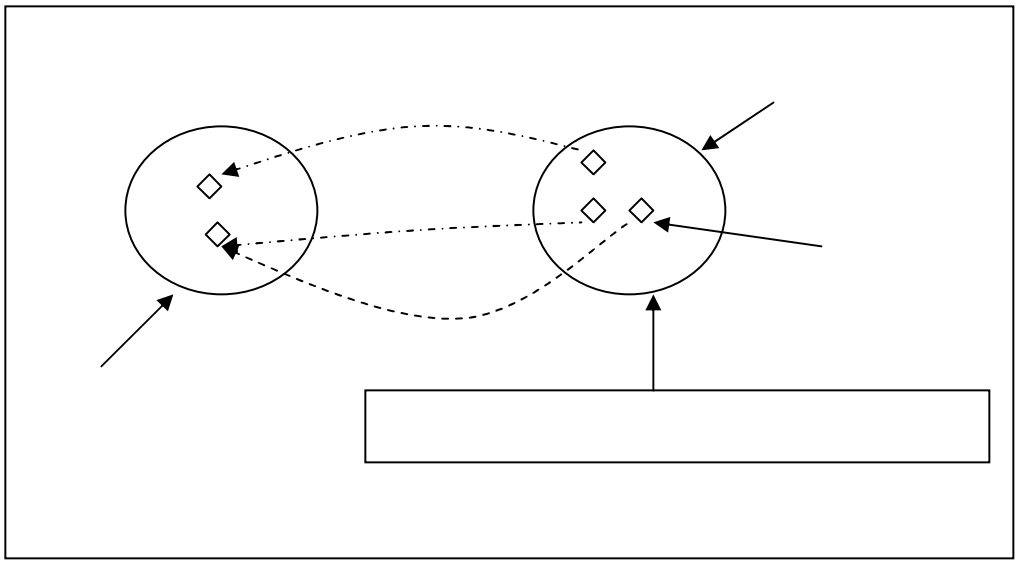
- Existencial (∃): Al menos uno (1) o más.
- Universal (∀): Para todos.
- Existencial (∋): Utilizado para los individuos y denota *al menos una relación*

Como ejemplo, tenemos:



La restricción: (∀ Poseen Vulnerabilidades) describe: Todos los individuos (de la clase - Sistema_Operativo) poseen Vulnerabilidades.

Otro Ejemplo es:



La restricción (∃ Es_de_un Sistema_Operativo) describe:
Los individuos de la clase *Vulnerabilidades* son de *al menos un* sistema Operativo.

Para la creación de las restricciones a nivel de clases o individuos, es necesario tener en cuenta algunos operadores lógicos y de agrupación:

- ⊓ : Operador And, este operador describe el cumplimiento de todas expresiones involucradas.
- ⊔: Operador Or - este operador describe el cumplimiento de “al menos” una expresión involucrada.
- (): paréntesis – Delimita una expresión dada.

Las restricciones definidas para la ontología son:

Clase	Sub-Clase	Restricción	Descripción
Sistema_Operativo		∃ Poseen Vulnerabilidades	Los sistemas operativos poseen al menos una vulnerabilidad.
Sistema_Operativo	Sistemas_Microsoft	Tiene_S.O-W ∋ Windows	El individuo Windows, tiene al menos una relación con la propiedad Tiene_S.O-W
Sistema_Operativo	Sistemas_Unix	(Tiene_S.O-U ∋ Unix) ⊓ (Tiene_S.O-U ∋ Linux)	Los individuos Unix y Linux, tiene al menos una relación con la propiedad Tiene_S.O-U
Vulnerabilidades		(Tiene_S.O-U ∋ Vulnerabilidades_Software) ⊔ (Tiene_S.O-W ∋ Vulnerabilidades_Software)	El individuo: Vulnerabilidades_Software, tiene al menos una relación con las propiedades Tiene_S.O-U o Tiene_S.O-W
Vulnerabilidades	Cliente_Correo	(Tiene_S.O-W ∋ Outlook) ⊔ (Tiene_S.O-W ∋ OutlookExpress)	Los individuos Outlook y OutlookExpress tiene al menos una relación con la propiedad Tiene_S.O-W
Vulnerabilidades	Cliente_Correo	Tiene_S.O-W ∋ Windows	El individuo Windows, tiene al menos una relación con la propiedad Tiene_S.O-W
Vulnerabilidades	En_Autenticacion	(Tiene_S.O-U ∋ Autenticacion) ⊔ (Tiene_S.O-U ∋ contraseñas)	Los individuos Autenticación o contraseñas, tiene al menos una relación con la propiedad Tiene_S.O-U
	En_Autenticacion	(Tiene_S.O-W ∋ Autenticacion) ⊔ (Tiene_S.O-W ∋ contraseñas)	Los individuos Autenticación o contraseñas, tiene al menos una relación con la propiedad Tiene_S.O-W
	En_Autenticacion	(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades por autenticación tiene al menos un sistema operativo, Microsoft o Unix
Vulnerabilidades	En_Bases_De_Datos	(Tiene_S.O-U ∋ DBMS) ⊔ (Tiene_S.O-U ∋ SQL)	Los individuos DBMS y SQL tiene al menos una relación con la propiedad Tiene_S.O-U
	En_Bases_De_Datos	(Tiene_S.O-W ∋ DBMS) ⊔ (Tiene_S.O-W ∋ SQL)	Los individuos DBMS y SQL tiene al menos una relación con la propiedad Tiene_S.O-W
	En_Bases_De_Datos	(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en Bases de datos tiene al menos un sistema operativo, Microsoft o Unix
Vulnerabilidades	En_Comparticion_Archivos	(Tiene_S.O-U ∋ Peer-To-Peer) ⊔ (Tiene_S.O-U ∋ GNUTella)	Los individuos Peer-to-Peer y GNUTella tiene al menos una relación con la propiedad Tiene_S.O-U
		(Tiene_S.O-W ∋ Kazaa) ⊔ (Tiene_S.O-W ∋ E-Donkey) ⊔	Los individuos Kazaa,E-Donkey y Peer-to-

		(Tiene_S.O-W ∋ Peer-To-Peer)	peer tiene al menos una relación con la propiedad Tiene_S.O-W
		(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en aplicaciones para compartir archivos tiene al menos un sistema operativo, Microsoft o Unix
Vulnerabilidades	En_Control_Versiones	Tiene_S.O-U ∋ CVS	El individuo CVS tiene al menos una relación con la propiedad Tiene_S.O-U
		∃ Tiene_S.O-U Sistemas_Unix	Las vulnerabilidades en Control de versiones, tiene al menos un sistema operativo Unix
Vulnerabilidades	En_Kernel	Tiene_S.O-U ∋ Kernel	El individuo Kernel tiene al menos una relación con la propiedad Tiene_S.O-U
		∃ Tiene_S.O-U Sistemas_Unix	Las vulnerabilidades en Kernel, tiene al menos un sistema operativo Unix
Vulnerabilidades	En_Mensajeria_Instantanea	(Tiene_S.O-W ∋ Messenger) ⊔ (Tiene_S.O-W ∋ AIM) ⊔ (Tiene_S.O-W ∋ AOL)	Los individuos Messenger, AIM y AOL tiene al menos una relación con la propiedad Tiene_S.O-W
		∃ Tiene_S.O-W Sistemas_Microsoft	Las vulnerabilidades en la mensajeria instantánea, tiene al menos un sistema operativo Windows
Vulnerabilidades	En_Protocolos	(Tiene_S.O-U ∋ HTTP) ⊔ (Tiene_S.O-U ∋ SMTP) ⊔ (Tiene_S.O-U ∋ SNMP)	Los individuos HTTP, SMTP y SNMP tiene al menos una relación con la propiedad Tiene_S.O-U
		(Tiene_S.O-W ∋ HTTP) ⊔ (Tiene_S.O-W ∋ SMTP) ⊔ (Tiene_S.O-W ∋ SNMP)	Los individuos HTTP, SMTP y SNMP tiene al menos una relación con la propiedad Tiene_S.O-U
		(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en protocolos tienen al menos un sistema operativo, Microsoft o Unix
Vulnerabilidades	En_Servicio_Transporte_Mail	(Tiene_S.O-U ∋ MTA) ⊔ (Tiene_S.O-U ∋ Sendmail)	Los individuos MTA y SendMail tiene al menos una relación con la propiedad Tiene_S.O-U
		(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en servicios de transporte de Mail tienen al menos un sistema operativo, Microsoft o Unix
Vulnerabilidades	En_Servicios	(Tiene_S.O-U ∋ NETBIOS) ⊔ (Tiene_S.O-W ∋ RPC)	Los individuos NETBIOS y RPC tiene al menos una relación con la propiedad Tiene_S.O-U y Tiene_S.O-W

			respectivamente.
		(Tiene_S.O-W ∋ BIND) ⊔ (Tiene_S.O-W ∋ RAS) ⊔ (Tiene_S.O-W ∋ NFS)	Los individuos BIND,RAS y NFS tiene al menos una relación con la propiedad Tiene_S.O-W
		(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en servicios (general) tienen al menos un sistema operativo, Microsoft o Unix
Vulnerabilidades	En_Servidor_Correo	(Tiene_S.O-W ∋ Exchange) ⊔ (Tiene_S.O-W ∋ DoS)	Los individuos Exchange y DoS tiene al menos una relación con la propiedad Tiene_S.O-W
		(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en el servidor de correo tienen al menos un sistema operativo, Microsoft o Unix.
Vulnerabilidades	Web_Browser	(Tiene_S.O-U ∋ FireFox) ⊔ (Tiene_S.O-U ∋ Netscape) ⊔ (Tiene_S.O-U ∋ Mozilla)	Los individuos Firefox, netscape y Mozilla tiene al menos una relación con la propiedad Tiene_S.O-U
		(Tiene_S.O-W ∋ FireFox) ⊔ (Tiene_S.O-W ∋ Netscape) ⊔ (Tiene_S.O-W ∋ Mozilla)	Los individuos Firefox, netscape y Mozilla tiene al menos una relación con la propiedad Tiene_S.O-W
		(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en Browser para Web tienen al menos un sistema operativo, Microsoft o Unix.
Vulnerabilidades	Web_Server_Services	(Tiene_S.O-U ∋ Apache) ⊔ (Tiene_S.O-U ∋ SunOne)	Los individuos Apache o SunOne tienen al menos una relación con la propiedad Tiene_S.O-U
		(Tiene_S.O-W ∋ Apache) ⊔ (Tiene_S.O-W ∋ IIS)	Los individuos Apache o IIS tienen al menos una relación con la propiedad Tiene_S.O-W
		(∃ Tiene_S.O-W Sistemas_Microsoft) ⊔ (∃ Tiene_S.O-U Sistemas_Unix)	Las vulnerabilidades en los servicios de servidores Web tienen al menos un sistema operativo, Microsoft o Unix.

Lista de restricciones para clases e Individuos.

Una vez creada y configurada cada clase, propiedad, individuo y sus restricciones, debemos verificar si la ontología quedo bien construida, para ello nos apoyamos en el motor de inferencia RACER.

Racer puede ser configurado desde Protégé, configurando el servidor y puerto por el cual se ejecutará desde el aplicativo Protége (Barra de herramientas) opción:

- *OWL*
- *Preferences*
- En la opción *Reasoner URL,* ingresar http://localhost:8080
- En la opción *Languaje Profile* seleccionar *OWL Lite*, dado que es mas ligero y rápido y puede evitar mensajes no soportados por el razonador.
- Luego *Close*

Luego de esto, ejecutar ***RACER.EXE*** que por defecto, correo sobre el puerto 8080[62], una vez abra la interfaz, desde Protégé tenemos las opciones *Check consistency y Classify taxonomy* (por el menú OWL), con estas opciones podemos verificar la construcción de la Ontología y su clasificación taxonómica.

Una vez obtenido el resultado, se abre una pestaña *SUBCLASS RELATIONSHIP*, donde podemos visualizar la relación de clases y a través del plugin *OWL viz tab* podemos visualizar la respectiva clasificación taxonómica que se genero, verificando que la construcción de la ontología si esta acorde a lo estipulado:

Relación de Clases:

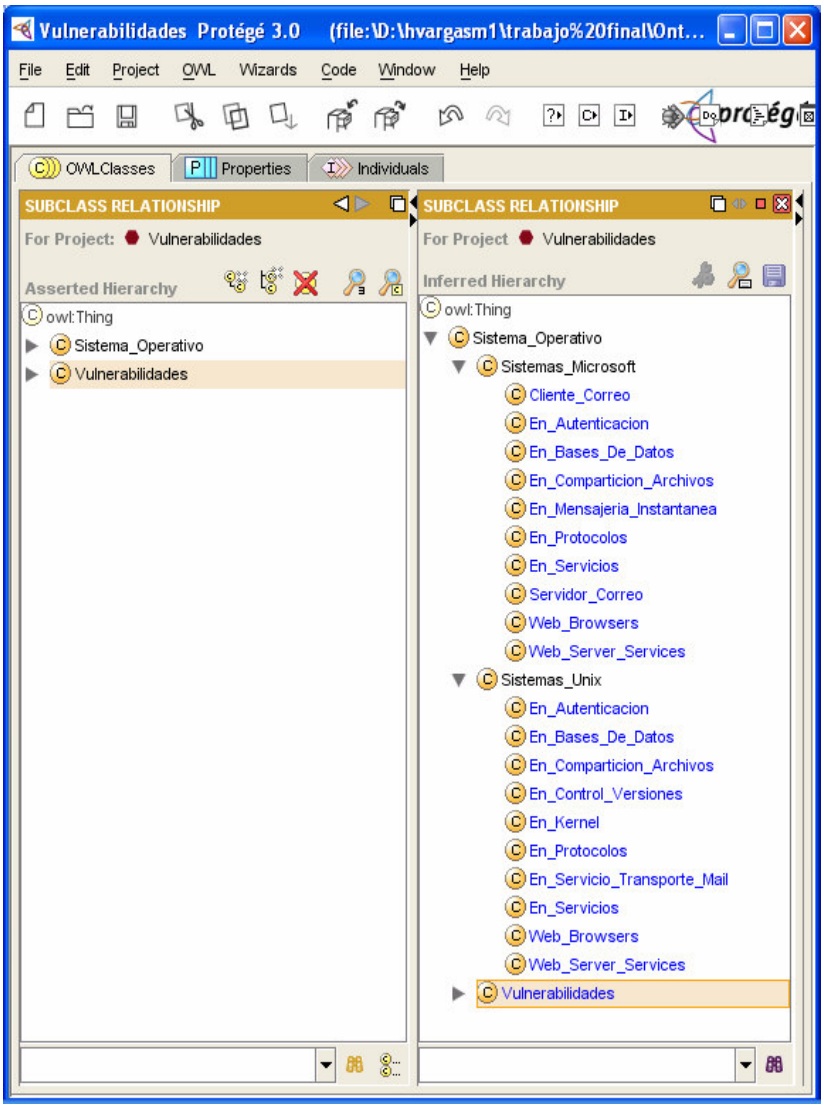


Figura 8: Relación entre las clases una vez ejecutado Racer.

Como se puede observar, de acuerdo con las restricciones, disyunciones y dominios asignados, las clases de clasificarán de acuerdo a su estructura y al sistema operativo que le corresponde.

Así, por ejemplo, de acuerdo con la restricción, las vulnerabilidades en Kernel sólo se presentan en los sistemas operativos UNIX (por tanto, no debe aparecer en los Windows).

Por otro lado, la visualización del árbol taxonómico es (con OWLviz):

Figura 9: Organización taxonómica de la Ontología.

6.3.5. **Construcción de descriptores y motores en UIMA.**

Para comenzar a trabajar con UIMA, es necesario que ECLIPSE (Version 3.1.0) y UIMA estén instalados (con los respectivo plugins dentro de Eclipse: EMF y UIMA)[63].

Una vez instalados, abrimos eclipse desde su ubicación. Se debe definir una carpeta de trabajo o *workspace* en el disco duro, en dicha carpeta de guardarán todos los proyectos (con descriptores, archivos fuentes, XML, etc.). Se ha creado nuestro workspace en la ruta *D:\Eclise\Workspace*, una vez se inicie Eclipse, éste solicitará esta ruta.

Paso seguido, debemos crear un proyecto java (*File>new>Project>Java proyect*) se solicitará el nombre del proyecto (para nuestro caso: Ontologias) en el cual, tendremos todo nuestro repositorio.

Debemos decirle al proyecto, que será una estructura UIMA, para ello, damos clic derecho al proyecto y seleccionamos *Add UIMA Nature*, esto nos creara la estructura jerárquica que UIMA utiliza para sus procesos.

En Eclipse, debemos trabajar bajo una perspectiva Java, con ello todo el proyecto estará regido bajo la maquina virtual y sus librería. Para hacer esto, es necesario desde Eclipse dar las opciones:

Window>Open Perspective>Java

Ahora bien, para la creación del programa se consideran los siguientes pasos:

a. Definir el tipo CAS (Common Analysis Structure) que el Annotator utilizará.

Estos tipos de estructuras pueden ser Enteros, Float, String, IntegerArray, FloarArray, StringArray, FSArray y Annotator. Este primer paso define un archivo XML llamado *Type System Descriptor.*

UIMA incluye un plugins especial para ayudar a editar el *Type System Descriptor* [64], con ello se usa Eclipse como interfaz para la edición de todos los descriptores y códigos java.

Parados en la carpeta *Desc*, damos clic derecho *New>Other>Type System Descriptor File* luego la opción *Next* para incluir el Nombre y la ubicación donde quedará. Para nuestro caso, este descriptor se llama ***typeSystemDescriptor.xml***

Una vez creado, damos clic derecho al descriptor y la opción *Open With>Component descriptor Editor*

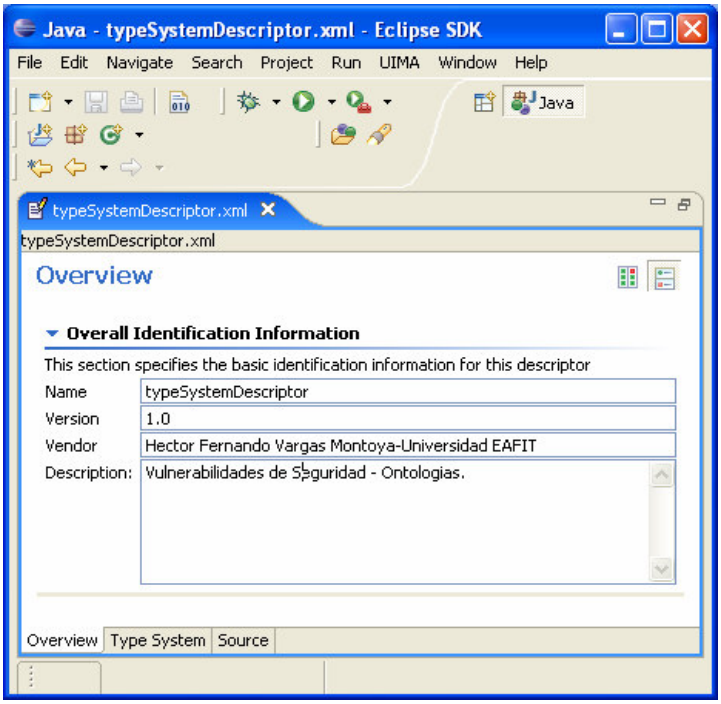


Figura 10: Creación de un descriptor tipo CAS en Eclipse.

En la pestaña *Type System* incluimos el tipo o los tipos de datos que queremos procesar, para nuestro proyecto necesitamos 1 clases, para leer los archivos de entrada y procesarlos (de acuerdo a la clase java que se cree.)

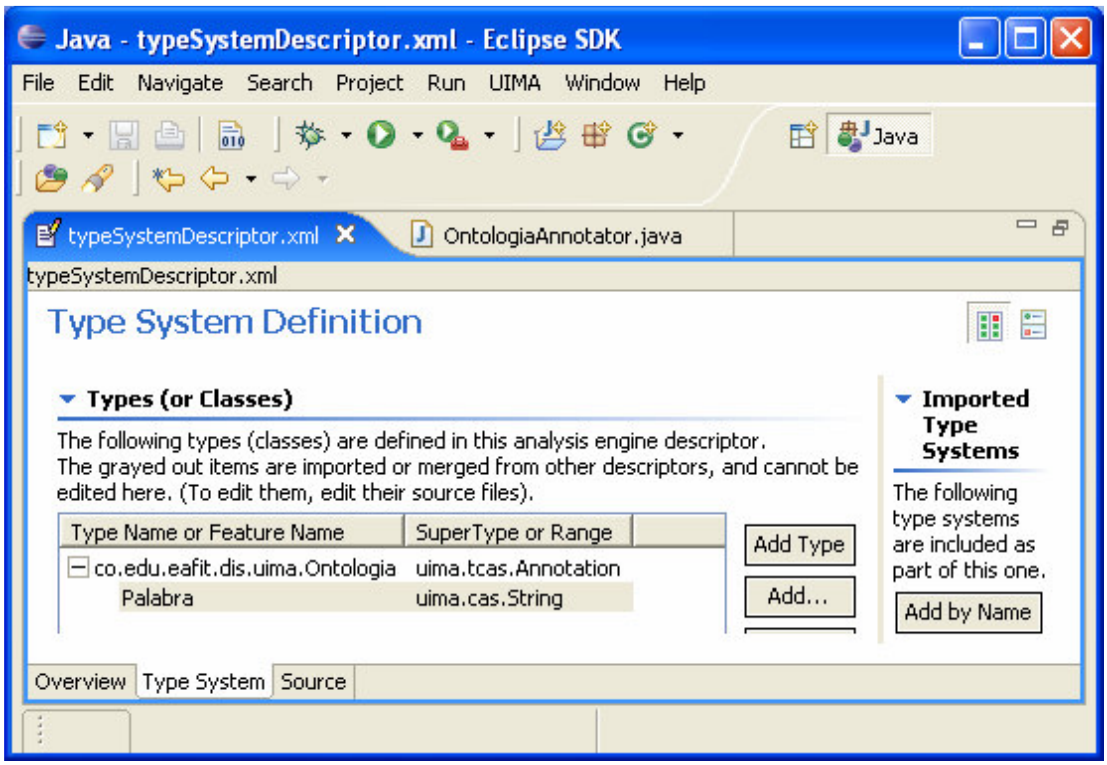


Figura 11: Parametrización del descriptor tipo CAS

La clase es *co.edu.eafit.dis.uima.Ontologia* con el tipo *Palabra (String)*, este String será la variable que requiere el programa java para guardar cada palabra que leerá del (los) archivo (s) de entrada y será pasada al programa que busca dentro de la Ontología.

b. Crear las clases Java para los tipos CAS creados en el punto anterior.

Una vez configurada esta definición, verificamos que la utilidad *JCasGen* de UIMA esté activa. Esta utilidad nos permite crear automáticamente las clases

Java que requiere el descriptor, si no está activa, nos vamos por opciones del menú *UIMA>Setting>Auto generate JCAS y Display Full.*

Ahora ya podemos salvar el archivo, se debieron haber creado en la carpeta ***scr*** un paquete (*co.edu.eafit.dis.uima*) con estas clases: ***Ontologia.java*** y ***Ontologia_Type.java*** (se debe verificar que las clases se hayan creado).

Si la utilidad no está configurada dentro de Eclipse, debemos ejecutar el archivo por lotes ***jcasgen,bat*** ubicado en el directorio ***bin*** donde fue instalado UIMA.

c. Escribir el código Java para el Annotator.

Este código Java nos permitirá leer y mover los diferentes archivos de entrada a procesar, un Annotator tiene varios métodos, entre los más comunes están:

- Initialize
- Process y
- destroy

Nuestra clase Annotator será un extend de la clase *JTextAnnotator_ImplBase*, que es una interfase para el uso de JCas.

Dando clic derecho sobre el paquete *co.edu.eafit.dis.uima >New>Class* y damos la descripción respectiva a nuestra clase:

- Nombre: OntologiaAnnotator
- SuperClase: com.ibm.uima.analysis_engine.annotator.JTextAnnotator_ImplBase

Luego damos *Finish.*

Como pre-requisito para la creación de esta clase, es conveniente crear en el disco duro, las carpetas de donde se obtendrán y moverán los archivos. En el disco C:\ se deben crear las siguientes carpetas:

C:\Ontologia\Input - En esta carpeta quedarán los archivos de entrada a procesar.

C:\Ontologia\Ontologia - Reside la ontología donde se hará la búsqueda.

C:\Ontologia\Procesado-NO-OK – Carpeta a donde se moverán los archivos procesados y que su contenido **NO** está dentro de la Ontología.

C:\Ontologia\Procesado-OK - Carpeta a donde se moverán los archivos procesados y que su contenido (2 ó mas palabras) **SI** fue encontrado dentro de la Ontología y que fueron clasificados como válidos.

Luego de crear estas carpetas, creamos la clase java, que contiene 3 partes:

1) **Inicialización:** Se inicializan las diferentes variables y se crea una lista de los archivos de entrada (en las carpetas definidas como pre-condición)
2) **Búsqueda:** Obtener dentro de cada archivo palabra por palabra, luego le pasa lo leído a la clase ***OntBuscar.java***[65] quien buscará en la Ontología si esta palabra corresponde a un Individuo o clase.
3) **Clasificación:** Una vez procesado el archivo (y por ende todas sus palabras) serán movidos a la carpeta respectiva.

Esta clase está configurada para leer periódicamente la entrada en busca de archivos nuevos (que serán creados desde el correo electrónico) y estará en espera media hora, hasta verificar nuevamente.
A continuación se da un descripción de cada una de las partes de componen esta clase:

1) **Inicialización:**

Hace referencia a las variables necesarias para la verificación de los archivos de entrada, se hace referencia a las carpetas donde residirán o a donde se moverán una vez procesados los archivos, el siguiente bloque de código ilustra la forma como se parametrizan las entradas y las salidas:

```
File dir = new File("c:\\ontologia\\Input");
File destinoOK = new File ("c:\\ontologia\\Procesado-OK");
File destinoNOOK = new File ("c:\\ontologia\\Procesado-NO-OK");
String[] Archivos;
String nombre = null;
```

Luego se busca dentro de la carpeta de entrada, si hay archivos (y cuántos):

```
while(true){
Archivos = null;
Archivos = dir.list();
int max = Archivos.length;
```

Una vez obtenida la lista, se envía cada nombre de archivo a la función ***Archivo*** esperando que sea retornado un exitoso o no.

2) **Búsqueda:**

Cuando la función ***Archivo*** recibe el parámetro del nombre del archivo encontrado, abre el archivo y lee palabras cuya longitud sea mayor o iguales a 3, esto debido a que en la Ontología, la longitud del menor nombre es de 3, con ello, nos ahorramos búsquedas inoficiosas de preposiciones o conjunciones.
En el siguiente bloque se inicializan los buffer y leemos el archivo encontrado:

```
FileReader entrada=null;
```

```
StringBuffer str=new StringBuffer();
String palabra = null;
.
.
entrada= new FileReader(nombre);
while((c=entrada.read())!=-1){
Archivo = str.append((char)c).toString();}
```

Luego abrimos el archive, lo leemos secuencialmente sepado por los espacios:

```
String[] docText = Archivo.split("\\W",-1);
```

Capturando así, cada palabra del archivo. Por cada palabra, invocamos la búsqueda en la ontología, haciendo el llamado a la clase ***OntBuscar,*** esta clase es la encargada de abrir la Ontología y buscar la palabra designada.

Para realizar operaciones sobre Ontologías, se hace uso de ***Jena***[66] y sus librerías, existen dos formas de acceder a la ontología: Por sentencias SQL y a través de la librería *iterator.ExtendedIterator.* Para este proyecto, se utilizó este ultimo. De la misma manera, se inicializan variables de entrada que permiten la ubicación de la Ontología:

```
FileWriter salida=null
String owl = null;
String busqueda = "#palabra”;
.
OntModel model = ModelFactory.createOntologyModel();
 model.read("file:///C:\\Ontologia\\Ontologia\\Vulnerabilidades.owl");
```

Así, en el objeto ***model*** hemos cargado la ontología y podrá ser listada (todos sus miembros: Clases, propiedades e individuos) con la sentencia:

```
ExtendedIterator iterClass = model.listSubjects();
```

Esto se almacena en un vector y se busca la palabra que requerimos, si la búsqueda es exitosa, retornamos la palabra, de lo contrario un "**no**".

3) Clasificación:

Una vez retornado el valor desde la búsqueda de la ontología, procedemos a mover el archivo a la carpeta respectiva:

```
if (c1 == "OK"){
Ontologia annotation = new Ontologia(aJCas);
annotation.addToIndexes();
File file = new File(nombre);
file.renameTo(new File(destinoOK, Archivos[x]));
} else {
File file = new File(nombre);
 file.renameTo(new File(destinoNOOK, Archivos[x])); }
```

Y esperamos el tiempo respectivo: Thread.sleep(1000*1800)

Para más exactitud, ver el código fuente en los anexos.

d. Crear un descriptor XML.

La arquitectura de UIMA requiere de información descriptiva acerca del Annotator representado en un archivo XML, este archivo es llamado *Analysis Engine Descriptor*. Este descriptor incluye:

- Nombre, descripción, versión y autor.
- Entrada y salida para el Annotator definidos en términos de tipos.
- Declaración de los parámetros de configuración que acepta el Annotator.

Para crear este motor de análisis, damos clic derecho en la carpeta ***Desc*** *>New>other>UIMA>Analysis engine Descriptor File* luego NEXT, damos el Nombre (para el proyecto es: OntologiaDescriptor) y OK.

Una vez creado, damos clic derecho sobre este > *Open with>Component Descriptor Editor* y parametrizamos en la pestaña *Overview:*

- Nombre clase Java: co.edu.eafit.dis.uima.OntologiaAnnotator , es el nombre de la clase construida en el ítem anterior.
- Name: OntologiaDescriptor
- Version: 1.0
- Vendor: Hector Fernando Vargas Montoya-Universidad EAFIT
- Description: Vulnerabilidades de Seguridad - Ontologias.

Y nos vamos a la pestaña *Type System* y configuramos:

- Type Name:
  - Add Type: co.edu.eafit.dis.uima.Ontologia
  - Super Type: uima.tcas.Annotation
- En Add..
  - Feature Name: Palabra
  - Range Type: uima.cas.String
  - Feature Name: Nombre

- Range Type: uima.cas.String

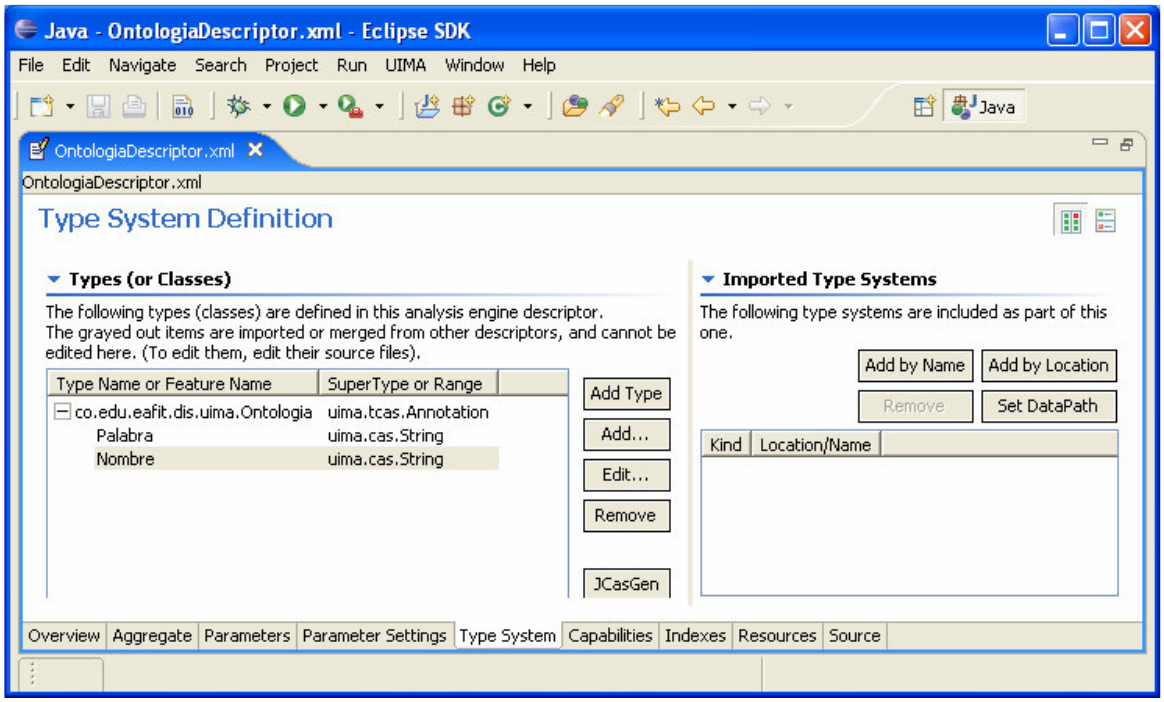


Figura 12: Parametrizaciòn de un *Analysis Engine Descriptor*

Ahora en la pestaña *Capabilities* configuramos las salidas seleccionándolas todas.

e. Prueba del annotator.

Teniendo desarrollado el Annotator, UIMA incluye una herramienta llamada ***Document Analyzer,*** la cual nos permite subir toda la interfase, se debe ejecutar el archivo *DocumentAnalyzer.bat* ubicado en la carpeta *Bin* de la instalaciòn de UIMA.

Debemos tener, como mínimo, un archivo en la carpeta de entrada, de lo contrario, el programa terminará.

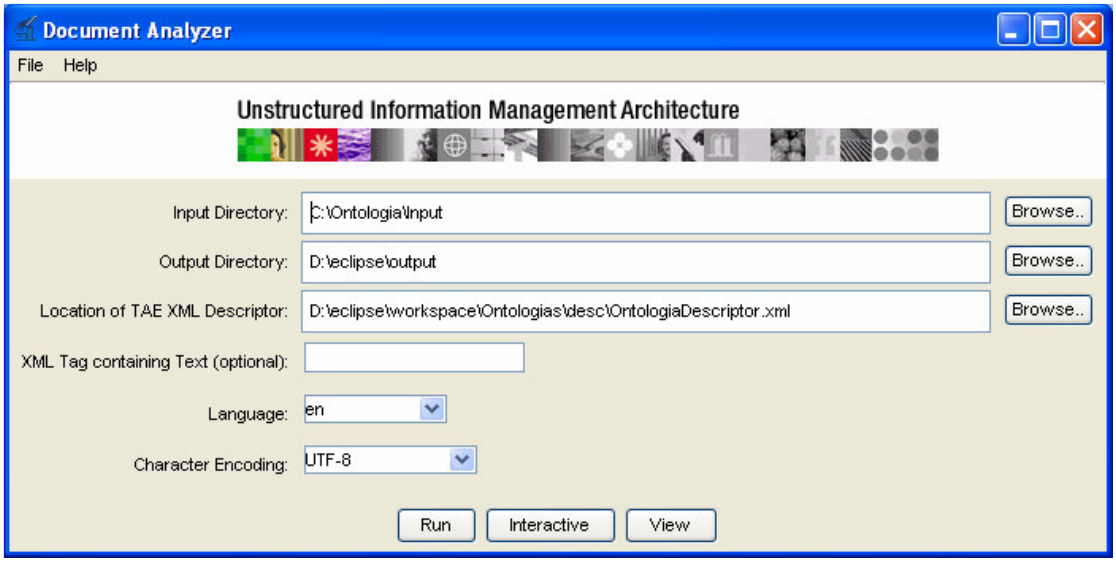


Figura 13: Ejecución del *DocumentAnalyzer.bat*

6.3.6. **Construcción de macro en Outlook**

Outlook 2002 o 2003 ofrece una gama amplia de posibilidades, conjuntamente con Visual Basic, para procesar los correos electrónicos se visualizan.

El objetivo de esta macro es, que cuando llegue un nuevo correo electrónico, se verifique cuál fue la fuente (remitente) que lo envió y si es de las listas de distribución definidas, se guarde en disco duro en la carpeta C:\Ontologia\Input que fue la definida para los archivos de entrada. Este programa es la fuente de información para los programas Java que buscan en la Ontología, dado que descargará sobre dicha carpeta, cada correo que de acuerdo al criterio, fue filtrado.

Como prerrequisito, es necesario que, en la bandeja de entrada (Inbox) se cree una carpeta llamada Vulnerabilidades, que servirá para mover todos loc correos que sean filtrados de acuerdo con el remitente.

Desde Outlook, abrimos el editor de macros: Herramientas>Macro>Editor de Visual Basic, damos doble clic a la función ThisOutlookSession para comenzar a editar la macro respectivo.

La macro está compuesta por funcionalidades Visual Basic, capaces de realizar una acción determinada al momento de llegar un correo electrónico, invocando la inicialización de la aplicación: CreateObject("Outlook.Application") . Luego de llegado un correo, se verifica el nombre del remitente, contrastado con la lista definida[67]:

```
Const Dest0 As String = "urn:schemas:httpmail:fromemail = 'baseacis-bounces@acis.org.co'"
Const Dest1 As String = "urn:schemas:httpmail:fromemail = 'unaaldia-admin@hispasec.com'"
Const Dest2 As String = "urn:schemas:httpmail:fromemail = 'segurinfo@acis.org.co'"
Const Dest3 As String = "urn:schemas:httpmail:fromemail = 'segurinfo-bounces@acis.org.co'"
```

Si el destinatario corresponde a uno de los de la lista, se creará archivo tipo texto, cuyo nombre es el del remitente más un número aleatorio para evitar que sean sobreescritos. Se captura el Body del correo y se incluye dentro de este archivo que se creó. Una vez se culmine la operación, este correo será movido hacia la carpeta Vulnerabilidades residente dentro del Inbox.
Esto implica, que para cada correo nuevo, se verifica el origen y dependiendo de quién sea se procesa o no.

6.3.7. **Consolidación de desarrollos: Solución general.**

La solución general está dada por la unión de cada uno de los componentes en un todo, esto es:

1) Desde el correo electrónico se filtrarán los correos de orígenes que proveen la información referente a las diferentes vulnerabilidades de seguridad que a diario salen, guardando en disco duro (carpeta de entrada) cada uno de estos correos. Cuando un correo sea de uno de los orígenes definidos, se moverá el correo a otra carpeta, para evitar ser procesado 2 veces. Este será la fuente de la información para UIMA.
2) A través de UIMA se ejecuta el descriptor, levantando las clases Java que realizan una búsqueda periódica de nuevos archivos en la capeta de entrada, si se encuentra un archivo, éste será procesado, haciendo la búsqueda de su contenido dentro de la ontología y lo ubica en la carpeta respectiva dependiendo de como sea el resultado, exitoso o no.
3) Ambos programas (Outlook y UIMA) se ejecutan *Stand Alone* como bucles infinitos, con periodicidad de búsqueda dependiente de los parámetros, para Outlook, depende de la llegada de los correos y para UIMA cada media hora (30 minutos).
4) Es importante tener en cuenta que, para iniciar UIMA, es necesario que halla al menos un archivo en la carpeta de entrada (*C:\Ontologias\Input*).

6.4. PRUEBAS REALIZADAS

Para la verificación, tanto del funcionamiento de la herramienta como de la efectividad de su clasificación, la cual es la clasificar exitosamente al menos el 75% de las fuentes, se realizo la siguiente prueba:

- En la carpeta de entrada (*input*), se dejaron un total de 20 archivos procedentes de diferentes fuentes de información sobre vulnerabilidades de seguridad.
- La fuente de información más representativa de donde se extrajo el contenido de algunas vulnerabilidades fue http://www.clcert.cl/ y de http://www.cert.org.mx/
- Se contó con la colaboración de un experto en seguridad [68], quien analizo cada uno de los archivos guardados realizando la respectiva clasificación.
- Se realizo la clasificación con la Herramienta automática.

Una vez concluida la prueba, se obtuvieron los siguientes resultados:

No.	Nombre Archivo	Clasificación según el experto:	Clasificación según la Herramienta	Resultado: 0 –No exitoso 1 – Exitoso
1	Alcatel-8467590	No esta dentro del Dominio	No esta dentro del dominio	1
2	Alerta CLCERT-2002-001	No Esta dentro del Dominio	No esta dentro del dominio	1
3	Alerta CLCERT-2002-020	No esta dentro del Dominio	Si esta dentro del dominio	0
4	Alerta CLCERT-2003-009	Esta dentro del Dominio	Si esta dentro del dominio	1
5	Alerta CLCERT-2004-003	Esta dentro del Dominio	Si esta dentro del dominio	1
6	Cisco_system-0.73682763	No esta dentro del Dominio	No esta dentro del dominio	1
7	Hector Fernando Vargas Montoya-0.56782	Esta dentro del Dominio	Si esta dentro del dominio	1
8	Hector Fernando Vargas Montoya-0.101359	Esta dentro del Dominio	Si esta dentro del dominio	1
9	Hector Fernando Vargas Montoya-0.161379	Esta dentro del Dominio	No esta dentro del dominio	0
10	Hector Fernando Vargas Montoya-0.8873676	No esta dentro del Dominio	No esta dentro del dominio	1

11	Oracle-Vulne-0.7492925	Esta dentro del Dominio	Si esta dentro del dominio	1
12	phishing-768904	No esta dentro del Dominio	No esta dentro del dominio	1
13	RADIUS-876894560	No esta dentro del Dominio	No esta dentro del dominio	1
14	Una-al-dia-0.34561	Esta dentro del Dominio	Si esta dentro del dominio	1
15	Una-al-dia-0.254684	No Esta dentro del Dominio	No esta dentro del dominio	1
16	Una-al-dia-0.983456	No esta dentro del Dominio	No esta dentro del dominio	1
17	Una-al-dia-0.2534356	No Esta dentro del Dominio	No esta dentro del dominio	1
18	Una-al-dia-0.29384769	No esta dentro del Dominio	No esta dentro del dominio	1
19	UNAM-CERT-897689037	Esta dentro del Dominio	Si esta dentro del dominio	1
20	Vulnerabilidad en Cisco PIX	No esta dentro del Dominio	No esta dentro del dominio	1

Donde:

- ***Exitoso (valor 1):*** *Cuando la herramienta al clasificar coincidió con la clasificación dada por el experto.*
- ***No exitoso (valor 0):*** *Cuando la herramienta al clasificar,* ***NO*** *coincidió con la clasificación del experto.*

Total exitosos = 18
Total no exitosos = 2

7. PROBLEMAS PRESENTADOS DURANTE EL DESARROLLO DEL PROYECTO

Durante el desarrollo del proyecto, algunos percances se fueron presentando. En especial el que tiene que ver con la configuración y parametrización del Classpath.

Hubo mucha pérdida de tiempo hasta lograr configurar cada classpath: Para los archivos .jar, para UIMA y para Eclipse.

Con respecto a la integración de los diferentes plugins y eclipse, es conveniente tener en cuenta la configuración del servidor Proxy (en el caso tal que estemos detrás de una red LAN) dentro de Eclipse, dado que estos plugins se actualizan directamente desde Internet.

8. MANUAL DEL USUARIO

En los siguientes ítems se da a conocer la forma como, se ha instalado las diferentes herramientas para la construcción de la solución y su forma de uso. Es importante aclarar que se hará referencia a los manuales oficiales de instalación, haciendo énfasis en puntos claves de las instalaciones.

8.1.1. **Instalación**

Todos los archivos ejecutables y manuales serán incluidos dentro del CD entregable de este proyecto de grado.

Pre-requisito: Se debe instalar Java 1.4 o superior.

8.1.1.1. Protégé y RACER

- Obtener el software RACER de http://www.cs.concordia.ca/~haarslev/racer/: Es un archivo ejecutable, que podemos parametrizar de acuerdo con el puerto http que se requiera, por defecto, escucha por el puerto 8080.
- Obtener el software de instalación de protégé (Versión 3.0) desde http://protege.stanford.edu/ y su respectivo manual, seguir las instrucciones de instalación.
- Bajar los plugins OWLviz para visualizar las ontologías desde protégé en http://www.co-ode.org/downloads/owlviz/co-ode-index.php, una vez en disco duro, se debe descomprimir en la carpeta de Plugins donde quedó instalado Protégé.
- Configurar en protégé, el puerto por el cual hablara con RACER:
    - Menú *OWL>Preferentes* en el Tab *General*, opción *Reasoner URL: http://localhost:8080*

8.1.1.2. UIMA

- Obtener el software de instalación en http://www.research.ibm.com/UIMA/
- Instalar de acuerdo al procedimiento dado por el fabricante.

8.1.1.3. Eclipse

- Obtener el software de instalación desde www.eclipse.org versión 3.1
- Una vez instalado, podemos instalar los diferentes plugins que requerimos así:
  - Desde el menú ayuda>*Software Update*>*Find and Install*>*Search for new features to install*
  - Seleccionar los plugins EMF y Jena
- En la carpeta de instalación de UIMA, queda una carpeta ***EclipsePlugins*** que contiene los plugins para Eclipse (para trabajar con descriptores de UIMA con eclipse) para las versiones 2.0 ó 3.0. Se debe descomprimir en la carpeta de plugins de Eclipse.
- Para mas información, ver el manual de usuario de Eclipse y UIMA para la instalación de estos plugins.

8.1.1.4. Office Outlook 2002 - 2003

En la instalación de Microsoft Office, se debe instalar este componente.

8.1.2. Modo de uso

Pre-requisito:

- En el CD del proyecto del grado, se encuentra la carpeta comprimida de los archivos fuentes trabajados en eclipse, con el nombre ***Ontologías***, este comprimido, debe descomprimirse en disco duro
- Es necesario crear 4 carpetas donde residirán los archivos de entrada y salida:

  C:\ontologia\Input

  C:\Ontologia\Ontologia - Acá se debe guardar la ontología (.owl)

C:\Ontologia\Procesado-NO-OK

C:\Ontologia\Procesado-OK

1) Si se requiere modificar, visualizar o actualizar la ontología, se abre Protégé 3.0, abriendo la ontología respectiva:

2) **Eclipse:** Si se requiere visualizar, modificar o cambiar los archivos fuentes, iniciamos Eclipse.exe, que cargará todos los plugins contenidos en la carpeta respectiva, solicita el Workspace donde va a trabajar:

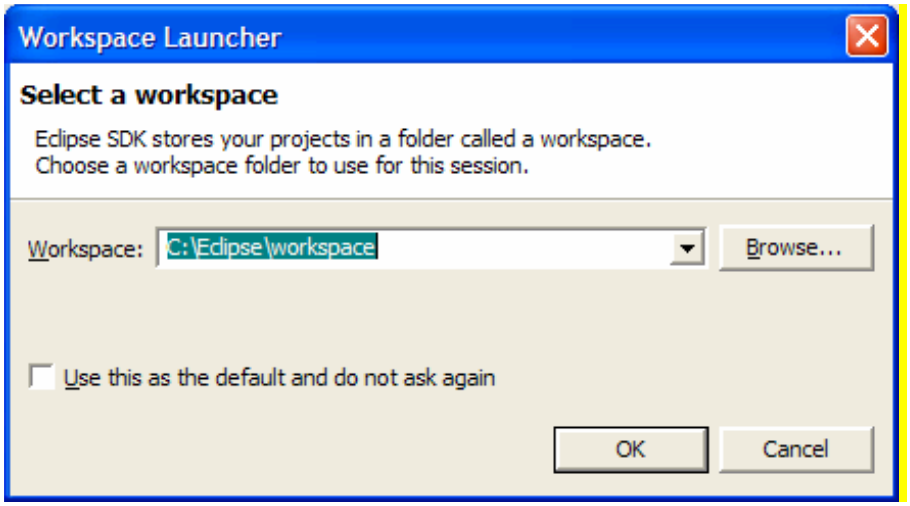


Figura 14: Selección del Workspace en Eclipse

- Se debe desplegar la página de inicio de Eclipse(la cual podemos cerrar)
- Se visualiza el explorador de paquetes, en este, damos clic derecho y la opción de *Import*

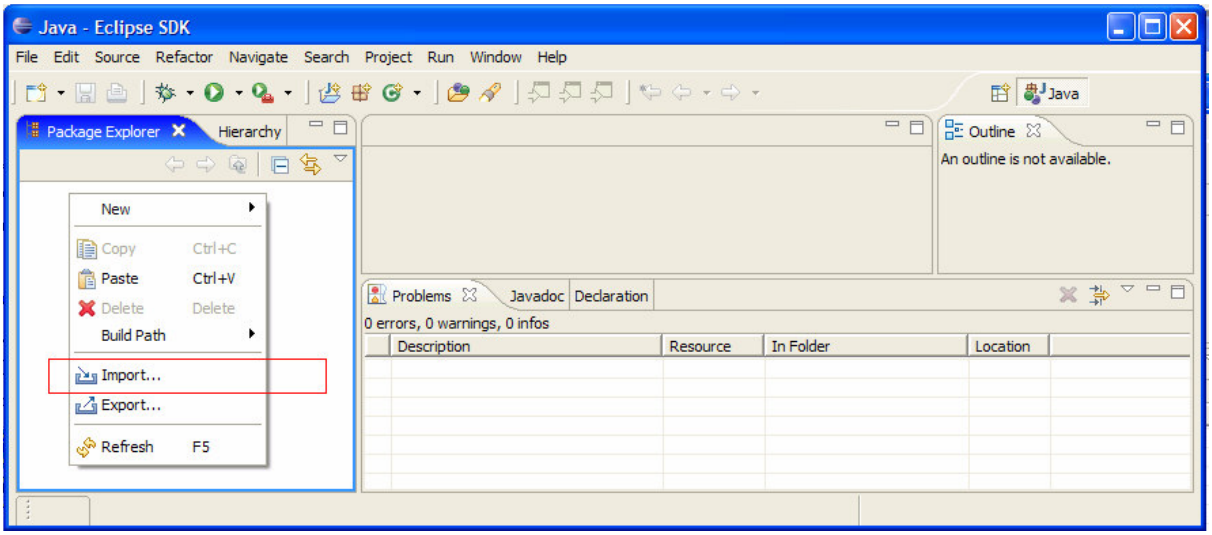


Figura 15: Import del proyecto dentro de UIMA

- Seleccionamos *Existing Projects into Workspace* luego *Next,* en la opción *Select root directory* seleccionado la carpeta ***Ontologías*** previamente guarda, luego la opción ***finish:***

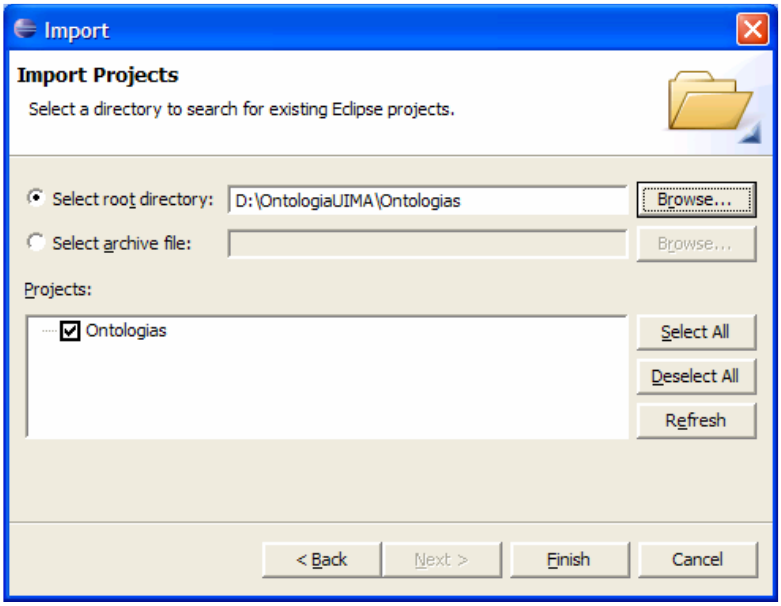


Figura 16: Proceso de *import* y selección del proyecto

- Se cargara el proyecto completo:

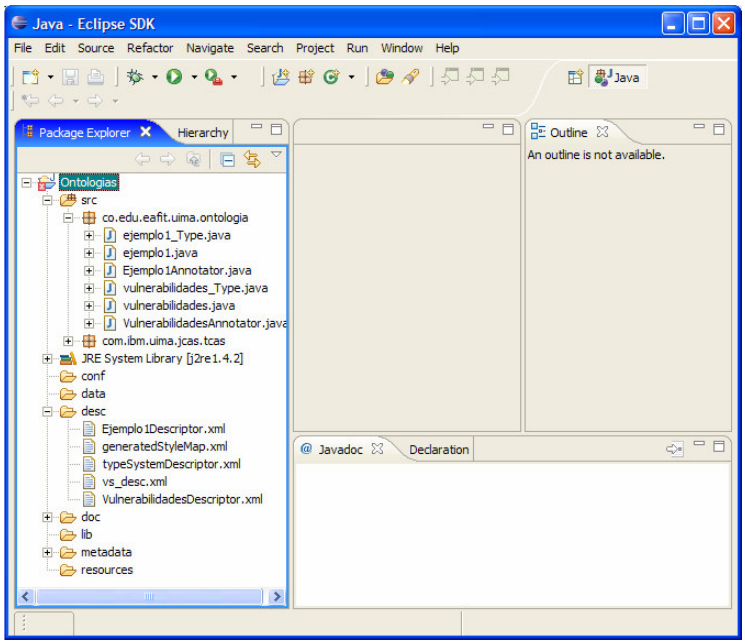


Figura 17: *Import* completo del proyecto dentro de Eclipse

3) Para visualizar las clases, solo basta con darle doble clic a la clase que se desee.
4) Para visualizar los descriptores o los motores, damos clic derecho, *open with>component descriptor editor*

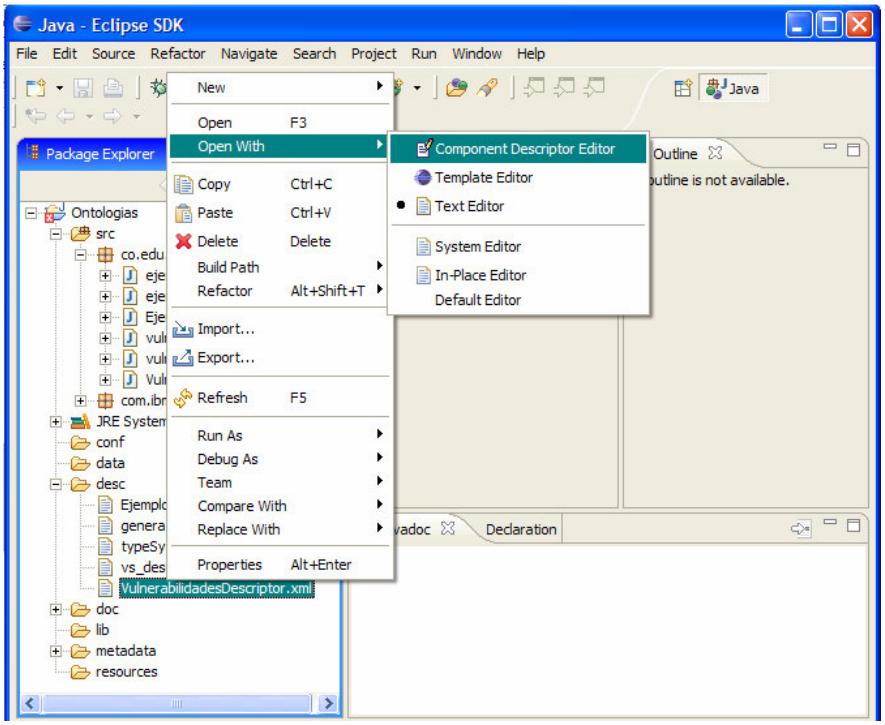


Figura 18: Acción para abrir y visualizar los descriptores

5) **Outlook:** El código fuente de la macro en outlook fue guardado como archivo texto dentro del CD de documentación del proyecto de grado, para incluirlo dentro de Outlook:

- Abrir Outlook 2002 o 2003
- En el menú herramientas>macros>editor de visual Basic
- Dar doble clic en ***ThisOutlookSession*****,** y copiar el contenido del archivo texto acá.
- Salvar el archivo y ejecutar (run) la macro.
- Ésta quedará pendiente hasta la llegada de un nuevo correo electrónico, alimentando la carpeta C:\Ontologia\input.

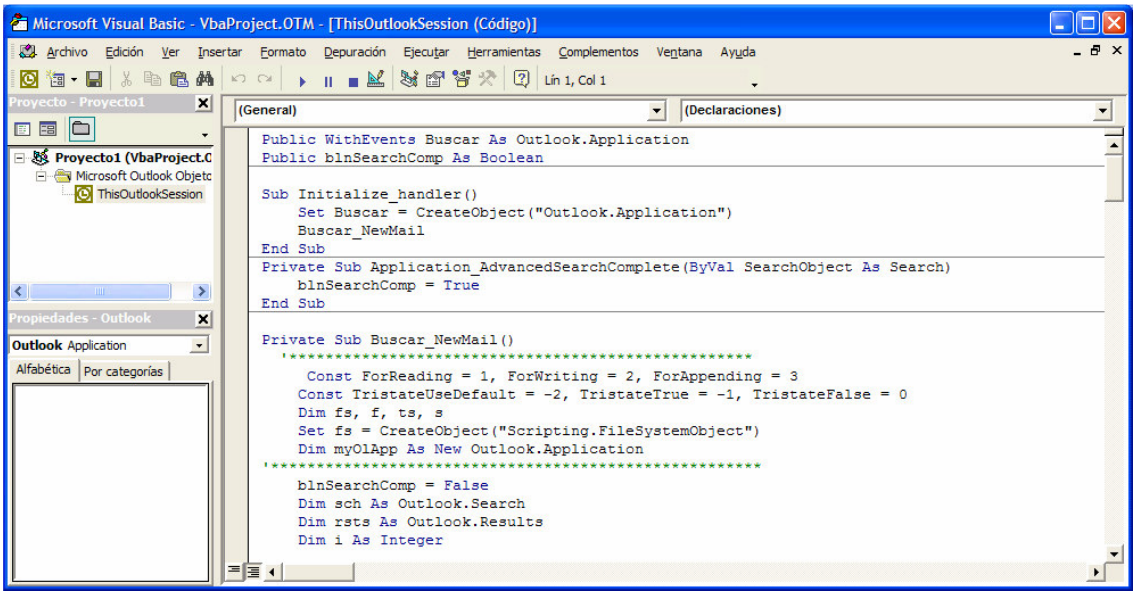


Figura 19: Visualización de la macro en Outlook

6) **UIMA:** Este es el último paso a realizar, para ello es conveniente tener las siguientes consideraciones:

- Archivos .jar: Para mayor comodidad, los archivos jena.jar, antlr.jar, y commons-logging.jar copiarlos en la carpeta de librerías de UIMA (%UIMA_HOME%\lib\)

- Modificar el archivo ***documentAnalyzer.bat*** adicionándole la opción *-cp "%UIMA_CLASSPATH%:."*

- Modificar el archivo *setUimaClassPath.bat* ubicado en la carpeta *BIN* de UIMA, adicionándole las líneas necesarias para que puede ver los archivos .jar:

```
set
UIMA_CLASSPATH=%UIMA_HOME%\lib\jena.jar;%UIMA_HOME%\lib\antlr.jar;%UIMA_HOME%\lib\co
mmons-
logging.jar;%UIMA_HOME%\docs\examples\resources;%UIMA_CLASSPATH%;%UIMA_HOME%\lib\ui
ma_core.jar;%UIMA_HOME%\lib\uima_cpe.jar;%UIMA_HOME%\lib\uima_jcas_builtin_types.jar;%UIMA
_HOME%\lib\uima_jcasgen_gen.jar;%UIMA_HOME%\lib\uima_tools.jar;%UIMA_HOME%\lib\uima_exa
mples.jar;%UIMA_HOME%\lib\uima_adapter_messaging.jar;%UIMA_HOME%\lib\uima_adapter_soap.j
ar;%UIMA_HOME%\lib\uima_adapter_vinci.jar;%UIMA_HOME%\lib\uima_cvd.jar;%UIMA_HOME%\lib\j
uru.jar;%UIMA_HOME%\lib\siapi.jar;%UIMA_HOME%\lib\xsdbeans.jar;%CATALINA_HOME%\webapps
\axis\WEB-INF\lib\activation.jar;%CATALINA_HOME%\webapps\axis\WEB-
INF\lib\axis.jar;%CATALINA_HOME%\webapps\axis\WEB-INF\lib\commons-
discovery.jar;%CATALINA_HOME%\webapps\axis\WEB-INF\lib\commons-
logging.jar;%CATALINA_HOME%\webapps\axis\WEB-
INF\lib\jaxrpc.jar;%CATALINA_HOME%\webapps\axis\WEB-
INF\lib\mail.jar;%CATALINA_HOME%\webapps\axis\WEB-
INF\lib\saaj.jar;%UIMA_HOME%\lib\vinci\jVinci.jar;%UIMA_HOME%\lib\xml.jar;%UIMA_HOME%\lib\dltj
50.jar;%UIMA_HOME%\lib\dltj50An.jar;%UIMA_HOME%\lib\icu4j.jar;%_REALLYNOQUOTES%;
```

- **Ejecución de *documentAnalyzer.bat:*** Este se debe ejecutar posicionados en la carpeta ***Bin*** del proyecto en eclipse, si nuestro proyecto está ubicado en *C:\Eclipse\workspace\ontologia* , dentro de éste está la carpeta *bin.* A través de la línea de comando, nos paramos en esta carpeta y ejecutamos ***documentAnalyzer.bat***

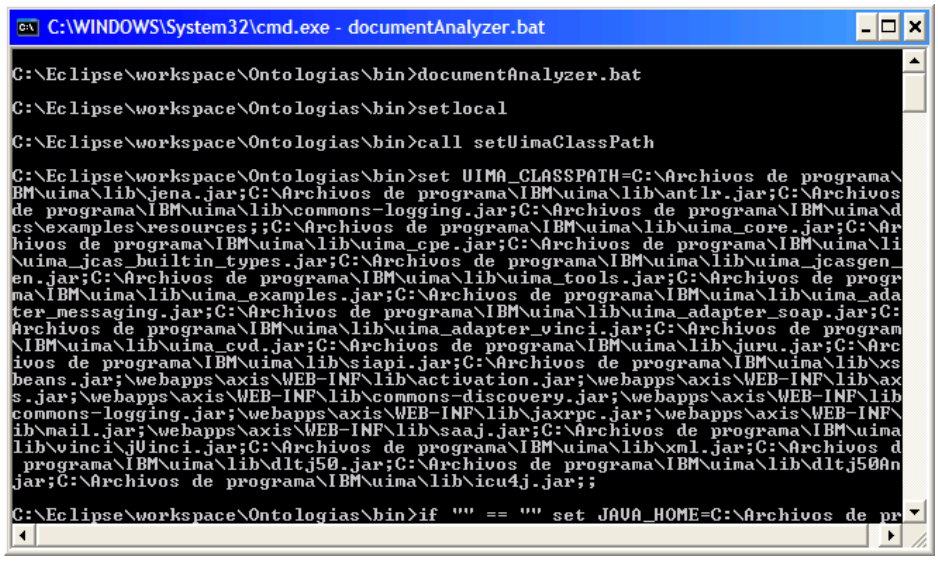


Figura 20: Opciones desplegadas con la ejecución de *documentAnalyzer.bat*

Debe desplegar la interfaz GUI de UIMA:

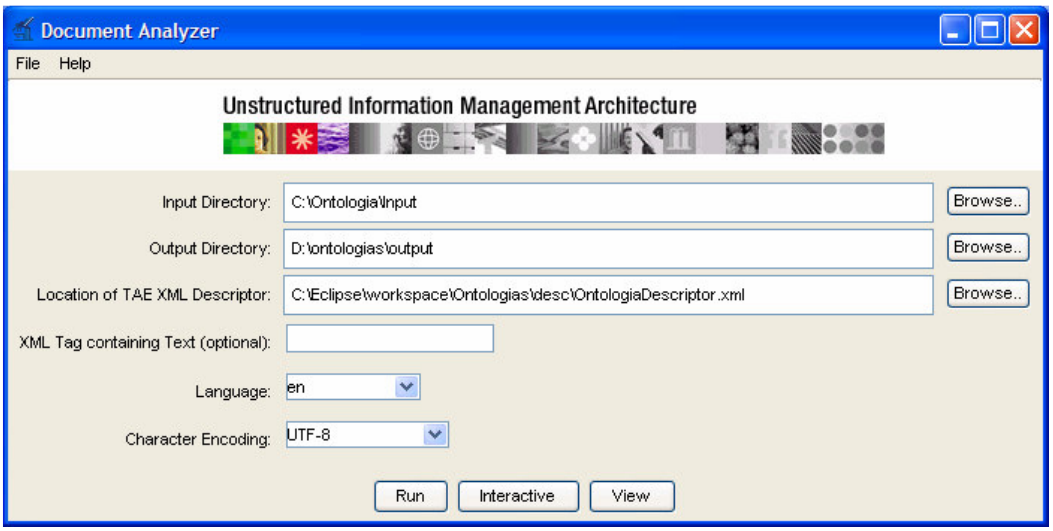


Figura 21: Interfaz GUI de UIMA

- Seleccionamos el directorio de entrada. Éste debe ser el mismo que fue configurado en Outlook para salida (C:\ontologia\input).
- Seleccionar el descriptor a utilizar, para el proyecto de grado es *OntologiaDescriptor*

- Luego *Run*, se queda leyendo los archivos que encuentre en la carpeta de entrada y los irá procesando (analizando y moviendo a la carpeta destino de acuerdo si fue encontrado en la ontología).

9. CONCLUSIONES

- Tener información centralizada y a tiempo genera una ventaja competitiva en cualquier campo de acción. Para el caso de seguridad, se estaría reduciendo el impacto económico que puede generar la materialización de un riesgo que no se conoce o que es difícil saberlo cuando existen tantas fuentes de información. Esta prueba de concepto es de gran apoyo para que los oficiales de seguridad o analistas de seguridad, filtren la información recibida.

- La herramienta informática basada en ontologías para la clasificación automática de información en seguridad, es una prueba de concepto que arroja un resultado exitoso, dado que, de acuerdo a la prueba realizada, se pudo clasificar exitosamente el 90% de la información, superando el mínimo establecido en el alcance (75%).

- Este resultado exitoso de la herramienta representa la posibilidad de tener información no estructurada, a partir de un dominio de conocimiento – Ontología-, en información útil para un público objetivo.

- Para cada dominio de interés dentro de una ontología, es necesario tener conocimiento claro de que se requiere y cual es el objetivo de la misma, esto concentrara y dará más efectividad a la clasificación de la información.

- Las Ontologías como herramientas potencialmente útiles dentro de la Web semántica, pueden extenderse a cualquier dominio de conocimiento. Para el caso de la seguridad informática, provee una

forma diferente de modelar el conocimiento y de adecuar las necesidades propias del área.

- Esta herramienta para la clasificación de información, se desea que tome una buena importancia para el idioma español, dado que, existen algunas otras que realizan esta clasificación semántica, pero en inglés -como WordNet- o acercamientos en idiomas como Holandés, Italiano, Alemán, francés, checo y Estoniano – como EuroNet [69].

10. RECOMENDACIONES

- Es necesario tener en claro en funcionamiento del *Classpath* y su configuración, dado que se puede perder mucho tiempo buscando problemas innecesarios.
- Es necesario aclarar que las personas interesadas en este prototipo como prueba de concepto o como base para uno nuevo, se enfoquen cuidadosamente en la reestructuración de la Ontología, con el objeto que no pierda la esencia, además enfocarse en el conocimiento de que pueda adquirir con respecto a UIMA, es una herramienta con muchos otros conceptos que pueden ser útiles.
- Sería conveniente, realizar pruebas con una población muestral de correos mas alta a la acá realizada y tener más de un experto realizando la clasificación.

11. TRABAJO FUTURO

Se pretende que en trabajos futuros, existan varios campos de acción, los cuales se pueden explorar, en especial, los siguientes:

El primero es la recuperación de información bilingüe y multilingüe, interrelacionando diferentes dominios de conocimiento que sean a fin e interactuando con Ontologías que en el momento están funcionando y arrojando buenos resultados.

El segundo es la construcción de filtros para obtener información de diferentes fuentes. Más aún, queda abierta la posibilidad de que la herramienta se apoye en un agente inteligente para recolectar automáticamente la información en las páginas Web de los diferentes grupos de hackers y permitir el tratamiento y clasificación automática de las mismas. Esto permitiría una actividad más proactiva por parte de los oficiales de seguridad al permitir detectar ciertas vulnerabilidades en herramientas software a tiempo y así, dar un costo-beneficio más conveniente para las organizaciones en cuestión de seguridad informática.

Adicional a esto, se deja abierta la posibilidad de darle una mejora a al herramienta, tal es el caso de complementar con la selección de las diferentes rutas donde residen los archivos a través de un pop-up (y no rutas estáticas como esta actualmente), trabajar más a fondo dentro de la ontología, todo el tema de sinónimos, que se ha manejado muy tímidamente en este proyecto, de la misma manera, al manejo de palabras compuestas (no limitarse solo a palabra únicas si no a la composición de las mismas con algún sentido semántico).

12. ANEXOS

Conjuntamente con este documento, se entrega un CD-ROM con la información referente a todos los anexos distribuidos en carpetas así:

Nombre Carpeta	Sub-Carpeta	Contenido
Anexos-instaladores-manuales	Eclipse	Instalador de Eclipse y su manual de usuario.
	Protege	Instalador protégé y sus manuales de referencia.
	Racer	Ejecutable RACER.EXE y su manual de referencia.
	Sans Institute	Información sobre las 20 vulnerabilidades publicadas por Sans Institute.
	UIMA	Instalador de UIMA y su manual de referencia.
Documentos		Contiene: • El documento del proyecto de grado (formato Word). • Presentación del proyecto • Articulo del proyecto, según requisitos exigidos como entregable con el proyecto de grado.
Macro-outlook		Código fuente de la Macro para Outlook.
Ontología		Contiene la Ontología desarrollada para el proyecto.
Proyecto-UIMA-eclipse		Contiene el proyecto para UIMA (comprimido) para ser importado desde Eclipse. Adicional a esto, las librerías .jar necesarias para interactuar con Jena.
Pruebas	Procesado-OK	Contiene los archivos que fueron procesados y catalogados por la herramienta como pertenecientes al Dominio, luego de la prueba (ítem 6.4 Pruebas realizadas)
	Procesado-NO-OK	Contiene los archivos que fueron procesados y catalogados por la herramienta como NO pertenecientes al Dominio, luego de la prueba (ítem 6.4 Pruebas realizadas)

13. BIBLIOGRAFÍA

a. **MONSALVE RIOS,** Mauricio Alberto: Utilización de Wordnet como Ontología para apoyar la Recuperación de documentos en un sistema gestor de información. proyecto de grado. Medellín 2005 - Universidad EAFIT- Biblioteca "Luis Echavarria Villegas".

b. **ONTOLOGÍAS**: OntoLingua: http://www.ksl.stanford.edu/software/ontolingua/, KA^2: http://ontobroker.semanticweb.org/ontos/ka2.html , SUMO: http://suo.ieee.org/SUO/SUMO/index.html, CyC: http://www.cyc.com/

c. **ESTUDIO** sobre ontologías: http://www.tramullas.com/pdf/ontologias.pdf

d. **HERRAMIENTAS** para la construcción de Ontologías: http://protege.stanford.edu/ , http://www.ksl.stanford.edu/software/chimaera/, http://www-ksl-svc.stanford.edu/ , http://www.geneontology.org , http://sourceforge.net/projects/geneontology , http://kaon.semanticweb.org/ , http://www.daml.org/ , http://www.ontoknowledge.org/oil/, http://oiled.man.ac.uk/ , http://www.ontoknowledge.org/oil/tool.shtml.

e. **WEB** Semántica: http://www.w3.org/

f. **MOTORES** de inferencia: http://www.sts.tu-harburg.de/~r.f.moeller/racer/ ó http://www.cs.concordia.ca/~haarslev/racer/, http://www.research.att.com/sw/tools/classic/

g. **ANÁLISIS** y recomendación sobre herramientas para la construcción de Ontologías: www.geneontology.org/email-go/go-arc/go-2003/att-1994/01-evalOntol.pdf

h. **SANS INSTITUTE**: http://www.sans.org/top20/

i. **MICROSOFT** outlook http://www.microsoft.com/latam/office/outlook/prodinfo/default.mspx

j. **ECLIPSE:** Software y documentación: http://www.eclipse.org

k. **UIMA:** Software y documentación: http://www.research.ibm.com/UIMA

l. **SEGURIDAD Y VULNERABILIDADES:** Grupos, empresas productoras de software o hardware y casas de antivirus: http://www.cert.org , www.hispasec.com , http://www.rediris.es , http://www.acis.org.co http://www.securityfocus.com , http://www.microsoft.com , http://www.symantec.com , http://www.acis.org.co , http://www.etek.com.co , http://www.mcafee.com , http://www.cve.mitre.org , http://www.sun.com

m. **DEFINICIONES** y diccionario: Real Academia Española de la Lengua, http://www.rae.es/ , http://es.wikipedia.org/wiki/

14. REFERENCIAS BIBLIOGRÁFICAS

[1] http://www.cert.org/stats/ : Informe estadístico anual (CERT/CC Statistics 1988-2005).

[2] Tomado de es.wikipedia.org/wiki/Framework

[3] http://www.sun.com

[4] Arevolo, W. y otros. "Hype Cycle for IT in Latin America." *Strategic Analysis Report,* R-20-2997. Stamford, CT: Gartner, Inc., June 19, 2003.

[5] Idem 1.

[6] http://www.cert.org , http://www.securityfocus.com, http://www.microsoft.com, http://www.symantec.com, http://www.acis.org.co, http://www.etek.com.co, http://www.mcafee.com, http://www.cve.mitre.org, http://www.sun.com

[7] Ver item 5.2. de este documento: Ontologias.

[8] Significado de Informática, Real Academia Española de la Lengua, http://www.rae.es/

[9] Definición tomada de: http://es.wikipedia.org/wiki/

[10] En un análisis de riesgos, existen 3 formas de tratar el riesgo: Mitigarlo, transferirlo o asumirlo, [esto depende de las circunstancias de cada organización o negocio.

[11] http://www.sans.org/top20/, versión en español: http://www.sans.org/top20/spanish_v2.php

[12] En el informe oficial de SANS, se tiene además, las posibles soluciones de las vulnerabilidades [mas representativas de acuerdo al sistema operativo.

[13] Tomado de http://www.doi.org/handbook_2000/glossary.html

[14] Ver numeral 5.6 de este documento: Recuperación de la información

[15] Extraído de http://www.w3.org/TR/2004/REC-webont-req-20040210/

[16] Ver numeral 5.3.6 de este documento: Lenguajes de marcado para la Web Semántica.

[17] **Using explicit ontologies in KBS development.** http://ksi.cpsc.ucalgary.ca/IJHCS/VH/

[18] Ver numeral 5.2.6 de este documento.

[19] Tomado del Tutorial Protégé, sesión 3.2.2, pág. 15.

[20] Ver numeral 5.4.1 de este documento.

[21] Sir Timothy (Tim) John Berners-Lee: creador de W3C http://es.wikipedia.org/wiki/Tim_Berners-Lee

[22] http://es.wikipedia.org

[23] **MONSALVE RIOS,** Mauricio Alberto: Utilización de Wordnet como Ontología para apoyar la Recuperación de documentos en un sistema gestor de información. proyecto de grado. Medellín 2005 - Universidad eafit- Biblioteca "Luis Echavarria Villegas.

[24] http://www.maccare.com.ar/entrevista_berners.htm

[25] http://www.w3c.es/divulgacion/guiasbreves/WebSemantica

[26] Fuente: http://www.w3.org/2001/09/21-orf/hagino-sw/swlevels.gif

[27] Tomado de http://www.cs.umd.edu/projects/plus/SHOE/spec.html

[28] Tomado de http://www.ea1uro.com/traductor.html

[29] http://www.ai.sri.com/pubs/full.php?id=676

[30] http://www.daml.org/

[31] http://www.w3.org/RDF/

[32] http://es.wikipedia.org/wiki/RDF

[33] http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[34] http://www.w3.org

[35] http://www.w3c.es/Prensa/2004/nota040210.html

[36] http://es.wikipedia.org/wiki/Inteligencia_artificial

[37] http://es.wikipedia.org/wiki/Lengua_natural

[38] http://www.sun.com
[39] http://www.eclipse.org/
[40] [La interfaz **IDE** (Integrated Drive Electronics * Integrated development environment) un entorno integrado de desarrollo.
[41] Ver sesión 5.5.5 de este documento.
[42] http://www.research.ibm.com/UIMA/
[43] http://www.ibm.com/us/
[44] http://www.microsoft.com/
[45] Tomado de http://www.microsoft.com/latam/office/outlook/prodinfo/default.mspx
[46] http://www.ksl.stanford.edu/software/chimaera/
[47] http://protege.stanford.edu/
[48] http://www.ontoknowledge.org/tools/ontoedit.shtml
[49] http://kaon.semanticweb.org/
[50] http://oiled.man.ac.uk/
[51] http://projects.semwebcentral.org/softwaremap/trove_list.php?form_cat=325&discrim=235
[52] http://projects.semwebcentral.org/softwaremap/trove_list.php?form_cat=316
[53] http://www.racer-systems.com/index.phtml
[54] http://jena.sourceforge.net/
[55] **MONSALVE RIOS,** Mauricio Alberto: Utilización de Wordnet como Ontología para apoyar la Recuperación de documentos en un sistema gestor de información. proyecto de grado. Medellín 2005 - Universidad eafit- Biblioteca “Luis Echavarria Villegas”
[56] http://www.gartner.com
[57] Baeza-Yates, Ricardo. Depto. de Ciencias de la Computación. Universidad de Chile
[58] Tomado de: http://tramullas.com/documatica/3-4.html
[59] Evaluación de herramientas: http://bioinformatics.oxfordjournals.org/cgi/reprint/19/12/1564?ijkey=16khpsMrGKoHz&keytype=ref
[60] Ver ítem 7.1.1.1 Instalación Protégé: Manual de Usuario.
[61] www.microsoft.com
[62] Para una configuración mas precisa, ver el manual del usuario.
[63] Ver Manual del usuario: Instalación.
[64] Ver manual del Usuario: Instalación.
[65] Se describe mas adelante y su código fuente esta en los Anexos.
[66] Ver ítem 5.5.8 Jena de este mismo documento.
[67] Ver sesión 6.3.3.Clasificación de fuentes de información. De este documento.
[68] Ingeniero Gonzalo Alberto Torres, Coordinador del Equipo de Seguridad Informática-Empresas Públicas de Medellín.
[69] EuroWordNet: http://www.illc.uva.nl/EuroWordNet