UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

"CONTROL DE ACCESO SEMANTICO"

T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADA EN CIENCIAS DE LA COMPUTACION

P R E S E N T A :

GRECIA GARCIA GARCIA

FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS: DR. SERGIO RAJSBAUM GORODEZKY

2005

UNIVERSIDAD NACIONAL AVENOMA DE MEXICO

orizo a la Direcció... de la UNAM a ... yaso el conten... ocional.
NOMBRE: Grecia García García
FECHA: 31 de mayo del 2005
FIRMA:

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

"Control de Acceso Semántico"

realizado por Grecia García García

con número de cuenta 09711317-7 , quien cubrió los créditos de la carrera de:

Lic. en Ciencias de la Comoutación

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director Propietario	Dr. Sergio Rajsbaum Gorodezky
Propietario	Dra. Elisa Viso Gurovich
Propietario	M. en C. José de Jesus Galaviz Casas
Suplente	Dr. David Arturo Rosenblueth Laguette
Suplente	Lic. en C. de la C. Manuel Alberto Sugawara Muro

Consejo Departamental de Matemáticas

Dr. Francisco Hernández Quiroz

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMÁTICAS

A mis padres, Enriqueta y Rafael.
A mis hermanas, Roma y Francia.
A mis amigos.

Agradecimientos

A mis padres y hermanas por su ayuda, apoyo y compañía de siempre.

A mi asesor Sergio Rajsbaum por todo el apoyo, tiempo y conocimiento que me dedicó.

A mis sinodales, David Rosenblueth, Elisa Viso, Manuel Sugawara y José Galaviz por sus valiosos comentarios y el tiempo que se tomaron en la revisión de este trabajo.

A los profesores de la facultad: Hugo Rincón, Rafael Rojas y Antonio Neme.

A Ruy Fabila, Mary Carmen Trejo, John Giddings, Josafat Guerrero, Gustavo González y Francisco Escalona por su compañía y ayuda en todo momento de la carrera.

Agradezco a mis amigos, que me conocen desde hace muchos años y a los que debo parte de lo que soy: Patricia Reyes, Carlos Salmerón, Mayra Martínez, Gisela Martínez, Edgar Gómez, Ramiro Sánchez, Josué Hernández, Luis Manuel Ortíz, Félix Velazco y Hugo Ávila.

A las personas que conocí en la facultad y que son un ejemplo a seguir: Karla Ramírez, Manuel Sugawara, Francisco Solsona, Ivan Hernández, Jorge Santos, César López y Yuri Vasilevski.

A las personas del instituto que me acompañaron en mi estancia ahí: Yesenia, José, Miguel Angel, Esteban, Pietra, Abraham, Paulina y Daniel.

A Rubén, Carlos y Federico del Departamento de Cómputo del Instituto de Matemáticas por la ayuda que me dieron en diversos problemas técnicos.

Índice general

Introducción

Motivación

La historia de la ciencia está llena de invenciones que tienen el propósito de mejorar la calidad de vida de la sociedad para dar un mejor futuro a las generaciones que están por venir, o que simplemente han surgido por necesidad. Para cada paso de los descubrimientos realizados, y los que están en desarrollo, ha sido fundamental la comprensión del medio en donde se desarrolla el problema a solucionar, los objetos involucrados en él y las relaciones entre ellos. Es interesante notar que estos aspectos también son considerados (de alguna manera) en las relaciones cotidianas entre las personas; lo cual se verifica al observar el comportamiento que adoptan frente a una situación.

De las creaciones que más han tenido impacto en la sociedad, la computadora se encuentra entre ellas. El uso de esta máquina se ha incrementado notablemente en los últimos años, y su empleo en diversas áreas ha permitido lograr un avance significativo para varios campos del conocimiento como: matemáticas, física y biología, entre otros.

Ahora ha llegado el turno de emular de alguna forma tal mecanismo de comprensión para mejorar la relación entre el humano y la computadora, y el Web Semántico es una opción muy prometedora para solucionar en gran parte esta necesidad[1]. Las razones para desear tal capacidad van más allá de una simple imitación del comportamiento humano, las consecuencias de tal poder nos permiten explorar los dominios que se han quedado perdidos entre la inmensa cantidad de información. Además, en algunas sociedades como la nuestra, su aplicación puede tener como una consecuencia el aumento de la confiabilidad en los procesos gubernamentales.

Adicionalmente, el surgimiento de una nueva tecnología como lo es el Web Semántico (que inició su desarrollo en 1999) ofrece muchas oportunidades para

[1] La idea de modelar algún comportamiento relacionado con la naturaleza del hombre no es absolutamente novedosa. Norbert Wiener y Arturo Rosenblueth por los años 40 ya habían descubierto que las funciones de los organismos vivientes, incluyendo el sistema nervioso del hombre, pueden ser susceptibles a un análisis matemático ([12]); sólo que hasta ahora se ha hecho mucho más presente este aspecto.

proponer ideas y participar en un suceso que será de gran relevancia en el futuro de la humanidad, tal vez tanto como lo ha sido el Web. Es por eso que su estudio (desde cualquier perspectiva) resulta tan interesante y de gran importancia.

Antecedentes

El *Web Semántico* es una extensión del Web actual que permite dar un significado bien definido a los recursos del Web para permitir el trabajo en cooperación entre las computadoras y el hombre. El significado de los recursos es especificado en vocabularios conocidos como *ontologías.*

Como bien lo hace notar Tonti *et al.* ([23]), la capacidad de los lenguajes de representación del Web Semántico, junto con las herramientas para manipularlos, los hacen propicios para otros tipos de aplicación. En particular, la atención de este trabajo está dirigida a permitir el *acceso* a los recursos de un sistema. El acceso se concederá por medio de permisos, los cuales se otorgarán sólo a las personas que ejerzan un *rol.* A esto se le conoce como *Control de Acceso Basado en Roles* y es sobre lo que también trata este trabajo.

El Control de Acceso es un problema muy importante en casi cualquier actividad puesto que el uso de la información por personas ajenas a un asunto dado, puede tener consecuencias muy graves; pero ¿cómo se integrará la semántica a este tema? Algunos trabajos agregan simplemente una descripción de metadatos utilizando solamente XML con las etiquetas que se consideren adecuadas. Sin embargo, este procedimiento no cuenta con una semántica real asociada, por lo menos para la computadora. Y es aquí en donde los lenguajes del Web Semántico entran en juego para proporcionar tales características.

Objetivo

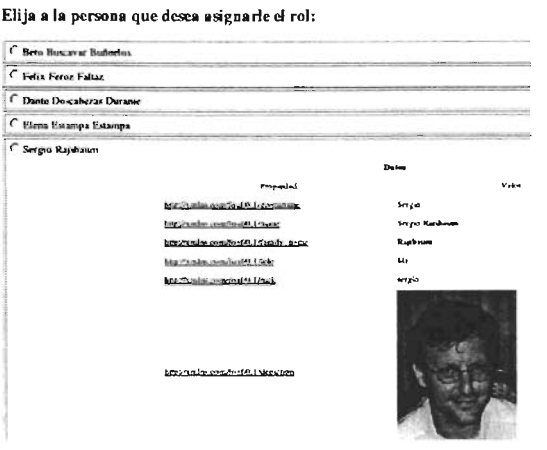
En particular, este trabajo inicia con un sistema que maneja un tipo de Control de Acceso para una compañía encargada de publicar una revista. Sin embargo, hace falta agregar algunas funcionalidades a esa aplicación con el propósito de observar las ventajas que le proporciona el Web Semántico. Para lograr esto, utilizo un modelo de Control de Acceso Basado en Roles estándar y lo uso como base para crear un modelo semántico, el cual sirve de guía en la construcción del sistema final. Lejos de crear sólo una aplicación web, se pretende obtener una aplicación que actúe automáticamente conforme al contenido de la información que maneja.

Así, nuestro objetivo es construir un sistema que trabaje con información semántica y que pueda realizar algunos procesos automáticamente. También se tendrá como resultado un sistema integrable al Web Semántico y se observarán

los efectos obtenidos de esta unión. Con esto, no sólo tenemos la ventaja de poder realizar procesos automáticos (que es lo que aporta seguir la arquitectura propuesta por Tim Berners-Lee) sino también la posibilidad de obtener todas las utilidades que ofrece el Web Semántico para cualquier aplicación web como: dar a conocer el servicio, mejor integración con otros sistemas (debido a que existirá una comprensión entre cada una de estas entidades), descripción del significado de los recursos, entre otras.

Una probadita de las ventajas

Una de las ontologías más relevantes es FOAF (*Friend Of A Friend*). Esta ontología expresa los conceptos más comunes asociados con personas y las relaciones que las involucran, lo que hace de este vocabulario uno de los más conocidos, y por lo tanto de los más usados. Cuando una ontología es utilizada en varios lugares (como es el caso de FOAF) el procesamiento sobre el significado de los datos se vuelve más sencillo. Emplear la misma información en distintos sistemas expresa que diferentes entidades comparten el mismo significado sobre algunos recursos y, en consecuencia, se obtiene una mejor integración entre distintas aplicaciones, incluyendo su incorporación semántica.



Como una muestra de la ventaja que ofrece un vocabulario común, el sistema desarrollado utiliza la información proporcionada por un archivo FOAF (que fue obtenida durante el registro de una persona en el sistema) para presentar algunas de las características que identifican a un individuo, como su correo electrónico o su fotografía. Este procedimiento tendrá como consecuencia el reconocimiento de un sujeto por elementos que lo involucran y que lo representan con más certeza. Así, cuando se muestre una lista de personas involucradas dentro del sistema, no sólo se presentarán los datos ordinarios, sino se hará una descrip-

ción tan detallada del individuo como la descripción que él mismo propuso en su archivo FOAF. Además, este procedimiento permite almacenar de forma estructurada la información sin saber de antemano el contenido de los datos.

Este ejemplo muestra que el Web Semántico no sólo proporciona la interoperabilidad entre diferentes aplicaciones, sino que considera el contenido semántico de los recursos que se comparten. La forma en que funciona el sistema final, está descrita en el manual presentado en el Apéndice A.

Distribución del Trabajo y Contribuciones

Para realizar este trabajo inicié con base en un sistema proporcionado por mi asesor, el Dr. Sergio Rajsbaum Gorodezky (y a quien agradezco esta contribución). Este sistema usa un tipo de Control de Acceso Semántico Basado en Roles, el cual describo en el Capítulo 3 debido a la importancia que toma en el trabajo.

Los temas fundamentales del trabajo son dos: el Control de Acceso y el Web Semántico; y los explico en el Capítulo 1. En el Capítulo 2 muestro una forma para combinar estas dos materias desde mi punto de vista y que tomo como base para alcanzar mi objetivo.

El Capítulo 4 trata sobre la adecuación del sistema presentado en el Capítulo 3 con base en el modelo propuesto en el Capítulo 2. Finalmente las conclusiones de este trabajo están presentadas en el Capítulo 5 así como los resultados, objetivos alcanzados y una idea de lo que se espera en el futuro.

Notas Finales

Doy gracias al proyecto PAPIIT "Técnicas del Web Semántico para la recolección de información distribuida y fundamentos teóricos" del cual surgió la idea para este trabajo y que me proporcionó el apoyo para realizarla. También agradezco al Instituto de Matemáticas por proporcionarme apoyo con la Beca de Lugar que se me otorgó. Reitero mi orgullo y satisfacción de pertenecer a esta casa de estudios, la Universidad Nacional Autónoma de México a la que doy gracias por hacerme partícipe de ella.

Finalmente quiero hacer mención que las equivocaciones cometidas dentro del texto, son responsabilidad mía.

Capítulo 1

Antecedentes

It is not the result of scientific research
that ennobles humans and enriches their nature,
but the struggle to understand while performing
creative and open-minded intellectual work.

- Albert Einstein, Berlín 1930.

1.1. Control de Acceso

Cada día aumenta la cantidad de datos almacenados en computadoras, sin embargo no siempre se desea que estén al alcance de cualquiera, por lo que es necesario que cada sistema regule el ingreso a ellos. Para esto, no sólo se requiere control sobre la(s) persona(s) que manipulará(n) los datos, sino también control sobre el tipo de acceso que se le(s) dará sobre ellos; por ejemplo, permiso de lectura de un archivo, permiso de ejecución, etc. Como parte de la regulación de los recursos se debe verificar que el acceso sea sólo a las personas autorizadas, esto es indispensable para asegurarse que los individuos sean realmente los que dicen ser.

De esta forma, un sistema de cómputo diseñado para realizar alguna operación de modificación o lectura de los datos del sistema, está obligado a tener un control de acceso a los recursos que resguarda.

1.1.1. Objetivos y construcción del Control de Acceso

El Control de Acceso (CA) tiene como objetivo determinar si un individuo puede realizar alguna acción sobre un recurso del sistema y, en caso de que esté autorizado, permitirle ejecutarla. Asimismo, tanto la asignación de los permisos a los usuarios como el hacer cumplir las condiciones de acceso a los

recursos forma parte de su finalidad.

Para lograr este objetivo son tres puntos los que se deben de seguir:

- Indicar las reglas sobre las cuales el acceso será regulado.
- Determinar la representación formal del CA (el modelo).
- Implementar el modelo.

Esto es, se requiere de la definición del conjunto de reglas que capturen todas las diferentes normas del sistema [18]; también es necesario un modelo que represente el control de acceso a seguir; y como último paso se implementa el modelo.

Hay que notar que es importante mantener la estructura y las relaciones entre las secciones del modelo al implementarlo, para asegurar que las propiedades teóricas se mantengan en la práctica. De esta manera, si el modelo representa un sistema seguro, entonces se dice que el sistema es seguro [18].

1.1.2. Características del Control de Acceso

Las características que se esperan del mecanismo del CA y que puedan ser verificadas en el modelo son:

- La detección de cualquier alteración.
- El mediar todos los accesos a los recursos.
- Observar que la sección de seguridad sea una región separada, en lo posible, del resto del sistema, además de comprender una pequeña parte de él para poder ser sometida a rigurosos métodos de verificación.

Debido a que la tecnología cambia con el tiempo y con ella las necesidades de los usuarios, se busca tener lo que yo llamaría *modelos perdurables.* Con esto me refiero a modelos suficientemente flexibles, independientes de la aplicación y fáciles de usar. Así, su tiempo de vida se incrementaría, además de que su mantenimiento sería relativamente fácil de efectuar.

La flexibilidad del sistema del CA se convierte en un factor muy importante por dos razones: la necesidad de un sistema a soportar los cambios de estructura de su organización y la posibilidad de integrarse en el futuro a otros grupos de sistemas (intrínsecamente heterogéneos y con sus propias reglas de acceso). Esto traería como consecuencia la obtención de un sistema interoperable, aunque según Yagüe [5], no es razonable esperar que un grupo de sistemas heterogéneos con diferentes propósitos y entidades, defina un conjunto homogéneo de criterios de autorización.

Un aspecto importante que se ha adoptado es separar la autenticación del usuario (con el uso de certificados o de las contraseñas) del control de acceso a los recursos. Aunque ambas partes son indispensables en el CA, tomar el reconocimiento del usuario por separado permite aplicar diversos métodos de autenticación, lo que simplifica la interoperabilidad con otros sistemas en caso de ser necesario.

1.1.3. Modelos de Control de Acceso

Existen diferentes modelos de CA que han sido desarrollados. Tres de ellos son los que han destacado, principalmente porque han servido como base a otros o han resuelto gran parte de las necesidades para varios sistemas. Estos son los siguientes:

- Control de Acceso Discreto (CAD)
- Control de Acceso Obligatorio (CAO)
- Control de Acceso Basado en Roles (CABR)

Aunque no hay una descripción muy específica de lo que trata cada uno de ellos, la idea general la presento a continuación:

Modelo de Control de Acceso Discreto

Un modelo de Control de Acceso Discreto (CAD) se identifica porque el acceso se basa principalmente en la identidad del individuo y el objeto al que quiere tener acceso. Las reglas se definen con base en estos dos datos y en algunas ocasiones se permite al usuario determinar el tipo de control que tiene sobre ciertos recursos.

Entre las primeras implementaciones de este modelo se usaron matrices de *usuarios* contra *recursos* y en cada entrada de la matriz se colocaban los permisos asignados al usuario. Evidentemente esta no era la forma de lograr un desempeño eficiente: el aprovechamiento de espacio en memoria no figuraba entre sus cualidades (el tamaño de la matriz y la información redundante muestran sus defectos).

Es así que se buscó otro tipo de implementación, como las Tablas de Autorización en donde las entradas no vacías de la matriz se colocan en una tabla con tres columnas que corresponden a los sujetos, acciones y objetos. Otra opción para el CAD más conocida es la de listas, llamada *Lista de Control de Acceso* (LCA) en donde cada recurso tiene asociada una lista de usuarios y cada uno de éstos apunta a la lista de permisos que posee para acceder al recurso.

Entre las complicaciones que se desatan en la representación con la LCA destaca principalmente la ineficiencia en las búsquedas. Para obtener *"todos los*

recursos a los que tiene acceso la persona X" se tiene que revisar cada lista relacionada a cada recurso y luego revisar si el usuario X tiene algún permiso sobre éste. Un ejemplo gráfico para la presentación de este modelo se ve en la Figura 1.1.

USUARIO	PERMISO	OBJETO
Edgar	lectura	Archivo1
Edgar	escritura	Archivo1
Ramiro	ejecución	Archivo2
Josué	lectura	Archivo2
Josué	escritura	Archivo2
Josué	ejecución	Archivo1

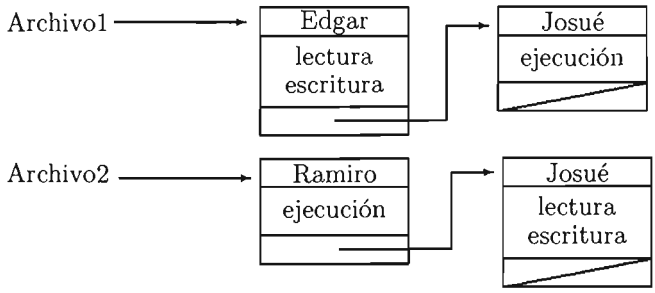


Figura 1.1 Tabla de Autorización y LCA

Modelo de Control de Acceso Obligatorio

Para el modelo de Control de Acceso Obligatorio (CAO) las reglas son determinadas por una autoridad central. La presentación más común de este modelo se distingue por la asignación de niveles de seguridad tanto a usuarios como a objetos.

Por ejemplo, sean *Secreto* (S) y *Público* (P) dos niveles de seguridad en donde $S > P$, es decir, tiene mayor relevancia el nivel S que el P. Si *Computación* y *Biología* son dos áreas que reflejan cierta funcionalidad entonces la jerarquía obtenida con estos datos se ve como en la Figura 1.2. Esta estructura está determinada por el orden dado a los niveles de seguridad y las combinaciones que se pueden dar entre las áreas establecidas; ahí se considera de mayor importancia el nivel colocado en la parte superior de la jerarquía. De esta forma se muestra qué tan sensible puede ser la información para así poder asociar a cada nivel con un grupo de permisos.

En general no es conveniente mantener una autoridad central manipulando todos los accesos a los recursos porque resulta muy complejo realizar cambios y no proporciona confiabilidad.

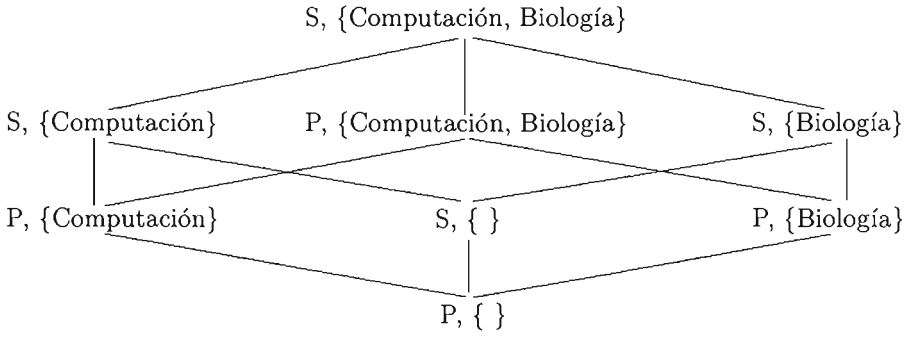


Figura 1.2 Modelo CAO

Modelo de Control de Acceso Basado en Roles

El modelo de Control de Acceso Basado en Roles (CABR) es un modelo de seguridad basado en reglas determinadas en función de los roles indicados para el sistema. Los roles, que ya están establecidos en cualquier organización, reflejan la actividad y responsabilidad en la empresa. Así, el acceso a los objetos está especificado por el papel que juega cada usuario en la organización. Dado que para el propósito del control de acceso es más importante conocer las responsabilidades que tiene el usuario a saber quién es la persona que realiza las operaciones, se propuso este modelo.

Sin embargo, este modelo no cubre todos los escenarios ya que en algunos casos es importante conocer la identidad del usuario que hace la petición y no mantenerla a un lado. Por ejemplo, en el caso de un hospital puede haber una regla que diga: *"el doctor puede recetar al paciente"*, pero no cualquier doctor puede recetar a cualquier paciente ya que cada paciente tiene un doctor asignado; así, es necesario identificar al doctor que le corresponde a cada enfermo para que pueda darle una receta.

El modelo basado en roles se estima como una forma de control flexible y madura, además de que, en comparación con otros modelos, se considera menos vulnerable.

Este trabajo está orientado al control de acceso con este modelo por lo que mostraré más aspectos de este tema a continuación.

1.1.4. Control de Acceso Basado en Roles

El modelo CAD en general se puede tratar como un modelo en el cual un individuo puede establecer las reglas de control de acceso para un conjunto reducido de recursos. En cambio, el CAO es un modelo en el cual la seguridad es asignada por una entidad externa. El CABR es independiente de los dos anteriores pero puede implementarse con tendencias hacia CAD o CAO.

El modelo CABR asocia a cada usuario de un sistema con uno o varios roles dentro de la organización en que trabaja. La relación entre el usuario y su rol se hace con base en sus funciones de trabajo y a las responsabilidades que le fueron asignadas. Además, a cada rol le es asociado un conjunto de permisos. A esta agrupación de privilegios, Baldwin les llama NPD (*Named Protection Domain*) [18].

Los conjuntos NPD están relacionados con los roles (a cada rol le corresponde un NPD), pero Baldwin también hace relaciones entre NPDs, lo que resulta en cadenas de privilegios. Los usuarios pueden acceder a los objetos activando los NPD que tengan asignados (indirectamente o no) pero están restringidos a activar sólo un NPD a la vez. Esta visión de Baldwin refleja mucho el fundamento del CABR que en general se tiene.

Un rol es una construcción semántica para esta política de acceso y se considera estable porque no cambia frecuentemente. Un rol representa competencia en una tarea específica que engloba autoridad y responsabilidad y debe reflejar la asignación de deberes.

El Control de Acceso se refleja en varios componentes del CABR, principalmente en las relaciones de *usuario-roles* y *permisos-rol*, las cuales muestran, a final de cuentas, los permisos que tiene asociado un usuario dentro del sistema. Es decir, las relaciones de *"asignación de usuarios"* y *"asignación de permisos"* son las que determinan el control de acceso del sistema. Ver Figura 1.3.

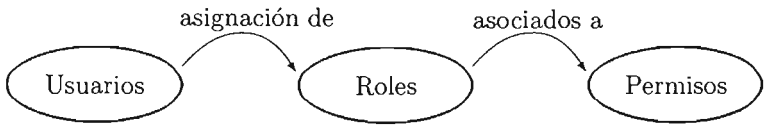


Figura 1.3 Relaciones en el CABR

Principios

Este modelo soporta tres principios de seguridad conocidos:

- Menor privilegio,
- separación de deberes y

- abstracción de datos.

El menor privilegio se refiere a asignar sólo los permisos necesarios a un rol. La separación de deberes trata de impedir el desempeño de un rol a menos que sea necesario para la realización de alguna actividad. Finalmente la abstracción de datos permite el manejo de estos en forma general; por ejemplo, fuera de los permisos comunes como la *lectura* y *escritura* se pueden necesitar otro tipo de permisos como pudiera ser el caso de *creación de cuenta* para una institución bancaria.

Diferencias entre un rol y un grupo

Principalmente el rol es un intermediario, entre los permisos y los usuarios, que provee control sobre la configuración del acceso al relacionar al usuario con los permisos que le corresponden. Así, un rol trae consigo una agrupación de usuarios junto con una colección de permisos mientras que un grupo sólo trae consigo el significado de conjunto.

Los grupos de usuarios pueden ser usados para implementar roles pero sus diferencias son demasiado grandes para confundirlos. Una diferencia semántica muy notable es que los roles pueden ser "activados" y "desactivados" a discreción de los usuarios, mientras que las membresías a los grupos no; es decir, un usuario puede decidir si jugar un rol o no, pero no puede decidir si permanece a un grupo o no.

Administración

Para el control del sistema es necesario al menos un administrador de éste, el cual se encargará de la asignación de permisos a los roles o señalar los roles que corresponden a las personas. El determinar este tipo de aspectos es responsabilidad de un pequeño grupo de la empresa que es indicado como personal de confianza. Un administrador también puede cambiar las reglas que gobiernan el control de acceso. A este control se le llama *política de administración* o *administrativa*.

Propagación de Privilegios

No se tiene aún un modelo que sea aplicable a todo tipo de situaciones y requerimientos; por ejemplo el CABR puede tener algunas complicaciones como en la siguiente situación:

> *"El jefe tiene mucho trabajo y pide a la secretaria que lo ayude redactando algunos oficios. La secretaria necesita obtener información (momentáneamente) que no tiene permiso de ver, pero su jefe sí. De esta forma, la secretaria necesitará de algunos privilegios de manera temporal".*

Este hecho indica que en algunas ocasiones puede surgir la necesidad de que los usuarios cedan algunos de sus privilegios a otras personas (lo que implica que también habría que tener un control sobre los roles de tal forma que no fuera posible que por descuido de un usuario, otro individuo se aprovechara del privilegio otorgado). Sin embargo, hay otras formas de propagación de privilegios, en el ejemplo anterior, donde sólo el privilegio es propagado a los niveles conectados cercanamente.

Ventajas del Control de Acceso Basado en Roles

Este modelo tiene la ventaja de simplificar la obtención de respuestas a preguntas como: *¿qué permisos tiene la persona X?* o *¿qué personas tiene el permiso Y?* lo que en otros modelos resultaría muy costoso (como en el CAD). La ventaja principalmente surge de agrupar los permisos en los roles.

El control de la membresía a los roles y a los permisos que son asignados a un rol, se encuentran entre las principales ventajas del modelo CABR. Esto se debe a que al reasignar un puesto a un individuo sólo basta con cambiar al rol que pertence para que, de esta manera, cuente con los permisos que le corresponden a su nuevo nivel [19]. Esto facilita la administración y la revisión del control de acceso así como el mantenimiento del sistema.

Los cambios de puestos en una empresa son medianamente frecuentes pero los privilegios que tiene un rol son aún menos recurrentes por lo que se vuelve una opción bastante aceptable utilizar este modelo.

Otras características favorables del CABR son las siguientes:

- La asignación de roles a usuarios es independiente de los permisos asignados a cada rol.
- La jerarquía de roles (si es que la hay) puede ser explotada para realizar implicaciones de permisos (aunque algunas veces no sea deseada esta parte).
- Se pueden agregar restricciones a las propiedades de cada uno de los conjuntos que se manejan (usuarios, roles o permisos) o a las relaciones que hay entre ellos.

Estudios del NIST[1] indican que los permisos a roles cambian poco y que los sectores comerciales y de gobierno utilizan mucho este tipo de control por lo que es un motivo más para utilizar este modelo.

[1] El NIST (*National Institute of Standards and Technology*) es un instituto norteamericano que tiene como misión desarrollar y promover sistemas de medidas, estándares y tecnología para aumentar la productividad, facilitar las transacciones y mejorar la calidad de vida. Para más información ver http://www.nist.gov/

Desventajas del Control de Acceso Basado en Roles

El CABR no puede controlar "sucesiones de operaciones o eventos" como otros modelos [20]. Por ejemplo, el proceso de titulación en la Facultad de Ciencias consta de varios pasos antes de obtener el título pero el control sobre la realización u orden de ellos no está bajo la regulación del CABR.

Este modelo funciona bien en casos donde hay grupos de personas que tienen actividades en común; sin embargo si fuera necesario mantener algunos permisos a cada usuario de forma particular, el modelo que he presentado tendría que ser modificado (por ejemplo, agregando un rol para una sola persona o haciendo otro tipo de actos catastróficos como ése) para lograr un comportamiento como el deseado. Por lo que es importante considerar si el CABR debe ser aplicado para cada situación.

A continuación presento el siguiente tema esencial que abarca este trabajo.

1.2. Web Semántico

El Web Semántico es una de las ideas iniciales dentro de la creación del Web que propuso Tim Berners-Lee pero que fue dejada atrás momentáneamente. Esta idea destaca principalmente porque da inicio a un ciclo en que la información no sólo tiene que ser entendida por el hombre, sino también, de alguna manera, por la computadora.

1.2.1. ¿Qué es el Web Semántico?

El Web Semántico se ha introducido para automatizar, reusar e integrar datos a través de diferentes aplicaciones. La definición dada por Tim Berners-Lee, James Hendler y Ora Lassila [4] para el Web Semántico es la siguiente:

> *"El Web Semántico es una extensión del Web actual en el que a la información se le da un significado bien definido, habilitando a las computadoras y gente a trabajar en cooperación."*

Hacer más fácil la relación entre humanos y computadoras es una de las cosas que han sido perseguidas desde que las personas tuvieron que interactuar con ellas. Así que tiene mucho sentido mejorar este vínculo.

Antes mencioné que *"la información debe ser entendida por la computadora"*, con lo que me refiero a que los lenguajes utilizados para expresar la información no sólo deben estar en una forma procesable para la máquina, sino que debe haber una manera de proveer el significado para ellos. Una definición dada por Yagüe *et al.* ([5]) al respecto es la siguiente:

> *"Un lenguaje puede ser* entendido *por una computadora si tiene una semántica asociada: los símbolos y estructuras del lenguaje deben referirse a un modelo subyacente (porque el significado sólo existe si está en relación con algo)."*

El mapa a seguir para alcanzar un Web Semántico se muestra en la Figura 1.4. Este mapa fue indicado por T. Berners-Lee y aunque algunos aspectos de él no han sido completamente aclarados [9], es la guía utilizada para llegar a ese objetivo.

Los niveles en la base de la pirámide de la Figura 1.4 muestran a los modelos de datos y tecnologías (Unicode, URI, XML, RDF y RDFS) que deben proveer de una sintaxis formal que permita el proceso automático del contenido. En especial, modelos de datos como RDF y RDFS deben proporcionar vocabularios que describan los conceptos de varios dominios y así permitir a agentes de software y humanos, compartir información y conocimiento. A estos vocabularios se les conoce como *ontologías*.

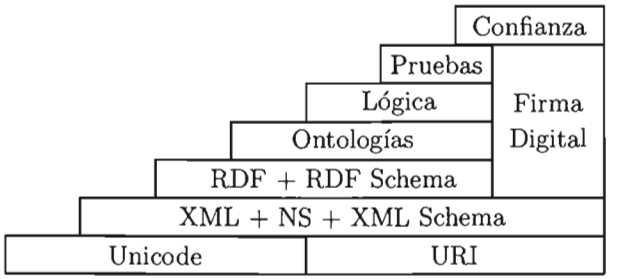


Figura 1.4 Arquitectura del Web Semántico

El Web Semántico intenta proveer el significado de los recursos de forma accesible para la computadora; y para lograr esto empieza asignando a cada recurso un identificador, el cual será un *Identificador Universal de Recursos* (URI, Uniform Resource Identifier). Dado que XML no tiene una semántica formal para la computadora, pero provee de flexibilidad e interoperabilidad entre diferentes aplicaciones y sistemas, es ocupado como una base de nuestras tecnologías que sí cubren ese aspecto, como lo son RDF y RDFS, entre otras. RDF define una convención sintáctica y un modelo de datos simple para representar la semántica de la información del Web de forma en que sea legible para la máquina; RDF Schema (RDFS) define modelos primitivos básicos sobre RDF.

Con RDF se puede describir cualquier recurso del Web y provee interoperabilidad entre aplicaciones. Además, ahora se considera que cualquier lenguaje de ontología debe cumplir con tres requerimientos importantes: ser intuitivo para

el uso humano; tener una semántica formal bien definida; y por último, debe tener una estrecha relación con lenguajes ya existentes como XML y RDF para asegurar la interoperabilidad.

Los siguientes niveles de la pirámide se refieren a procesos de inferencia y validación en los cuales se razona (es decir, involucra derivar ciertos valores) sobre ejemplares de una ontología. La inferencia puede ser usada para obtener datos sobre conocimiento implícito o explícito especificado en un vocabulario. Adicionalmente cada ontología debe tener una semántica asociada.

De esta manera la arquitectura del Web Semántico está basada en un gran número de tecnologías, aunque algunas de ellas aún están en desarrollo (sobre todo las capas superiores). Así, primero se describen lenguajes que permiten agregar la semántica al Web y que se colocan en la base de su arquitectura. Luego, se construyen herramientas para su procesamiento. Y, finalmente, se construyen aplicaciones que utilizan su potencial.

1.2.2. Ontologías

La definición dada por la W3C[2] Web Ontology Working Group Charter es:

> *"Una ontología define los términos usados para describir y representar un área del conocimiento."*

Su prometedor objetivo es el de compartir y reusar el conocimiento de algún dominio en particular, para ser comunicado entre personas y sistemas. Para Felman [6] una ontología es una especificación formal explícita (los conceptos y definiciones están explícitamente definidos) de una conceptualización (el modelo abstracto de un fenómeno del mundo) compartida (capturada de un conocimiento consensuado). Aunque existan diferentes conceptos de ella todos giran alrededor del mismo significado, de la misma semántica.

A final de cuentas, se busca obtener un lenguaje que exprese información legible para una computadora. Adicionalmente este tipo de descripción ofrece, a quien la adopte, una forma de definir su propia visión de cualquier área del conocimiento.

La conveniencia de estar disponible a todo el mundo facilita la distribución de información, lo que trae como consecuencia el poder de extender o utilizar una ontología ya antes escrita.

[2] El *World Wide Web Consortium* es una organización que desarrolla tecnologías para llevar al Web a su máximo potencial. Para más información ver http://www.w3.org/

Hay un gran número de grupos que han desarrollado lenguajes para la descripción de meta-datos, pero los que han resaltado y se han convertido en estándares son RDF y OWL. Aquí se trabaja solamente con RDF.

1.2.3. RDF

Una forma para la descripción de las ontologías y fuentes de información es por medio de RDF (*Resource Description Framework*). Las etiquetas de XML tienen asociado un significado semántico para los humanos, pero para las computadoras, tal significado es muy restringido. Así, se han desarrollado modelos de datos para la descripción de esta información, entre ellos RDF.

RDF es un lenguaje para la presentación de información acerca de los recursos en el Web, particularmente, para la representación de meta-datos acerca de esos recursos. Incluso se pueden representar conceptos que no pueden ser mostrados en el Web como el concepto de *persona*, por ejemplo.

RDF provee de un marco de trabajo común para expresar información y para que pueda ser intercambiada entre aplicaciones sin pérdida de datos. Está basado en la idea de identificar cosas usando los identificadores del Web llamados URIs (*Uniform Resource Identifiers*) y describiendo recursos en términos de objetos, propiedades y valores de propiedades. Esto permite representar enunciados acerca de los recursos como una gráfica de nodos y arcos etiquetados, que representan el *objeto* (también conocido como *recurso*), su *propiedad* y su *valor* (ver Figura 1.5).

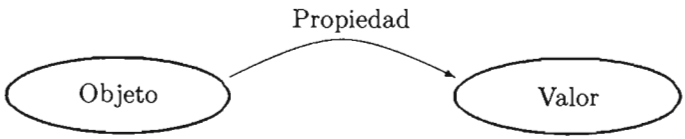


Figura 1.5 Representación gráfica de la descripción de datos en RDF

Ejemplo 1: Para decir *"La página http://www.ejemplo.org/indice.html fue creada por Luis el día 31 de febrero"* se representa como se muestra en la Figura 1.6.

Obsérvese que para indicar los recursos se usa un óvalo, mientras que para un valor que no tiene un URI que lo identifique se utiliza un rectángulo; independientemente de esto la representación sigue siendo una gráfica. El URI indica la ubicación de la ontología y por lo tanto el lugar donde se encuentra el concepto que se necesita (*fecha-creacion* y *creator*). Luis no es un recurso propiamente del Web así que se utiliza un identificador asignado únicamente a él y el cual es *http://www.ejemplo.org/ids/luisID*.

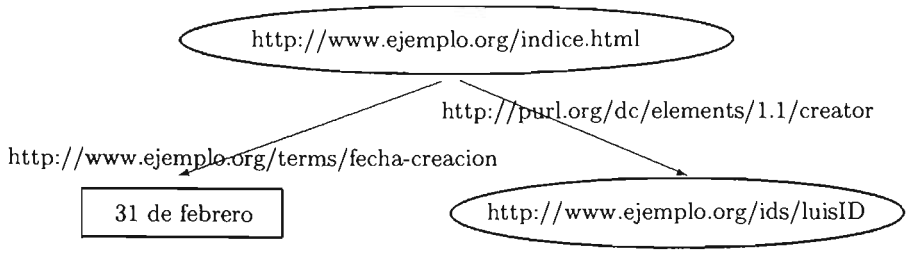


Figura 1.6 Ejemplo "La página http://www.ejemplo.org/indice.html fue creada por Luis el día 31 de febrero"

RDF provee también un formato de serialización basado en XML (llamado RDF/XML) para registrar e intercambiar estas gráficas. La gráfica de la Figura 1.6 corresponde en RDF/XML a lo siguiente:

```
1.  <?xml version="1.0"?>

2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:dc="http://purl.org/dc/elements/1.1/"
4.           xmlns:ejterms="http://www.ejemplo.org/terms/">

5.    <rdf:Description rdf:about="http://www.ejemplo.org/indice.html">
6.     <ejterms:fecha-creacion>31 de febrero</ejterms:fecha-creacion>
7.     <dc:creator rdf:resource="http://www.ejemplo.org/ids/luisID"/>
8.    </rdf:Description>

9.  </rdf:RDF>
```

Los *espacios de nombres* como *rdf*, *dc* o *ejterms*, (líneas 2, 3 y 4), permiten separar conceptos con diferentes significados pero que se escriben de igual forma; además proveen de una forma más accesible de la escritura. También se usan para otro tipo de representación: por *tripletas*. El Ejemplo 1 también se escribe como a continuación muestro:

```
ej:indice.html  ejterms:fecha-creacion  ''31 de febrero''
ej:indice.html  dc:creator              ejids:luisID
```

En donde se debe entender:

ej	como	http://www.ejemplo.org/
ejterms	como	http://www.ejemplo.org/terms/
dc	como	http://purl.org/dc/elements/1.1/
ejids	como	http://www.ejemplo.org/ids/

Nótese que las tripletas tienen el orden de: objeto, propiedad, valor.

Con este lenguaje cualquiera puede definir una nueva ontología, extender alguna o utilizar una o varias definidas previamente (esto último correspondería a crear un ejemplar de una ontología). Un ejemplo de definición de una nueva ontología es el siguiente:

```
<?xml version="1.0"?>
  <rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

    <rdf:Description ID="RFC">
      <rdfs:label>Registro Federal de Causantes</rdfs:label>
    </rdf:Description>
  </rdf:RDF>
```

RDFS

RDF provee una forma de expresar enunciados sobre recursos, usando propiedades y valores. Sin embargo, las comunidades de usuarios necesitan definir vocabularios con el propósito de describir tipos específicos de recursos con propiedades específicas. RDF por sí mismo no tiene medios para definir tales *clases* específicas para cada aplicación así como sus *propiedades*, pero a la extensión que proporciona este vocabulario se le conoce como RDFS (*RDF Schema*).

Una clase en RDF Schema corresponde a un concepto genérico de una categoría. Las clases RDFS pueden ser usadas para representar casi cualquier categoría de cosas, como documentos, personas, bases de datos o conceptos abstractos. Las clases se describen principalmente usando el recurso de RDFS *rdfs:Class* y las propiedades *rdf:type* y *rdfs:subClassOf*.

Supongamos que en una escuela (identificada por *www.escuela.edu*) se quiere usar RDFS para describir información sobre las materias que se impartirán el siguiente semestre. La escuela necesita una clase que represente la categoría de cosas conocidas como Materias y cada recurso que forme parte de esa clase se le llamará *ejemplar* (del inglés *instance*). Cada clase es un recurso que tiene como propiedad *rdf:type* y cuyo valor es *rdfs:Class*, con rdf y rdfs los espacios de nombres definidos como:

```
http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

y

```
http://www.w3.org/2000/01/rdf-schema#
```

respectivamente. Así que la clase Materia se describe así:

```
ej:Materia    rdf:type    rdfs:Class
```

con *ej* indicando la referencia al URI *http://www.escuela.edu/esquemas/cursos/*. De esta forma, Álgebra es una de las materias impartidas en el curso entonces se puede expresar de la siguiente manera:

```
ejemplo:Álgebra    rdf:type    ej:Materia
```

con *ejemplo* indicandola *http://www.escuela.edu/esquemas/cursos2004/*.

Para definir una subclase se realiza lo siguiente. Supóngase que Seminario es una subclase de Materia; para indicarlo se hace lo siguiente:

```
ejemplo:Seminario    rdf:type           rdfs:Class
ejemplo:Seminario    rdfs:SubClassOf    ej:Materia
```

Otro punto importante es la descripción de propiedades que caracterizan a las clases. En RDF Schema éstas se describen usando la clase *rdf:Property* y las propiedades *rdfs:domain* y *rdfs:range*.

Todas las propiedades se describen como ejemplares de la clase *rdf:Property*. Por ejemplo, cada materia tiene como propiedad un número de créditos, y para declarar esto como la propiedad se hace lo siguiente:

```
ej:creditos    rdf:type    rdf:Property
```

RDFS provee de un vocabulario para describir cómo las propiedades y las clases son destinadas a usarse con datos. La propiedad *rdfs:range* es usada para indicar que los *valores* de una propiedad particular son ejemplares de una clase designada. Por ejemplo, para indicar que la propiedad de *ej:profesor* tiene valores que son ejemplares de *ej:Persona* se escribe:

```
ej:Persona    rdf:type      rdfs:Class
ej:profesor   rdf:type      rdf:Property
ej:profesor   rdfs:range    ej:Persona
```

Estos enunciados indican que *ej:Persona* es una clase, *ej:profesor* es una propiedad y que los enunciados RDF que usan la propiedad *ej:profesor* deben tener ejemplares de *ej:Persona* como valores en una tripleta.

La propiedad *rdfs:domain* se usa para indicar que una propiedad particular sólo se puede aplicar a una clase particular. Por ejemplo, si para *www.escuela.edu* la propiedad *ej:autor* se aplica a ejemplares de la clase *ej:Libro* entonces tenemos los encunciados RDF siguientes:

```
ej:Libro    rdf:type      rdfs:Class
ej:autor    rdf:type      rdf:Property
ej:autor    rdfs:domain   ej:Persona
```

Estos enunciados indican que *ej:Libro* es una clase, *ej:autor* es una propiedad y que los enunciados RDF que usan la propiedad *ej:autor* tienen ejemplares de *ej:Libro* como objetos en una tripleta.

Nodos en blanco

RDF representa sólo relaciones binarias, así que para expresar una relación con diversos componentes es necesario dividirla en grupos de este tipo de relaciones. Para esto se utilizan los *nodos en blanco* que representan conceptos derivados de definiciones más generales. Para cada relación n-aria uno de los participantes es elegido como *sujeto* de la relación y un nodo en blanco es creado para representrar el resto de ella; los participantes restantes son entonces representados como propiedades separadas del nuevo recurso representado por un nodo en blanco [11].

Por ejemplo, si Luis (del Ejemplo 1, Figura 1.6) tiene la dirección *"Agua Naval No. 6, Michoacán"* su representación sería:

```
ejids:luisID   ejterms:dirección   ''Agua Naval No. 6, Michoacán''
```

pero para dividir los datos de la dirección de Luis en una estructura consistente en valores separados como: calle, número y ciudad, es necesario agregar un recurso extra y hacer enunciados sobre él. Su representación gráfica se ve como en la Figura 1.7 mientras que en tripletas es:

```
ejids:luisID     ejterms:dirección     ????
????             ejterms:calle        ''Agua Naval''
????             ejterms:número       ''6''
????             ejterms:ciudad       ''Michoacán''
```

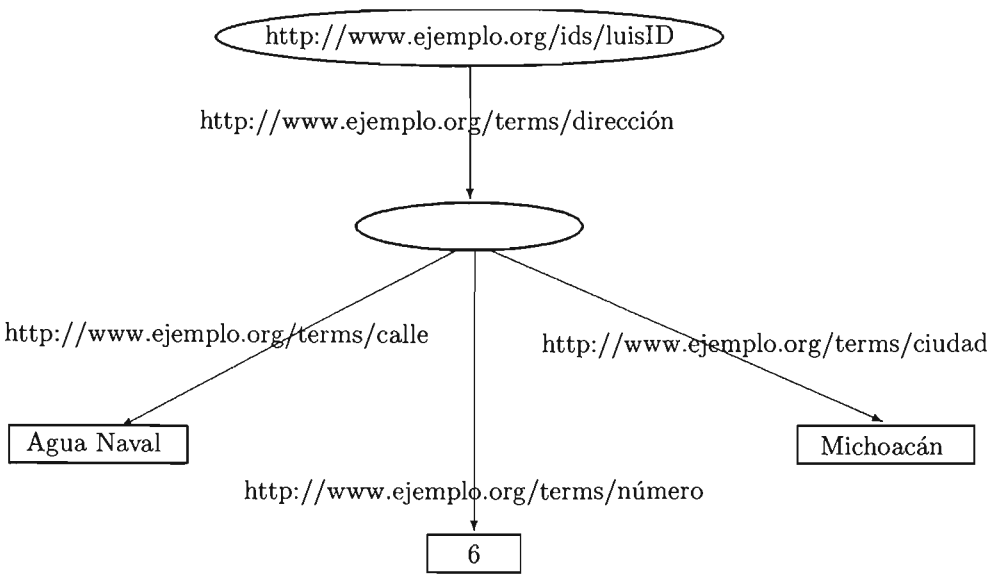


Figura 1.7 Ejemplo del uso de un nodo en blanco.

El recurso marcado con ???? indica la presencia de un nodo en blanco y que se maneja como *anónimo* pues sólo tiene el propósito de conectar las diferentes partes de la relación que expresa la dirección de una persona. Una gráfica compleja puede contener varios nodos en blanco y para diferenciarlos en una representación de tripletas se deben de identificar de la forma

_:nombre

en donde *nombre* es un identificador dado al recurso en blanco al que se representará. Así, en el caso anterior las tripletas que resultan son:

```
ejids:luisID        ejterms:dirección     _:dirLuis
_:dirLuis           ejterms:calle         ''Agua Naval''
_:dirLuis           ejterms:número        ''6''
_:dirLuis           ejterms:ciudad        ''Michoacán''
```

Los nodos en blanco corresponden al objeto o al valor de un enunciado, pero nunca a una propiedad. Además deben ser usados de forma privada en cada gráfica, es decir, no se debe hacer mención a este tipo de nodos fuera de ella ya que sólo son auxiliares para la descripción y no tienen un significado asociado.

Capítulo 2

Control de Acceso y Web Semántico

The thing that fascinated Licklider about the TX-2 was its graphics display...Hypnotised by the possibilities of this, Licklider became hooked on computing and took to spending hours at a time with the machine. He had seen the future - and it worked for him.

- John Naughton, *A Brief History of the Future*

En realidad no hay un método estándar que haga fácil la creación de ontologías y mucho menos alguno que proporcione una forma de agregar semántica a un modelo dado. Aun así se han hecho trabajos que integran ambos temas pero sin especificar cómo lo hacen. Yagüe *et al.* [5], explican brevemente su procedimiento: ellos trabajan con la definición de modelos semánticos sobre los componentes de un modelo de control de acceso dado. De manera similar a esta propuesta, también construyo en este trabajo un modelo a seguir para el control de acceso común (específicamente el basado en roles), y a esta construcción dedico este capítulo.

2.1. Una propuesta para la creación de un Control de Acceso Semántico

Para iniciar en la construcción de un modelo semántico, tomo una de las ideas ya existentes del Control de Acceso y luego agrego la de Yagüe *et al.* A continuación doy una explicación más amplia sobre este punto.

2.1.1. Elección de un modelo de Control de Acceso

En la sección 1.1.1 mencioné tres puntos para añadir el Control de Acceso a un sistema [18]:

1. Política de Seguridad.
2. Modelo de Control de Acceso.
3. Implementación del Modelo.

Por "Política de Seguridad" me refiero al establecimiento de las reglas que se planean seguir, las cuales son proporcionadas por los interesados en el sistema. El "Modelo de Control de Acceso" es un modelo teórico que muestra una representación formal del mecanismo de Control de Acceso a seguir. Finalmente, se requiere implementar lo señalado por el modelo.

Tanto la Política de Seguridad como la Implementación, son dos aspectos dependientes del problema y de la forma en que se realice. Ambos puntos tienen un grado de incertidumbre asociado, ya sea por la especificación dada por el usuario del sistema sobre lo que desea de éste, o por la forma en que será implementado por el programador. Aun así, los dos forman parte esencial de lo que se desea obtener.

El punto que a mi consideración es más importante es el referente al Modelo de Control de Acceso. Éste, además de reflejar la estructura y relaciones del sistema, provee de una visión genérica aplicable a un conjunto de sistemas con un propósito común (o al menos similar) en seguridad.

Hay muchos trabajos sobre modelos que pueden encajar de forma genérica pero simplemente ninguno ha llegado a ser "el modelo". Cada una de las propuestas carece de propiedades que sean adecuadas para todo sistema, sin embargo, algunas de ellas (las de más éxito) han sido estudiadas por varios grupos para obtener una mayor ventaja. No obstante, se han hecho extensiones de aquellos modelos sobresalientes. El modelo que presento, y sobre el cual trabajo, es el CABR (mencionado en las secciones 1.1.3 y 1.1.4).

2.1.2. Idea intuitiva de cómo agregar la semántica a un modelo

Es en el Modelo de Control de Acceso en donde se involucra el Web Semántico. Con esto me refiero a seguir los pasos de la Arquitectura del Web Semántico (Figura 1.4) para que ese sistema, además de mediar la entrada a los recursos, también pueda integrarse a una red de información semántica. Lo que pretendo con esto es integrar al Control de Acceso Basado en Roles las bases del Web Semántico, específicamente a su modelo pero con la precaución de utilizar los

estándares dictaminados por la W3C para el desarrollo de información semántica.

El siguiente problema es el de cómo agregar esta tecnología a un modelo. Yagüe *et al.* trabajan con la definición de modelos semánticos sobre los componentes de un modelo de control de acceso (aunque el modelo que trabaja es para sistemas distribuidos heterogéneos) [5]. Con base en este método trabajaré en una forma similar reemplazando cada componente con uno similar pero que se refiera específicamente a la información semántica del conjunto que le corresponde.

¿Por qué elegir un modelo de ese estilo y no proponer uno nuevo? En principio porque ya hay un ejemplar de éste que parece funcionar bien (el propuesto por Yagüe y sus colegas), pero principalmente por "compatibilidad hacia atrás". Con esto me refiero a la posibilidad de utilizar todo el conocimiento obtenido de los muchos trabajos realizados con respecto a este tema (al concerniente al CABR) y, para aquellos que funcionen bien, "sólo basta" con interpretarlos de una manera distinta para que puedan evolucionar e integrarse de manera natural al Web Semántico.

Adicionalmente, es de especial interés que la integración de ambientes heterogéneos sea de forma inmediata y tal que la semántica de la información no se vea afectada. Así, el Web Semántico proporciona esta cualidad para Control de Acceso al presentar sus datos con la semántica que se les ha asociado.

2.2. Modelo Semántico fundado en el Control de Acceso Basado en Roles

2.2.1. Modelo Base

Un modelo básico, que cubre una gran cantidad de familias de modelos del Control de Acceso Basado en Roles, propuesto por Sandhu *et al.* se muestra en la Figura 2.1. Se presentan los conjuntos principales del CABR [20]: usuarios, roles y permisos; así como las relaciones más importantes entre ellos: la asignación de roles a usuarios y la asignación de permisos a roles.

Los conjuntos y relaciones representan la concepción fundamental del CABR. Sin embargo es necesario indicar otras características que reflejan una situación real; éste es el caso del grupo que representan las sesiones y la presentación de medidas administrativas.

Cada sesión se mapea a un usuario, el cual activa un subconjunto de roles. El concepto de sesión se equipara en el modelo con la noción tradicional de *sujeto* en la literatura de Control de Acceso [20]. Así, un sujeto se trata como un proceso que se ejecuta a nombre de la persona real y que se encarga de enviar

las peticiones al sistema [18].

Los permisos para modificar los conjuntos y relaciones fundamentales del CABR se llaman *Permisos Administrativos.* Para estos permisos es necesario contar con un conjunto de roles que reflejen la responsabilidad que está sobre ellos, y para ello se presenta el conjunto de *Roles Administrativos.* De igual forma que con los roles y los permisos para los usuarios comunes, se agrega la relación *Asignación de Permisos Administrativos.*

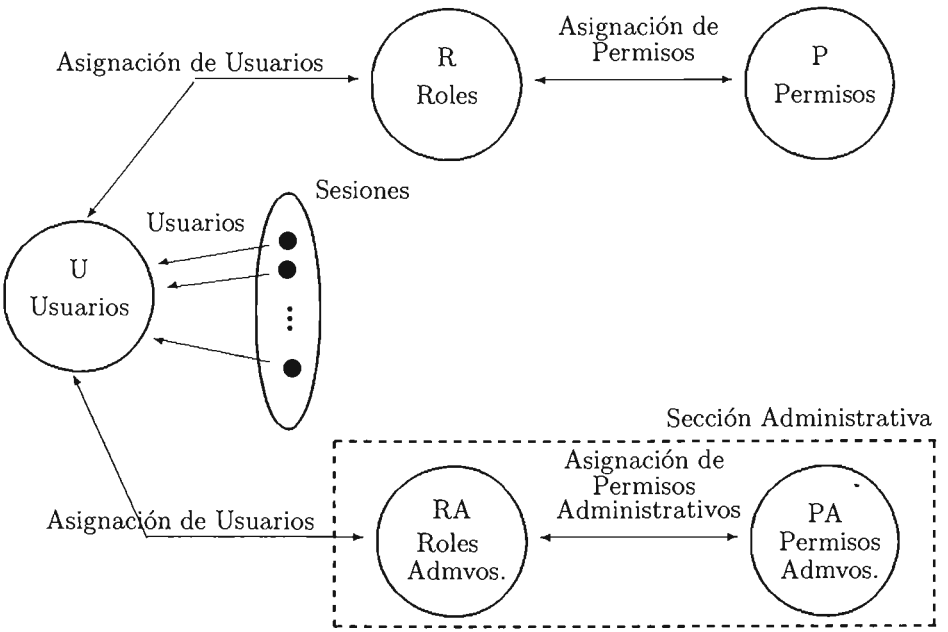


Figura 2.1 Modelo Base (Sandhu *et al.*)

La Sección Administrativa es importante para separar los permisos correspondientes a la funcionalidad del sistema de las tareas administrativas concernientes a él. Esta sección debe ser manejada por individuos confiables. Otra ventaja de esta división es que se permite que los administradores puedan manejar las relaciones y conjuntos del control de acceso, pero a la vez no puedan usar ninguno de los privilegios ellos mismos.

2.2.2. Extensiones al Modelo Base

Jerarquías y Herencia

El Modelo Base de Sandhu sólo introduce las características fundamentales del CABR. Sin embargo, es común encontrar jerarquías dentro del conjunto de

roles que reflejen la organización, autoridad o responsabilidad que juega cada individuo.

Para el control de accceso común, añadir jerarquías al modelo implica considerar la posibilidad de transferir o compartir permisos. En una estructura jerárquica de roles se refleja en gran parte la forma de asignación de permisos a cada rol por lo que se puede tomar en consideración tal orden para ceder o negar las autorizaciones a roles de otros niveles. Algunos ejemplos de estas concesiones son: el acumular permisos de los roles que están por abajo (o por arriba) de la posición de un rol con respecto al orden señalado por la jerarquía, compartir permisos entre los roles cercanos, hacer inferencias de permisos en base a la jerarquía, incluir permisos transitivos o herencia de permisos.

Una forma para representar de manera más precisa la jerarquía de roles, y a su vez limitar el alcance de la herencia es el establecimiento de *roles privados.* Un rol privado comparte algunos permisos y responsabilidades de un rol dado pero también tiene propiedades muy particulares a su actividad y nadie más comparte sus características privadas. En una jerarquía de roles se muestra como un elemento sin elementos por encima de él [19]. En la Figura 2.2 se muestra un ejemplo de roles privados: un Supervisor de Proyecto puede tener los mismos permisos que un Programador, pero hay información que sólo este último desea ver (como los módulos aún no terminados del sistema) y que no son convenientes presentar al Supervisor del Proyecto aún; lo mismo pasa con el Ingeniero de Pruebas (Ing. de Pruebas).

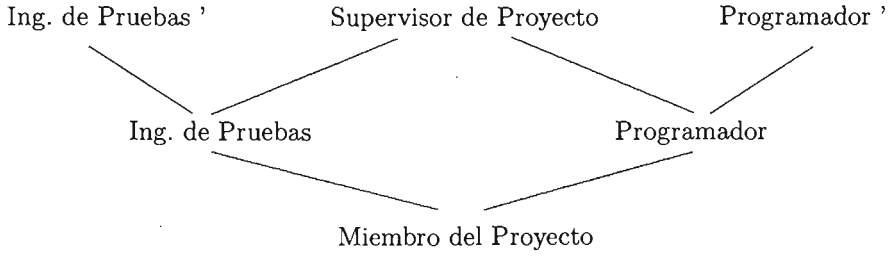


Figura 2.2 Ejemplo de Roles Privados

Así, el agregar jerarquías al CABR implica mantener un orden adecuado en la organización de los roles para contar con una mejor distribución de los privilegios, en caso de que se desee utilizar la jerarquía para precisar la asignación de permisos.

Restricciones

Algunos toques particulares de cada sistema son indispensables para la funcionalidad que se requiere, en especial las restricciones que se efectuarán sobre él. Este tipo de características se puede aplicar en todas las relaciones del Modelo Base. Las más comunes tratan sobre la cardinalidad de los miembros de un conjunto, otras se refieren a la definición de roles disjuntos (roles que no pueden ser impartidos por una misma persona) y algunas más restringen las operaciones de los roles administrativos.

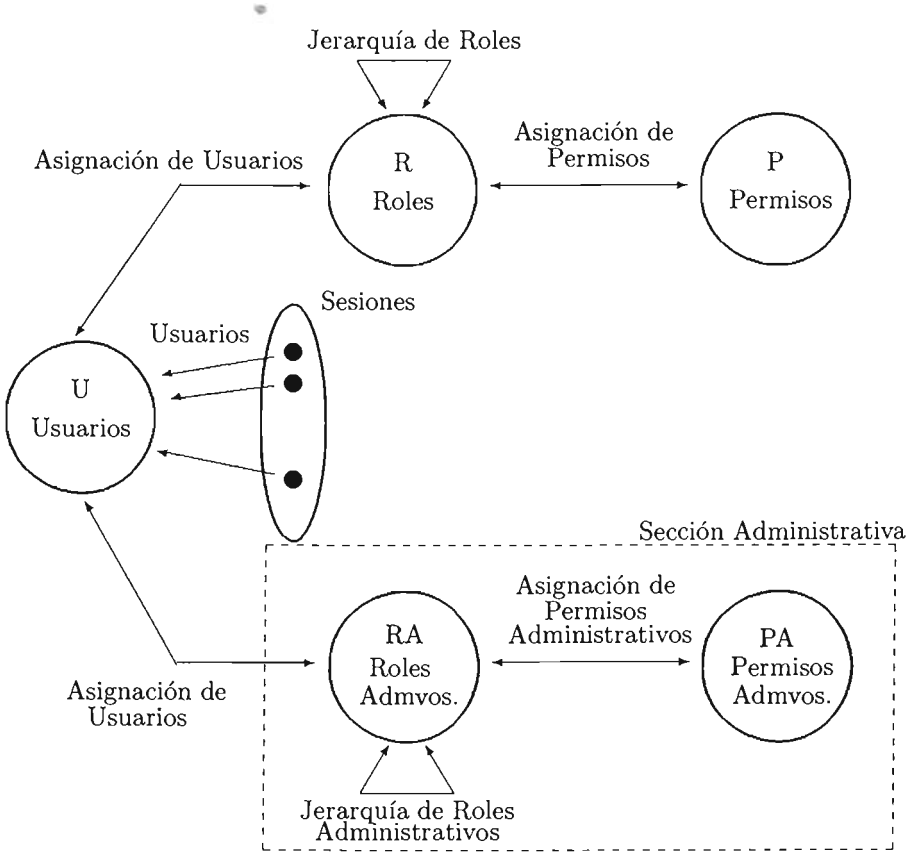


Figura 2.3 Modelo Básico Extendido

Modelo Básico Extendido

En las relaciones de trabajo, en general, el cargo de una persona tiene algunas propiedades: la jerarquía del puesto permite estar al mando de alguna tarea o indicar permanecer bajo las órdenes de alguien con un puesto mayor. Debido a esto, es importante agregar la relación que provee esta característica al Modelo Base de la Figura 2.1. A este modelo lo llamaré Modelo Básico Extendido, el

cual queda como se ve en la Figura 2.3.

2.2.3. Incorporación de la semántica al Modelo Básico Extendido

Un modelo semántico creado a partir de la definición semántica de cada uno de los diferentes componentes del Modelo Básico Extendido, no sólo debe considerar la descripción de sus grupos y asociaciones, también hay que tomar en cuenta toda la semántica de su contexto. Es decir, el agregar la semántica al modelo, debe tomar en cuenta dos aspectos:

1. Especificaciones generales (indicadas en este caso por el Modelo Básico Extendido).

2. Especificaciones de los conceptos y relaciones implícitas del problema.

El punto uno involucra presentar la semántica de cada uno de los componentes del modelo. El punto dos contempla el escenario general en que se desarrolla el problema; como los objetos que forman parte del flujo de trabajo cotidiano de la organización y sus características. Con respecto a esto último se debe tomar en cuenta a todos aquellos objetos importantes que se manipulan y, de ser necesario, las relaciones que hay entre ellos (similar a una jerarquía de objetos); además, hay que considerar sus propiedades o estados.

Así que para agregar la semántica es necesario añadir:

1. Los objetos involucrados y la relación entre ellos (si es que existe).

2. Características de los objetos.

2.2.4. Un Modelo Semántico para el Control de Acceso Basado en Roles

A lo largo de esta sección el modelo al que me refiero es de la Figura 2.3, a menos de que indique lo contrario. Por el momento tomaré sólo los conjuntos y las relaciones involucradas directamente con el problema por lo que omitiré el conjunto Sesiones así como su relación con el conjunto Usuarios.

La descripción semántica de cada componente y relación del modelo se logra con la presentación en RDF/XML de su información. Como resultado de este nuevo panorama, se obtiene un conjunto de *categorías* o *colecciones* semánticas. Por ejemplo, los Roles forman una categoría mientras que los Roles Administrativos forman otra y la Asignación de Permisos es otra diferente a las dos anteriores.

Para el nuevo modelo en construcción hay que observar que debido a la naturaleza de su presentación (dentro de un contexto semántico) es posible tratar

agrupaciones de las colecciones formando nuevas. Primero, los conjuntos R y RA pueden describirse en una sola categoría porque tratan de un grupo común, el de los roles; además, es posible identificar los roles administrativos de los comunes por medio de una propiedad que los distinga. Lo mismo sucede con los permisos de los conjuntos PA y P, con ellos se puede crear la categoría *permisos*.

Al igual que con los roles y los permisos, las relaciones entre los conjuntos forman parte de otro tipo de colecciones. La asignación de roles a usuarios, tanto a los administrativos como a los comunes, se coloca en una categoría. Mientras tanto, la asignación de permisos a los roles se reune en otra colección. De forma análoga se trabaja para las jerarquía de roles.

Así, contamos con las categorías de:

- Usuarios.
- Roles.
- Permisos.
- Asignación de Usuarios.
- Asignación de Permisos.
- Jerarquía de Roles.

Estas colecciones aún no son suficientes para tener las características generales del control de acceso de forma semántica, falta obtener la información del ambiente en el que se aplicará el modelo. Así pues, hay que tomar en cuenta todos aquellos objetos que se manejan (como pueden ser cartas, libros, recibos, en fin, todo aquello que sea propio del contexto). Todos estos objetos podrían también tener relaciones entre ellos, esto es, una carta no podía existir sin un borrador antes, por ejemplo. De esto se deriva la existencia de relaciones y propiedades sobre los objetos. Así que dadas las categorías anteriores, se agregan las siguientes:

- Objetos y
- Características de los objetos (como sus propiedades, relaciones o estados)

Nótese que en algunas ocasiones, para otorgar un permiso se tienen que cumplir algunas condiciones, esto es, verificar que las características del rol y del recurso sean apropiadas para que se permita ejecutar una acción. Por ejemplo, el permiso *Puede Demoler Casa* está inducido por la acción *Accion Demoler Casa*, si el rol es de *Arquitecto* y la casa a demoler *Está Construida* entonces se dice que las condiciones están dadas para poder ejecutar la acción (ver Figura 2.4).

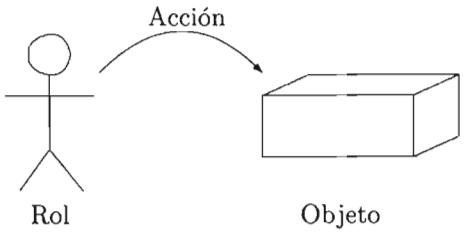


Figura 2.4 Permiso

De esta forma, la relación Asignación de Permisos no es de gran utilidad dado que las asignaciones de los permisos a los roles están determinadas por las condiciones que se deben cumplir para cada acción. Debido a las observaciones hechas se tiene como consecuencia que dada una acción a realizar, hay que verificar si las condiciones necesarias para efectuarla (es decir, el rol y el objeto) se dan, para que el permiso sea otorgado. Ver Figura 2.5.

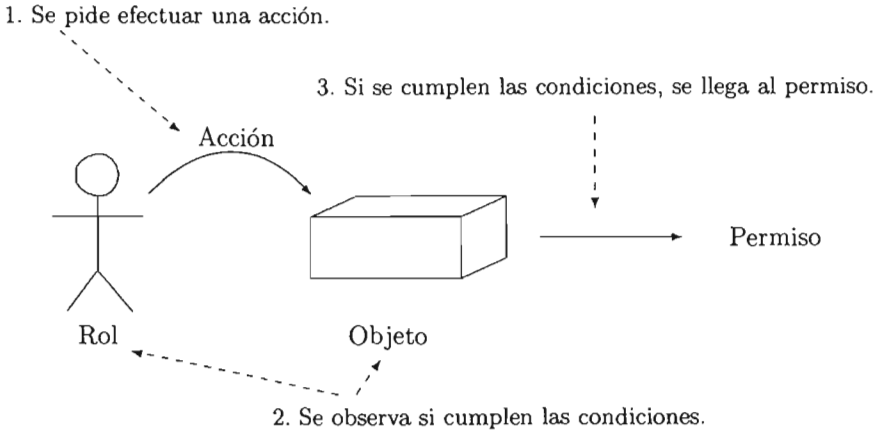


Figura 2.5 Obtención de un permiso

Con base en el Modelo Básico Extendido, el nuevo modelo a crear se ve afectado principalmente en la relación *Asignación de Permisos* y debe ser aumentado para tomar en consideración también el concepto de acción. A partir de esto surge la categoría de *acciones* y *permisos*.

Finalmente tenemos las siguientes categorías, las cuales describen no solamente un tipo reducido del modelo actual (lo que nos ayuda a utilizar un diagrama menos complicado), sino también sugieren una forma de implementarlo (en sí, el Modelo Básico Extendido resulta ser una representación de la noción principal del control de acceso común pero repetida: una para los usuarios normales y otra para los administradores):

- Usuarios.

- Roles.
- Asignación de Usuarios.
- Jerarquía de Roles.
- Objetos.
- Características de Objetos.
- Permisos.
- Acciones.

El modelo con las características que acabo de señalar se ve como en la Figura 2.6.

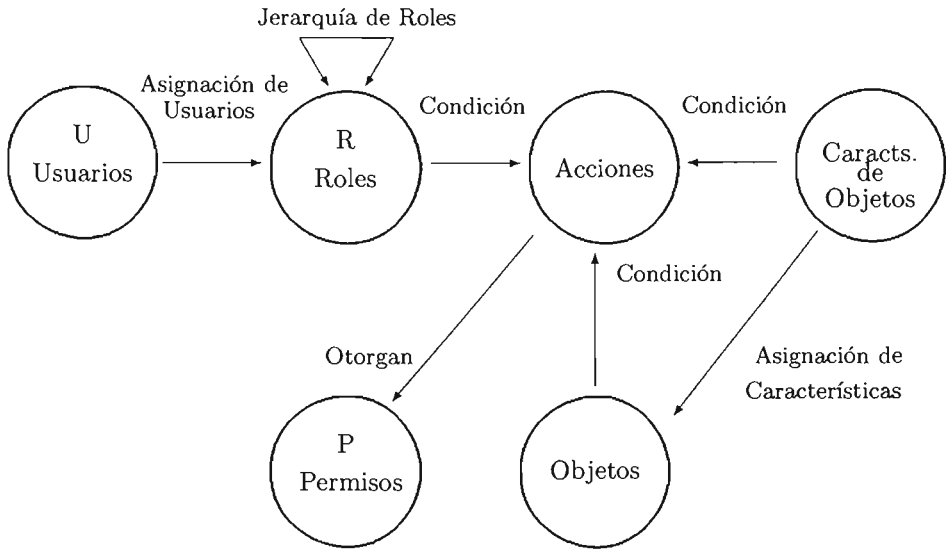


Figura 2.6 Principales características de un modelo semántico para el Control de Acceso Basado en Roles

Este modelo representa la información que debe contener el sistema, los usuarios representan la parte real del sistema, mientras que los roles, objetos, permisos y acciones las características más descriptivas del problema. Así, un sistema que utilice el modelo debe verificar que las relaciones se cumplan y la información semántica asociada a cada conjunto también (en el caso de los permisos). En consecuencia, el sistema que presento aquí sólo se hará cargo de recibir peticiones, identificarlas, verificar si se pueden cumplir, arrojar la

respuesta y efectuar la operación si fue permitida; todo esto de forma independiente a la descripción semántica.

Los actos importantes se colocan en la categoría de acciones y en ésta se ven las colecciones que la afectan, como son los roles, los objetos y sus características. Una vez dado el permiso se verifica que las condiciones y las relaciones se cumplan. Hay algunas acciones que no dependen del rol ni del objeto, sino de las relaciones entre ambos (por ejemplo, el creador de un documento puede leer el documento sin importar cual sea el rol que tiene y el tipo de documento que desea ver) por lo que no necesitan de una acción que las describa.

Adición de la autenticación

La parte de autenticación (es decir, la verificación de los datos proporcionados por la persona para identificarla) se hará antes de establecer una sesión y asignar dicha sesión a un usuario (Figura 2.7). Debido a que el enfoque de este trabajo está más relacionado con el CABR realizaré dicho trabajo por medio de un nombre de usuario y una contraseña. Toda esta operación se toma como una sección individual dentro de todo el sistema (para hacerlo modular).

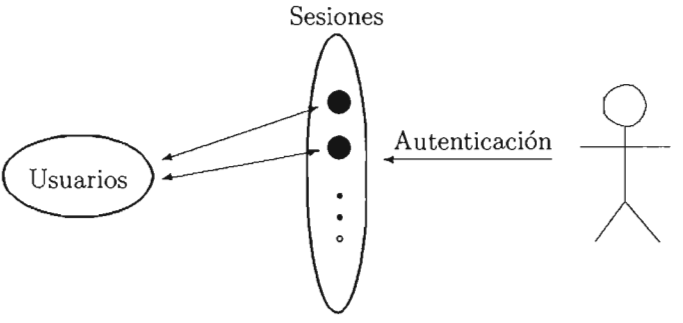


Figura 2.7 Autenticación

Para obtener un permiso se revisa que las condiciones de las acciones se cumplan, lo que hace complicado el acceso de alguien que no cumpla con los requisitos. Además, cada acción puede ser registrada y verificada, lo que permite proporcionar más confiabilidad.

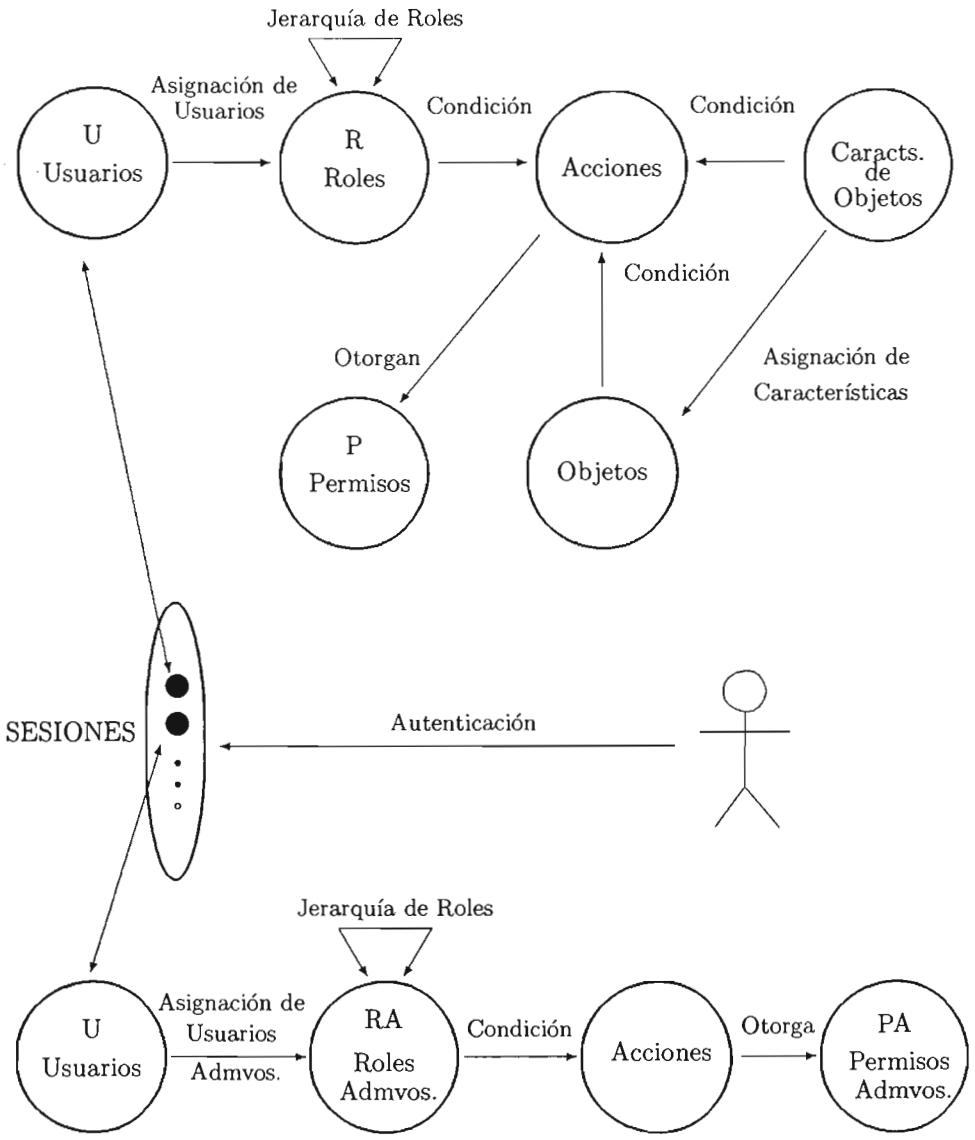


Figura 2.8 Modelo Semántico para el Control de Acceso Basado en Roles

En conclusión

Finalmente, el diagrama completo del Modelo Semántico para el CABR queda como en la Figura 2.8. A diferencia de los usuarios comunes, los administrativos no tienen colecciones de objetos (y por lo tanto, tampoco características de objetos) puesto que los objetos que modifican son justo las categorías corres-

pondientes a los usuarios normales. Cabe mencionar que este modelo representa una forma de agrupar y relacionar la información semántica del sistema (que estará presentada en RDF/XML); la parte correspondiente al código, sólo se encargará de obtener, almacenar y verificar que los datos se cumplan además de mostrarlos, pero el control de los recursos se deberá solamente a las especificaciones hechas en RDF.

Capítulo 3

Un Sistema que utiliza un Control de Acceso Semántico

La realidad de las cosas depende sólo de la opinión.
Todo en la vida es tan obscuro, tan diverso,
tan opuesto que no podemos asegurarnos de ninguna verdad.

- Erasmo de Rotterdam, *Elogio de la locura*

Establecer un modelo sin llevarlo a la práctica es un procedimiento poco fructífero. Un sistema que he tomado como base para desarrollar uno que implemente el modelo descrito en la sección 2.2.4 es el creado por Rajsbaum y Zheng [17]. Este sistema implementa un modelo de Control de Acceso Semántico que sólo cubre una parte del modelo descrito en el capítulo anterior y trabaja en particular con el Control de Acceso Basado en Roles.

Ya que este sistema es una parte importante de este trabajo, lo describo en este capítulo. El sistema está desarrollado en Java 2 Plataform SE 1.4.1, y el manejo de gráficas RDF es por medio de un marco de trabajo en Java para la construcción de aplicaciones, llamado Jena, en su versión 1.5.0.

3.1. Planteamiento del problema

El escenario es una compañía editorial. En esta compañía se publica una revista periódicamente. Además, se desea un sistema que maneje el acceso a todos los documentos que se emplean ahí para producirla.

Para la revista hay un *Editor en Jefe* quien inicia el proceso de publicación asignando un grupo de editores (llamados *Editores de Edición*) responsables de algún ejemplar (también llamado *Edición*) que sale a la venta. Estos editores están a cargo de buscar algún número de *Trabajos Escritos*, cada uno de los cuales es redactado por uno o más *Autores* quienes los envían a *Críticos* anónimos para ser sometidos a revisión. Durante el proceso de revisión, los Críticos envían las revisiones a los Autores, quienes a su vez pueden enviar respuestas a los Críticos o nuevas versiones de los documentos. Debido a que el Editor de Edición tiene que manipular una gran cantidad de Trabajos (cada uno con su propia complejidad de aprobación), tiene que asignar un responsable (*Editor de Artículo*) para cada uno de ellos y darle seguimiento. Al final, los Editores de Artículo deciden cuándo aceptar los artículos[1].

En este escenario se contemplan roles y objetos que interactúan a través de acciones. Con respecto a los objetos se distinguen dos clases: los documentos reales y los virtuales. Los documentos reales que aparecen son los trabajos escritos, las revisiones (de los documentos) y las respuestas. Los documentos virtuales incluyen la revista, la edición y los artículos que se presentarán en cada revista. Todas estas colecciones se incluyen para organizar los objetos reales en una jerarquía significativa, esto es, una revista contiene varias ediciones y cada edición consta de una colección de varios artículos, los cuales pasan por una serie de revisiones antes de ser aceptados (ver Figura 3.1).

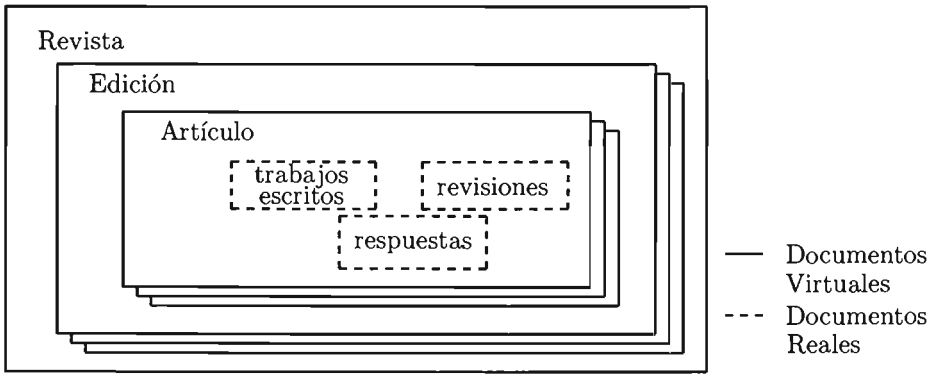


Figura 3.1 Jerarquía de documentos

Así, los roles que forman parte del sistema son Editor en Jefe, Editor de una Edición, Editor de un Artículo, Autor del artículo y Crítico del artículo. Esto forma el orden que se ve en la Figura 3.2.

[1] La traducción al idioma inglés de Editor en Jefe, Editor de Edición, Editor de Artículo, Autor y Crítico es *Editor in Chief*, *Editor of Issue*, *Editor of Submission*, *Author* y *Reviewer*, respectivamente.

Las acciones que se permiten en el sistema son: asignar editor, asignar revisor (crítico), crear edición, crear artículo, enviar borrador (que es el artículo del autor aún sin aprobar), enviar revisión, enviar respuesta, listar personas, listar roles, agregar reglas, borrar reglas y ver reglas. Ninguna de las acciones que afectan reglas tiene restricciones de uso (es decir, cualquiera tiene acceso sin importar su rol).

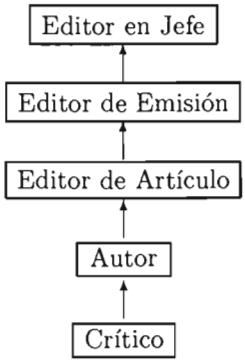


Figura 3.2 Orden en los Roles

3.2. Arquitectura del Sistema

La arquitectura está determinada por tres capas (ver Figura 3.3). La más profunda almacena todos los datos que maneja el sistema; en este caso se guardan en el paquete `ArchivosRDF`. El paquete `Vocabulario` contiene la representación de los conceptos más comunes, tanto los generales (como rol, persona, etc.) como los particulares (artículo, revista, etc.) representados en objetos de Java.

En la capa intermedia se encuentran los componentes que determinan la lógica de la aplicación. El paquete `Reglas` es el corazón del sistema ya que es el que procesa una petición y regresa una respuesta. El paquete `Utilería` proporciona herramientas para facilitar el trabajo del programador, mientras que el paquete `Acción` sirve de intermediario entre una petición por parte del usuario al *Manejador de Reglas* que se encuentra dentro del componente `Reglas`.

Finalmente, la capa más externa de la arquitectura es la que interactúa con el usuario y se encarga de presentar los resultados que fueron obtenidos del procesamiento de su petición. En esta capa destaca el paquete `Vistas`, el cual genera una presentación del resultado dependiendo del grado que ocupen los roles y los objetos que estén involucrados en la respuesta. El paquete `InterfazDeRed` implementa la interfaz de red para el sistema mientras que el paquete `Web` implementa la aplicación web del sistema. Por último, hay una sección nombrada Sistema, dedicada a atender las peticiones del usuario en la consola y activar la

interfaz de red.

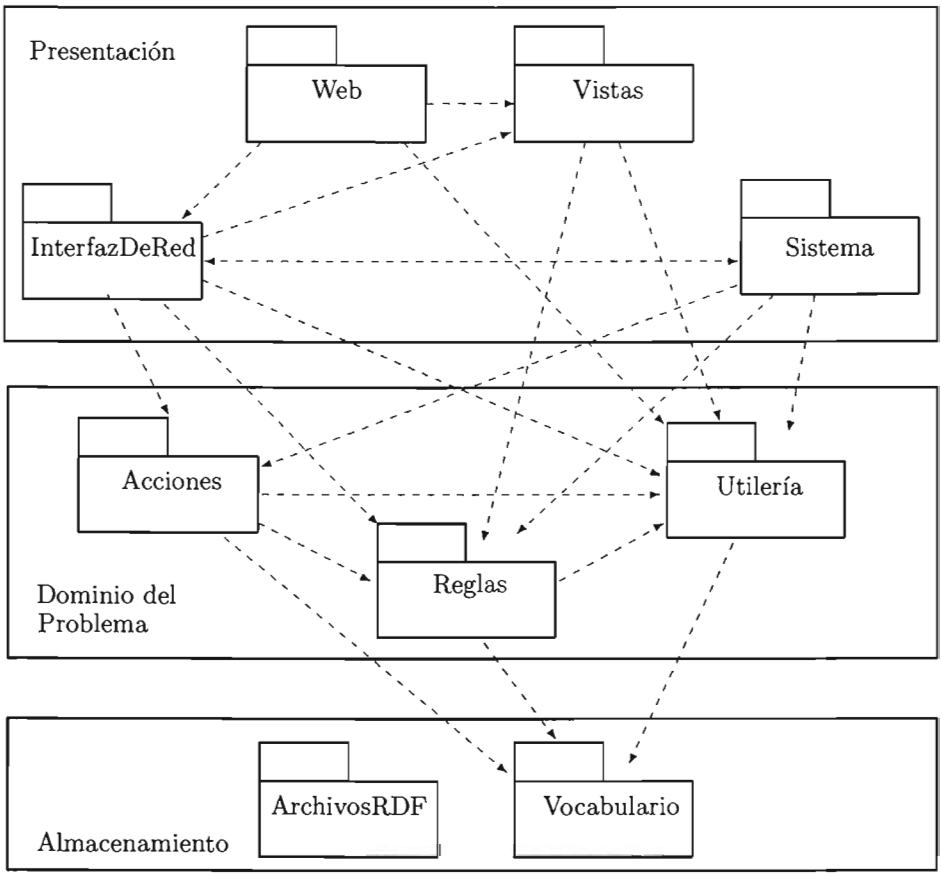


Figura 3.3 Diagrama de Componentes

Es necesario explicar un poco más a detalle los componentes esenciales de este sistema para entender cómo fue integrada la semántica y es lo que presento en seguida.

3.2.1. Componentes del sistema presentados en RDF/XML

Sin duda, esta parte es la más importante del sistema puesto que toda la información se mantiene ahí, además de que expresa su funcionamiento mejor que el código del programa, el cual solamente se encarga de manipularlos adecuadamente. Toda esta información es la que se encuentra en el componente `ArchivosRDF`, en la base de la arquitectura.

Los datos contienen reglas, acciones, roles, objetos y características de objetos, los cuales, en su conjunto describen el propósito del sistema. Adicional-

mente, como una parte que se debe considerar en cualquier implementación del modelo de CABR, están definidos los conceptos derivados de este paradigma (como el de rol, permiso, persona, etc.). Por supuesto, también se encuentra la presentación de los datos reales los cuales emplean las ontologías anteriores.

Un aspecto importante de notar es la aparición del concepto de reglas. Esta noción es la que describe la relación entre el objeto y el rol para otorgar un permiso. A continuación muestro más sobre la forma en que está representada la información.

Definiciones generales del Control Acceso Basado en Roles

Estas características muestran conceptos generales en RDF que son indispensables para el CABR. Los conceptos de rol, objeto, acción y regla son agregados además de contar con estados, estados de transición y permisos. Esto cubre un marco general del contexto en que estamos situados. Un ejemplo sobre esto es lo siguiente:

```
1.  <rdfs:Class rdf:ID="Role">
2.    <rdfs:label xml:lang="en">Role</rdfs:label>
3.    <rdfs:comment>Roles inside a information system</rdfs:comment>
4.  </rdfs:Class>

5.  <rdf:Property rdf:ID="registeredBy">
6.    <rdfs:label xml:lang="en">registeredBy</rdfs:label>
7.    <rdfs:comment>People registering this role</rdfs:comment>
8.    <rdfs:domain rdf:resource="#Role"/>
9.    <rdfs:range rdf:resource="#Person"/>
10. </rdf:Property>
```

Aquí se muestra la definición de un rol con una de sus propiedades: *registrado por* (*registeredBy*). En las líneas 3 y 7 se dan descripciones de la definición y la propiedad de la definición, respectivamente. El dominio de la propiedad *registrado por* es un rol, mientras que el rango es una persona; lo que se muestra en las líneas 8 y 9. Un ejemplo de permiso está a continuación:

```
1.  <rdfs:Class rdf:ID="Permission">
2.    <rdfs:label xml:lang="en">Permission</rdfs:label>
3.    <rdfs:comment>Class of access permissions</rdfs:comment>
4.    <rdfs:subClassOf rdf:resource="&rdf;Property"/>
5.  </rdfs:Class>

6.  <Permission rdf:ID="canReadPublic"/>
7.  <Permission rdf:ID="canReadAll"/>
```

Obsérvese que se define un recurso *permiso* (*Permission*, línea 1) y el cual es subclase de *propiedad* (*Property*, línea 4). También se definen otros dos recursos

de tipo permiso: *permiso de lectura pública* (*canReadPublic*) y *permiso de lectura de todo* (*canReadAll*) (líneas 6 y 7). De manera similar, se crean los conceptos restantes.

Descripción de los datos específicos del problema

Hay otras descripciones, que a diferencia de los conceptos anteriores, son particulares a la aplicación. Aquí se presentan los roles, objetos y características de objetos correspondientes al problema. Hay que observar que en algunas ocasiones la relación entre los objetos es inherente a su descripción debido a que las propiedades de cada uno indican alguna relación con otros objetos e incluso roles. Por ejemplo, la de un recurso *artículo* (*submission*) es como a continuación:

```
1.  <rdfs:Class rdf:ID="Submission">
2.    <rdfs:label xml:lang="en">Submission</rdfs:label>
3.    <rdfs:comment>The class of submissions</rdfs:comment>
4.    <rdfs:subClassOf rdf:resource="&iss;Folder"/>
5.  </rdfs:Class>

6.  <rdf:Property rdf:ID="author">
7.    <rdfs:label xml:lang="en">author</rdfs:label>
8.    <rdfs:comment>Author of the submission</rdfs:comment>
9.    <rdfs:domain rdf:resource="#Submission"/>
10.   <rdfs:range rdf:resource="#Author"/>
11.  </rdf:Property>
```

En este ejemplo se define un artículo (línea 1) que es una subclase de *carpeta* (*Folder*, línea 4) y se muestra la propiedad de autor, la cual indica que se aplica si el dominio es un artículo (línea 9) y el rango es un recurso autor. Utilizando las relaciones de dominio y rango se establecen relaciones entre los objetos y los roles. A continuación muestro un ejemplo en donde se relacionan los objetos por medio de la propiedad subclase:

```
1.  <rdfs:Class rdf:ID="Folder">
2.    <rdfs:label xml:lang="en">Folder</rdfs:label>
3.    <rdfs:comment>
4.      The class of grouping entities that help to organize
        documents
5.    </rdfs:comment>
6.    <rdfs:subClassOf rdf:resource="#DataItem"/>
7.  </rdfs:Class>
```

Aquí se indica que el recurso carpeta es una sublcase de *objeto* (*DataItem*, línea 6). Otros conceptos que se definen son las reglas y las acciones, pero esto merece más atención:

Presentación de reglas con RDF/XML

La regla está definida por sus características generales, como su categoría, una descripción y la definición de ésta. Un recurso de tipo *Patrón de Enunciado* (PE) encapsula la definición. El Patrón de Enunciado describe los datos que son necesarios para comprobar que la relación entre el sujeto y objeto (ver Figura 2.4) es la correcta. En una gráfica RDF se ve más o menos como lo muestra la Figura 3.4 (se omiten los URI para mejor lectura de ésta).

El sistema no ocupa una máquina de inferencia para las reglas. En vez de eso se usan *consultas* (en inglés, *queries*). Esto es debido a que tanto las máquinas de inferencia como las *consultas* usan la estructura *si... entonces*; además, Jena soporta este tipo de acciones. Así, el PE contiene el enunciado, descrito en RDQL[2], que se ejecutará sobre la gráfica que contenga los datos del sistema. El resultado del PE indicará los roles (en la posición del sujeto) que tienen permiso (señalado por el predicado) de realizar alguna acción sobre ciertos objetos (en la posición del objeto).

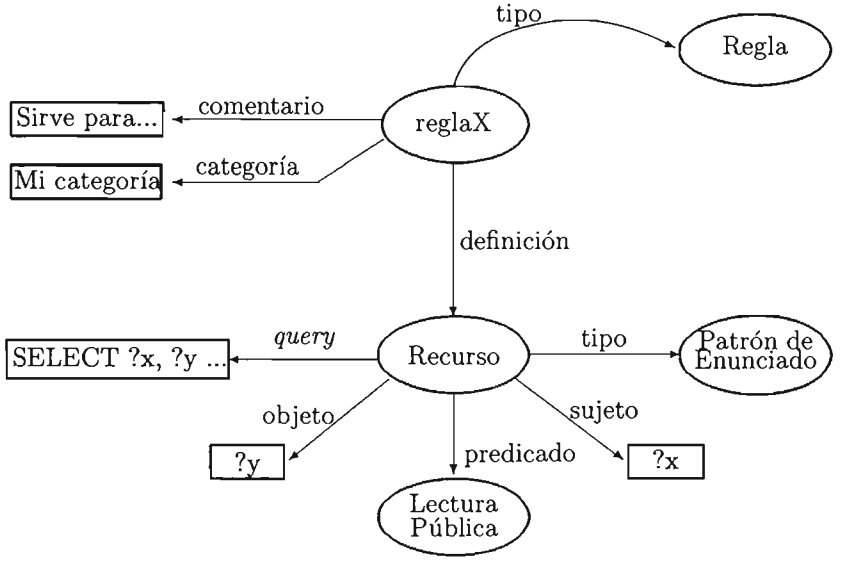


Figura 3.4 Representación gráfica de una regla

3.2.2. Manipulación de las Reglas en el Sistema

El control de las reglas se hace por medio de un Manejador de Reglas (MR) que se ubica dentro del componente `Reglas` de la arquitectura (Figura 3.3). El

[2]RDQL es acrónimo de *RDF Data Query Language* y es un lenguaje para extraer información de gráficas RDF para modelos de Jena [3].

MR contiene todos los enunciados de las reglas, en donde cada uno de ellos tiene su definición por medio de un recurso de tipo Patrón de Enunciado (ver Figura 3.4).

Cuando un usuario hace una petición al sistema, éste se expresa por medio de acciones; es decir, un usuario no pregunta *¿puedo crear un artículo?* sino simplemente da la orden *¡crea un artículo!*. Una vez identificada la acción, se indica el rol y el objeto sobre el que se desea aplicar. Por medio del Manejador de Reglas (que contiene todas las reglas) se compara cada predicado (señalado por el permiso que maneja ese enunciado) de cada regla con el que le corresponde a la acción (para nuestro ejemplo corresponde a *puedeCrearArtículo*); si no se encuentra una regla que maneje tal predicado entonces se niega el permiso. En caso de que se encuentre la definición de la regla (es decir, señalado por su Patrón de Enunciado) con el predicado que nos interesa entonces se obtiene la *consulta* (*query*) de la definición de este recurso, el cual será ejecutado sobre los datos del sistema. El resultado de esta *consulta* nos da un conjunto de pares ordenados (x, y). Cada par tiene una relación semántica entre sus elementos, a saber, la indicada por los enunciados de la *consulta*. Por ejemplo, si la *consulta* es:

```
1.  SELECT ?x, ?y
2.  WHERE  (?x, <rdf:type>, <jss:Editor>),
3.         (?x, <iss:withRegardTo>, ?submission),
4.         (?submission, <iss:member>, ?y)
5.  USING  rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>,
6.         iss FOR <file:info-system-schema#>,
7.         jss FOR <file:journal-system-schema#>
```

entonces las tuplas son pares en donde el primer recurso es de tipo editor (línea 2) y el segundo es de un documento referente a un artículo (como puede ser una revisión o una respuesta) del cual el editor esté a cargo (línea 3). Esto significa que a los pares (x, y) obtenidos de la *consulta* se les puede aplicar el predicado del PE. Esto se puede entender mejor si se coloca así:

x *predicado* y

Por ejemplo, si el predicado fuera *puede leer todo* entonces se tiene:

x *puede leer todo* y

lo que se refiere a lo siguiente (si se ocupa la *consulta* anterior): *"Los editores pueden leer todo sobre los documentos de los que están a cargo"*.

Así, las acciones y las reglas trabajan en conjunto para otorgar un permiso. Por un lado, las acciones determinan las restricciones que se aplican para realizar una actividad e indican el permiso que se es concedido si se cumplen. Por el otro, las reglas comprueban la relación entre los datos para realizar tal acción y al

final, si la petición de ejecutar un permiso para un rol sobre cierto recurso se verifica, entonces el permiso es concedido.

3.2.3. Interacción con el usuario

La interfaz con el usuario, tanto del Web como por modo texto, está creada con base en paquete `Vistas`, el cual contiene una abstracción de la presentación de la información. Este paquete presenta las vistas, para las personas, de los objetos del sistema. Sin embargo, esta información la muestra con base en un orden que está establecido en el código, lo que resulta en una gran desventaja en caso de que se modifiquen los rangos de los roles en la empresa. El cambio de cargos puede ser muy poco probable, pero aun así se debe tomar en consideración.

3.3. Modelo Semántico asociado a este sistema

En este sistema se toman en consideración los conjuntos de usuarios, roles, objetos, permisos, acciones, cacterísticas de objetos y reglas. Aquí, las reglas verifican la relación entre el objeto y el rol para otorgar el permiso, así que éstas toman el lugar de los permisos en el modelo (ver Figura 3.5).

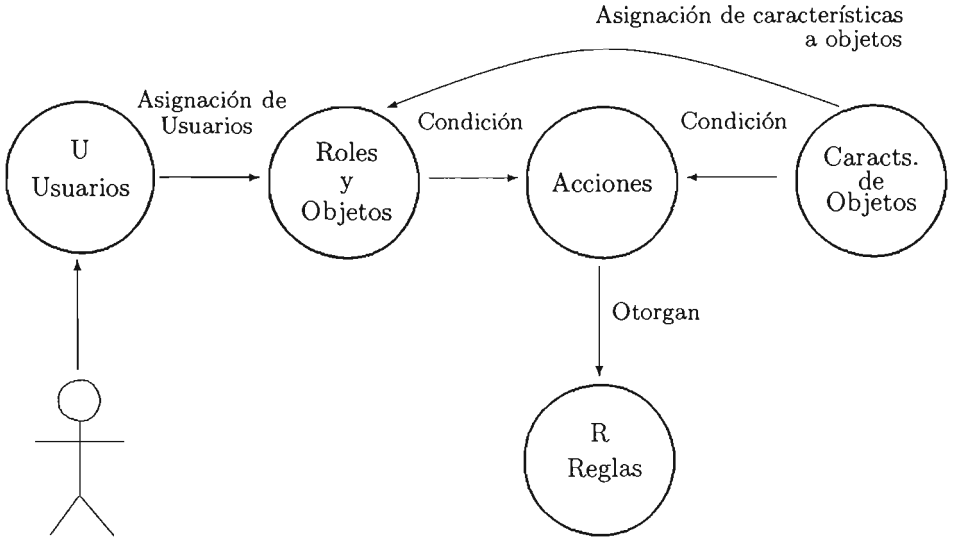


Figura 3.5 Modelo Semántico del Sistema

Este modelo cubre muchos de los factores necesarios en el CABR, pero hay algunas características interesantes que posee, y que no afectan de ningún modo importante: la definición de roles y objetos en una misma ontología. Tal parece

que esto es consecuencia de la estrecha relación que hay entre ellos (lo cual, no afecta en gran medida porque ambos siguen siendo determinantes en las acciones que se ejecuten). Aun así, hay elementos que hacen falta a este modelo.

3.4. Comparación de Modelos Semánticos

En general se cubren las colecciones más importantes indicadas en la sección 2.2.4 (Figura 2.8) y que caracterizan al CABR. No obstante, aún hace falta una sección administrativa y dividir adecuadamente los recursos manipulados. Además, no cuenta con una sección de autenticación. La jerarquía de roles tampoco está especificada y sólo la de objetos se puede deducir con base en las propiedades entre ellos.

Adicionalmente, hace falta almacenar en disco la nueva información que se obtiene, además de mejorar las interfaces para el usuario. La más importante de estas interfaces es la dedicada al personal administrativo, ya que se debe tomar en consideración que este personal no cuenta con conocimientos de RDF para hacer los cambios que se desean.

Capítulo 4

El Sistema de Control de Acceso Semántico a Recursos Basado en Roles

Nace el arte de la experiencia por varios modos,
mostrando el camino con el ejemplo.

- Manilio, *Astrología*

4.1. Objetivos del sistema

La finalidad de este trabajo es construir un sistema que se apegue al Modelo de Control de Acceso Semántico Basado en Roles descrito en la sección 2.2.4. Para alcanzar este propósito es necesario observar que de la sección 3.4 se obtiene una lista de características para agregar al sistema proporcionado por Rajsbaum [17], las cuales muestro a continuación:

1. Agregar la sección administrativa.
2. Agregar la sección de autenticación.
3. Agregar el orden de los roles.
4. Mejorar las interfaces de usuario.
5. Agregar el uso de la ontología FOAF.

La parte más compleja se encuentra en el primer punto, porque en ella es necesario agregar otro conjunto de roles y permisos para los administradores.

Además, es importante mantener los permisos de los roles encargados de la administración del sistema separados de los utilizados por los usuarios comunes. Esto tiene el fin de evitar involucrar nociones dirigidas a diferentes personas y, por lo tanto, accesos incorrectos.

Parte esencial sobre el Control de Acceso es la identificación del usuario, por lo que se debe realizar algún tipo de control al respecto. Para este propósito elegí un tipo de autenticación básica, por *nombre de usuario* y *contraseña*. Dado que este trabajo es una iniciativa para la integración de la semántica a un modelo, me parece conveniente iniciar con un método sencillo de autenticación.

El orden en que se encuentra el conjunto de roles está en código, lo cual no es nada flexible, así que es conveniente agregar una clasificación descrita en RDF. Aunque es poco probable un cambio en la organización, es mejor tenerla representada de tal forma que se pueda prevenir cualquier problema en el futuro.

En el Control de Acceso no figura una preocupación por la interfaz de usuario, pero es importante desplegar los datos de la mejor manera posible, para un mejor aprovechamiento de ellos. Así, hay que tomar en consideración que las funcionalidades otorgadas por el sistema se deben presentar tanto en una interfaz de texto como en la Aplicación Web. Adicionalmente se presenta la opción de elegir el idioma en que se desea interactuar con la aplicación usando un vocabulario nuevo para el despliegue de los mensajes.

En el Web Semántico es importante el uso de distintas ontologías. Sin embargo, el sistema presentado en el capítulo anterior hace uso de vocabularios creados específicamente para solucionar el problema sin presentar ninguna interacción con otros. Así que para obtener una verdadera integración con el Web Semántico, es necesario hacer un cambio sobre algunas definiciones; en particular la descripción de *persona* es cambiada por la definida en FOAF. La ventaja de realizar esto se encuentra en poder reutilizar la información creada previamente.

Como una ventaja más del empleo de distintas ontologías y de la integración con otros sistemas, sin pérdida de su contenido semántico, se aprovecha la descripción proporcionada por cada archivo FOAF. Esta información es dada en el registro de una persona en el sistema y los datos adquiridos son resguardados como contenido adicional a la descripción del individuo. De esta forma, la caracterización de un individuo permanece en el contexto que él mismo quiera proporcionar con su archivo FOAF.

4.2. Extensión al sistema inicial

Cada una de las modificaciones mencionadas en la sección anterior, se realizó de la siguiente manera: primero la autenticación, luego se agregó el orden para los roles, después la sección administrativa, seguí con el cambio en las

definiciones de algunas ontologías y por último se mejoró la interfaz del usuario. Se trabajó así para tener completamente lista la sección del usuario común, pero de igual forma hubiera sido que se tomara otro orden. El desarrollo de cada una de estas modificaciones la describo a continuación.

4.2.1. Sección de autenticación

Para esta sección se creó la definición de *login* y *password* en la ontología que contiene la información referente al sistema. También se agregaron los datos respectivos para los usuarios y administradores por medio de una descripción en RDF. Esta especificación quedó así:

```
<rdf:Property rdf:ID="login">
  <rdfs:label xml:lang="en">login</rdfs:label>
  <rdfs:comment>User name for identification</rdfs:comment>
  <rdfs:domain rdf:resource="&foaf;Person"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:ID="password">
  <rdfs:label xml:lang="en">password</rdfs:label>
  <rdfs:comment>The password for a user</rdfs:comment>
  <rdfs:domain rdf:resource="&foaf;Person"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

Para incrementar un poco más de seguridad se codificó el *password* con el algoritmo de Blaise de Vigenère como está detallado en [7]. Este algoritmo es un método polialfabético que utiliza la *tabla de Vigenère* y una llave (el *login*) para su propósito. En palabras de Galaviz y Magidin: "La idea es utilizar un monoalfabeto[1] distinto para cada carácter del mensaje, en el orden especificado por la llave" [7]. Además de extender el vocabulario, se crea una clase que manipule estos datos y use el algoritmo de Vigenère cada vez que sea necesario. Adicionalmente, se aplicó el algoritmo SHA-1 al resultado que arroja el algoritmo de Vigenère para evitar que el *password* se pueda descifrar fácilmente.

Es importante mencionar que aquí se usa el correo electrónico para distinguir a las personas. Uno de los problemas aún vigentes en el Web Semántico es precisamente la forma de reconocer a un individuo, pero ya que todavía no se resuelve este conflicto, usaré esta manera (que es de las más aceptadas) para referirme a ellas. Un ejemplo de la forma en que permanecen almacenados los datos está a continuación:

[1] En un monoalfabeto "se tiene una correspondencia entre el alfabeto de escritura y el alfabeto de cifrado, y esta correspondencia está fija durante todo el proceso de cifrado y descifrado" [7].

```
<foaf:Person>
  <vCard:EMAIL>aaa@example.com</vCard:EMAIL>
  <iss:login>loginA</iss:login>
  <iss:password>ee86e77668e9d38ba21a16ef1baac925d49949e9</iss:password>
</foaf:Person>
```

4.2.2. Orden en los roles

Para el grado de los roles fue necesario agregar un orden *lessDegreeThan* al vocabulario. Además, para cada rol se especificó el rango que le corresponde; esto fue tanto para los roles del sistema como para los administradores. El orden se definió así:

```
1.  <rdf:Property rdf:ID="lessDegreeThan">
2.    <rdfs:label xml:lang="en">degree</rdfs:label>
3.    <rdfs:comment>
4.       Relation where the responsibilities of the subject
5.       are lower in the hierarchy than those of the object
6.    </rdfs:comment>
7.    <rdfs:domain rdf:resource="#Role"/>
8.    <rdfs:range rdf:resource="#Role"/>
9.  </rdf:Property>
```

En la primera línea está especificado el nombre del orden. Las líneas 3 a la 6 dan una descripción de esta nueva propiedad. Finalmente, en la línea 7 se indica que la propiedad será aplicada a un rol, mientras la línea 8 muestra que el objeto del enunciado que ocupe esta propiedad también será un rol. Esta definición se aplicó como muestro en seguida:

```
1.  <rdfs:Class rdf:ID="EditorOfIssue">
2.    <rdfs:label xml:lang="en">EditorOfIssue</rdfs:label>
3.    <rdfs:comment>
4.        The role class of editors of a journal issue
5.    </rdfs:comment>
6.    <rdfs:subClassOf rdf:resource="&iss;Role"/>
7.    <iss:lessDegreeThan rdf:resource="#EditorInChief"/>
8.  </rdfs:Class>

9.  <rdfs:Class rdf:ID="EditorInChief">
10.   <rdfs:label xml:lang="en">EditorInChief</rdfs:label>
11.   <rdfs:comment>The role class of editors in chief</rdfs:comment>
12.   <rdfs:subClassOf rdf:resource="&iss;Role"/>
13. </rdfs:Class>
```

En la línea 1 se tiene al recurso Editor de Edición (*EditorOfIssue*) como una clase. En la línea 6 se indica que este recurso es un Rol (porque es subclase de *Role*). La línea 7 muestra que la propiedad *lessDegreeThan* está aplicada al

Editor de Edición y señala al Editor en Jefe (*EditorInChief*, que también es de tipo Rol, línea 12) como un rol de grado mayor que él.

4.2.3. Interfaces de usuario

Para hacer más amigable el sistema para el usuario se decidió traducir la información. El idioma en que las interfaces estaban inicialmente era el inglés; y se abrió la posibilidad de elegir entre ese idioma y el español. Es deseable trabajar sobre la descripción en RDF para hacer esta tarea y así evitar hacer modificaciones en el código; sin embargo, esto trae una nueva complicación.

En el análisis de esta situación se encontró que en general, la definición de un nuevo recurso va ligada con un comentario acerca del objeto o idea que representa. Frecuentemente la definición también tiene una etiqueta que da un nombre al recurso para que sea identificado por las personas (el cual regularmente tiene el mismo nombre local[2] del recurso creado y se identifica por el uso de la propiedad *rdfs:label*). La propiedad *rdfs:label* permite señalar el idioma en que se presenta la información, como en el siguiente ejemplo:

```
1.  <rdf:Property rdf:ID="creator">
2.    <rdfs:label xml:lang="en">creator</rdfs:label>
3.    <rdfs:comment>
4.      The creator (role) of this data item
5.    </rdfs:comment>
6.    <rdfs:domain rdf:resource="#DataItem"/>
7.    <rdfs:range rdf:resource="#Role"/>
8.  </rdf:Property>
```

La línea 2 muestra la etiqueta dada al recurso y el idioma en que se escribe (por medio del atributo *xml:lang*); sin embargo, esta información no tiene una representación de tripletas en RDF. Como consecuencia, expresar el idioma en que está una etiqueta no es posible.

Para un contexto general, una solución a este problema es crear otro recurso, que represente el mismo concepto, pero con las descripciones en cada uno de los idiomas que se desea traducir. Sin embargo, esta respuesta no deja claro que dos recursos sean equivalentes (o pertenezcan a clases equivalentes) ni en RDF ni en RDFS existe una propiedad que represente dicho concepto (aunque OWL provee la propiedad *owl:equivalentClass* para tener esta asociación pero se aplica sólo entre clases y no sobre cualquier recurso).

La representación final quedó como a continuación explico. Inicio con un recurso (digamos X), que representa el concepto principal del mensaje que deseo

[2] El *nombre local* es el nombre dado al nuevo recurso. Esto es, si se toma el URI del recurso, la última parte que lo compone es su nombre local.

dar. A X le sigue una sucesión de nodos en blanco (para esto se usa la definición de rdf:Seq) que apunta a una lista de recursos (que también son nodos en blanco). Cada uno de estos recursos finales contiene una descripción de X además de una propiedad que señala el idioma en que está escrita la descripción. Así lo que se tiene que hacer es identificar al recurso del cual se describe el mensaje y luego encontrar su descripción dependiendo del idioma en que se desee obtener. Un ejemplo del vocabulario para hacer esto está a continuación:

```
1. <rdfs:Class rdf:ID="FileNotFoundException">
2.   <rdfs:label xml:lang="en">FileNotFoundException</rdfs:label>
3.   <rdfs:comment>
       Package identified for the java.io.FileNotFoundException class
     </rdfs:comment>
4.   <rdfs:subClassOf rdf:resource="#io"/>
5.   <rdfs:subClassOf rdf:resource="#Exception"/>
6.   <rdf:type rdf:resource="#A0"/>
7. </rdfs:Class>

8.  <rdf:Description rdf:about="#A0">
9.    <rdf:type
        rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"/>
10.   <rdf:_1 rdf:resource="#A1"/>
11.   <rdf:_2 rdf:resource="#A2"/>
12. </rdf:Description>
13. <rdf:Description rdf:about="#A1">
14.   <dc:language>es</dc:language>
15.   <rdfs:label>
        Fuente de datos no identificada
      </rdfs:label>
16. </rdf:Description>
17. <rdf:Description rdf:about="#A2">
18.   <dc:language>en</dc:language>
19.   <rdfs:label>
        Source of data was not found
      </rdfs:label>
20. </rdf:Description>
```

Para indicar el idioma en que está un mensaje se utiliza la definición realizada por la iniciativa de Dublin Core[3] como se puede ver en las líneas 14 y 18. La definición del mensaje se encuentra en la primera línea mientras que en los siguientes renglones se presenta en detalle el tipo de mensaje que representa. Obsérvese que el comentario dado en la línea 3 sugiere que cada mensaje debe estar dentro de un orden que se establece dependiendo del lugar que da origen al mensaje. Así, se crean una serie de clases para establecer la clasificación así como los tipos de mensaje que pueden ser: *Fatal*, *Error*, *Warning*, *Debug*, *Exception* e *Information*. En las líneas de la 8 a la 12 se presentan los posibles idiomas

[3]Doublin Core Metadata Inicitative es un foro que tiene como propósito desarrollar estándares de meta-datos que soporten una amplia cantidad de propósitos y modelos de trabajo. Para más información ver http://dublincore.org

en los que puede expresarse un mensaje, mientras que en las líneas 13 a la 20 se muestra, finalmente, el mensaje que nos interesa. Un ejemplo de un mensaje completo que fue definido se encuentra a continuación:

```
<rdfs:Class rdf:ID="journalman">
  <rdfs:label xml:lang="en">journalman</rdfs:label>
  <rdfs:comment>
    Package identified for the journalman system
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Package"/>
</rdfs:Class>

<rdfs:Class rdf:ID="action">
  <rdfs:label xml:lang="en">action</rdfs:label>
  <rdfs:comment>
    Package identified for the journalman.action info
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Package"/>
  <rdfs:subClassOf rdf:resource="#journalman"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Succeed">
  <rdfs:label xml:lang="en">succeed</rdfs:label>
  <rdfs:comment>
    Package identified for the succeed of an action
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#action"/>
  <rdfs:subClassOf rdf:resource="#Information"/>
  <rdf:type rdf:resource="#I0"/>
</rdfs:Class>

<rdf:Description rdf:about="#I0">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"/>
  <rdf:_1 rdf:resource="#I1"/>
  <rdf:_2 rdf:resource="#I2"/>
</rdf:Description>
<rdf:Description rdf:about="#I1">
  <dc:language>es</dc:language>
  <rdfs:label>Acción exitosa</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="#I2">
  <dc:language>en</dc:language>
  <rdfs:label>Action succeeded</rdfs:label>
</rdf:Description>
```

En este ejemplo se define el mensaje *journalman.action.succeed* y el cual será presentado al dar el idioma en que se desee presentar.

Esta pequeña discusión, que podría parecer vanal, tiene mucha importancia porque las barreras del idioma pueden ser la diferencia entre la integración de

un sistema o no.

4.2.4. Sección administrativa

La creación de esta sección resultó ser la más interesante ya que inicialmente se tenía planeado establecer permisos sobre los conjuntos y relaciones fundamentales del Control de Acceso para los usuarios comunes (ver Figura 4.1). Estas operaciones eran crear usuarios, roles y permisos, además de asignar roles a usuarios y asignar permisos a roles.

Sin embargo, durante el análisis de la situación se encontró que algunas actividades eran demasiado complicadas para efectuarlas, mientras que otras no estaban contempladas.

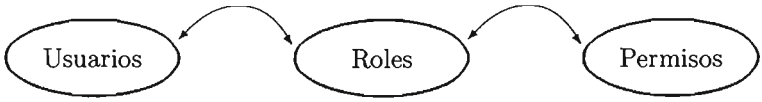


Figura 4.1 Relaciones Básicas del Control de Acceso Común

Un administrador debe tener la facultad de crear un nuevo usuario. Esto implica que también se debe tomar en consideración la asignación de un nombre de usuario y una contraseña para que tal persona pueda ingresar al sistema además de crear un recurso que lo represente. Así que de la Figura 4.1 se debe agregar esta característica.

Una vez agregado el usuario, el siguiente paso es asignarle un rol. Esta operación es sencilla porque sólo hay que crear la propiedad conveniente para relacionar a un usuario con un rol. Pero no cualquier rol puede ser asignado debido a que algunos de ellos requieren saber cuál es el objeto sobre el que va a actuar. A cambio de esto, existe la posibilidad de asignar un rol (por el usuario común) durante algún proceso (como asignar editor).

La creación de un rol es una actividad mucho más compleja. Crear un rol no sirve de nada si no hay alguna acción que pueda ejecutar y en caso de que hubiera una, haría falta crear el permiso asociado a ella. Además, la acción también debe ser descrita en código, lo que agrega complejidad en su elaboración.

Tal vez pueda ser pasada por alto la creación de una acción, pero no un permiso. La creación de un permiso implica indicar una relación explícita y clara (porque en su descripción se hace uso de RDQL), con respecto a las asociaciones entre el sujeto y el objeto que forman parte en él; además de conocer perfectamente la estructura del sistema y su descripción en RDF. Si se superan esas trabas es posible crear un permiso.

La creación de objetos es un caso similar al de acciones. Éstos necesitan relacionarse con alguna acción o con los permisos para poder utilizarse, lo cual lleva al problema anterior. Evidentemente, la evolución del Control de Acceso a un control semántico, trae nuevas complicaciones. No obstante, estos conflictos se podrían resolver si se contara con un editor de RDF y se tuviera una abstracción suficiente para evitar la escritura en RDF/XML. Ambas cosas aún están en desarrollo, lo cual hace que no se pueda aprovechar todo el potencial del Web Semántico todavía.

Con esto, sólo algunas actividades del Modelo Semántico propuesto en la sección 2.2.4 fueron realizadas. Finalmente quedaron sólo las siguiente operaciones:

1. Creación de un usuario.
2. Asignación de datos para autenticación.
3. Asignación de rol a usuario.
4. Creación de reglas.

quedaron también otras actividades como listar y borrar permisos. Cada uno de estos privilegios está asignado a un tipo de administrador.

4.2.5. Uso de la ontología FOAF

FOAF es acrónimo de *Friend Of A Friend* y es un proyecto acerca de la creación de páginas que puedan ser legibles para la computadora sobre la descripción de personas, las relaciones entre ellas, las cosas que pueden hacer y crear [8]. La fuerza del Web Semántico está en las relaciones que hay entre diversas entidades y esta correspondencia se facilita si se comparte el vocabulario.

En el sistema inicial, el vocabulario está limitado a las definiciones que el programador hizo, además de interactuar sólo con las ontologías esenciales. Para cambiar esta situación hay que eliminar la definición dentro de una de las ontologías sobre la descripción de *persona* y cambiarla por la de FOAF, y hacer lo mismo en todos aquellos lugares en donde se utilice dicho concepto. Esto es, inicialmente se tenía algo como lo siguiente:

```
1.  <rdfs:Class rdf:ID="Person">
2.    <rdfs:label xml:lang="en">Person</rdfs:label>
3.    <rdfs:comment>The class of people</rdfs:comment>
4.  </rdfs:Class>

5.  <rdf:Property rdf:ID="login">
6.    <rdfs:label xml:lang="en">login</rdfs:label>
7.    <rdfs:comment>User name for identification</rdfs:comment>
8.    <rdfs:domain rdf:resource="#Person"/>
```

```
9.    <rdfs:range rdf:resource="&rdfs;Literal"/>
10. </rdf:Property>
```

De las líneas 1 a la 4 se hace una definición del recurso *persona*. En la línea 8 se establece que el *login* (definido en la línea 5) se aplica a un resurso del tipo *Person*. Para cambiar esta situación se realizó lo siguiente:

```
1. <?xml version="1.0"?>
2.  <!DOCTYPE rdf [
3.    <!ENTITY rdf   "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4.    <!ENTITY rdfs  "http://www.w3.org/2000/01/rdf-schema#">
5.    <!ENTITY vCard "http://www.w3.org/2001/vcard-rdf/3.0#">
6.    <!ENTITY tns   "file:info-system-schema#">
7.    <!ENTITY foaf  "http://xmlns.com/foaf/0.1/">
8.  ]>

9.  <rdf:RDF xmlns="&tns;" xmlns:vCard="&vCard;" xmlns:rdf="&rdf;"
10.  xmlns:foaf="&foaf;" xmlns:rdfs="&rdfs;">

11.  <rdf:Property rdf:ID="login">
12.    <rdfs:label xml:lang="en">login</rdfs:label>
13.    <rdfs:comment>User name for identification</rdfs:comment>
14.    <rdfs:domain rdf:resource="&foaf;Person"/>
15.    <rdfs:range rdf:resource="&rdfs;Literal"/>
16.  </rdf:Property>

...
```

Se eliminó la definición de *person* y se hizo uso de la de FOAF (línea 7), que tiene su espacio de nombres especificado en la línea 10. Para aplicar la definición de *persona* hecha en FOAF sólo falta colocarla en los lugares en donde se hace mención de tal recurso; como es en la propiedad *login* (ver línea 14). En este ejemplo se agregó la definición de otros espacios de nombres (ver líneas de la 2 a la 8).

Este cambio de la información muestra que el costo de realizarlo es mínimo ya que principalmente se debe modificar sólo la ontología y las instancias de ella (aunque hay cambios muy leves en el código). Para mostrar la ventaja de esta alteración (y de su interacción con el Web Semántico) se agregó la opción de agregar el registro de una persona a partir de la lectura de un archivo con su descripción FOAF mostrando que, de esta forma, es posible reutilizar la información. La versión de Jena no permitió leer un documento creado con el FOAF-a-Matic[4] pero sí su descripción en tripletas (conocida como N-Triple); sin embargo, también fue necesario crear un *reconocedor* (*parser*) para verificar

[4]Es una herramienta para crear un documento FOAF de una persona. Para más detalles ver: http://www.ldodds.com/foaf/foaf-a-matic.html

que el documento esté en un formato adecuado porque Jena 1.5.0 no da soporte a esta operación.

La lectura del archivo FOAF no sólo se reduce al registro de una persona, sino también trata de obtener los datos adicionales que proporciona un documento de este tipo. Esto trae como consecuencia un aspecto más cercano de lo que representa a una persona en el Web. Agregar estos datos y mostrarlos cada vez que se proponga a la persona para asignarle un nuevo rol, hace que, por ejemplo, Juan Pérez ya no sea más "un Juan Pérez" sino el Dr. Juan Pérez, que vive en la Ciudad de México y que tiene publicados varios artículos en su página Web, además de que se le puede escribir a cierto correo electrónico.

Un ejemplo de integración de este sistema con otros se presenta justo en este punto. La definición que se tiene en el sistema por *correo electrónico* es la de *vCard:EMAIL*[5] mientras que la que se obtiene del archivo FOAF es *foaf:mbox*[6]. Sin embargo, ambas definiciones representan lo mismo para este contexto, así que el proceso de estos datos se hace de forma similar. Para otro sistema el significado de una de estas definiciones pudiera no ser el mismo, así que tendría que tomar algunas medidas para utilizar las descripciones de la manera que más le convenga.

4.3. Otras caraterísticas agregadas

Durante la creación del sistema se agregó otra característica que se cree de gran utilidad y la cual describo en seguida.

4.3.1. Registro de operaciones

Resulta importante tener un registro sobre las operaciones efectuadas. Para lograr esto, se utiliza la definición de las acciones para indicar si debe ser *registrada* o no. En la línea 6 del ejemplo que está a continuación, se indica que la operación *actCreateIssue* (crear Artículo) debe ser registrada:

```
1.  <iss:Action rdf:ID="actCreateIssue">
2.    <iss:roleType rdf:resource="&jss;EditorInChief"/>
3.    <iss:requires rdf:resource="&tns;canCreateIssue"/>
4.    <iss:feature>Repeatable</iss:feature>
5.    <iss:whetherCheckState>No</iss:whetherCheckState>
6.    <iss:whetherLog>Yes</iss:whetherLog>
7.  </iss:Action>
```

La especificación de esta *acción* señala que se debe llevar a cabo su registro, así como las condiciones en que se realiza. De esta forma sólo basta con llamar

[5]Se puede ver la definición en http://www.w3.org/2001/vcard-rdf/3.0#EMAIL
[6]Su definición está en http://xmlns.com/foaf/0.1/mbox

a la clase que se encarga del registro (con los datos necesarios) para poder hacer la anotación de la actividad.

Para guardar los datos también se representan por medio de RDF/XML, un ejemplo de cómo se ve esta operación almacenada es como muestro a continuación:

```
1.  <rdf:RDF
2.    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
3.    xmlns:NSO='file:info-system-schema#'
4.    xmlns:dc='http://purl.org/dc/elements/1.0/'
5.  >

6.  <rdf:Description rdf:about='file:my-journal-system#actCreateIssue'>
7.    <rdf:type rdf:resource='file:info-system-schema#Action'/>
8.    <NSO:roleType rdf:resource='file:journal-system-schema#EditorInChief'/>
9.    <dc:date>2004-09-27</dc:date>
10.   <NSO:moment>18:38:24:231</NSO:moment>
11. </rdf:Description>

12. </rdf:RDF>
```

Aquí se muestra el resultado de crear una nueva edición (*actCreateIssue*, línea 6) por el Editor en Jefe (línea 8) el día 27 de octubre del año 2004 (línea 9) a las 18:38 horas 24 segundos y 231 milisegundos (línea 10). La defincón del momento en que se efectuó la operación está descrita en las ontologías de la información general del sistema. Para la definición de la fecha (línea 9) utilizo la ontología de la organización Dublin Core Metadata Initiative.

4.4. Observaciones sobre este sistema

Lo que busqué al hacer estas modificaciones al sistema que presenté en el Capítulo 3, fue acercar ese sistema al modelo que presenté en la sección 2.2.4. Las características más relevantes fueron agregadas y en general se conservó el modelo que propuse. La única diferencia se encuentra en que la definición de los roles y los objetos está en la misma ontología, lo cual no representa un problema.

Las operaciones efectuadas por el personal administrativo resultan más complicadas de efectuar. Algunas de ellas resultan sencillas de realizar en el Control de Acceso común pero para la perspectiva que aquí se toma no es tan sencillo hacer las verificaciones sobre los datos. Así, algunas operaciones se hacen más simples, mientras que el costo de eso es tener otros procedimientos más complicados, los cuales se pueden resolver para una siguiente versión de este trabajo.

Capítulo 5

Conclusiones

El obtener información por medio de referencias cruzadas es (en parte) lo que hizo al Web exitoso, y la causa de ello fue directamente que el hombre, por naturaleza, trabaja de forma similar: por asociación de ideas. De alguna manera, poco a poco nos vamos acercando más a describir nuestro propio comportamiento en la computadora para mejorar el resultado de algunos procesos. Como consecuencia de esto, obtenemos un mejor *entendimiento* por parte de la máquina sobre los intereses del hombre; más aún, permite trabajar con métodos distintos a los tradicionales.

5.1. Importacia del Control de Acceso Semántico Basado en Roles

Desde mi punto de vista, una de las principales ventajas del Control de Acceso Semántico proviene de definir las políticas de acceso tomando en consideración la relación entre los objetos que toman parte en ellas. Este tipo de descripción permite hacer inferencias sobre tal información. Además, otorga otro tipo de resultados para el Control de Acceso: se puede aceptar o negar un permiso conforme a la relación que presenten los elementos que están en juego.

Otra ventaja de hacer descripciones semánticas es la de compartir e integrar información entre sistemas distintos. Si un grupo de sistemas utiliza una especificación semántica similar entre sus componentes, entonces la integración entre ellos podría obtenerse de forma automática. En el sistema del Capítulo 4 se agregó una opción de leer una descripción FOAF y con base en ella, dar de alta a una persona; esto es un ejemplo en el que se puede compartir información.

Las descripciones semánticas pueden ser tan detalladas como sea necesario. Esto permite considerar varios aspectos del Control de Acceso que comúnmente no son tomados en cuenta. Los estados sobre los objetos son uno de estos ejemplos: un artículo no puede ser aceptado si no se envía antes una primera versión

de él. Mientras más sea el detalle de una especificación, más cerca se estará de expresar la realidad.

5.2. Resultados

Los capítulos con mayor relevancia son el 2 y el 4 porque muestran la integración de la semántica a un sistema y su implementación, respectivamente. Para alcanzar estos resultados se tuvo que tomar en consideración el contexto en que se desarrolla cualquiera de las actividades del problema a solucionar; esto, con la finalidad de no dejar a un lado características que pudieron haber sido agregadas y que son importantes para obtener mejores efectos. Así el modelo final para el Control de Acceso Semántico Basado en Roles, surge a partir de la observación del objetivo del problema y su ambiente. La dificultad que se destaca en este proceso es la creación de ontologías, ya que deben ser lo más consistentes y completas posible.

Uno de los resultados más importantes de este trabajo está en el Capítulo 2 donde se muestra una iniciativa para un modelo de Control de Acceso Semántico. Algunos investigadores ([5], [16], [17]) que trabajan con este tipo de problemas, no presentan una explicación de cómo hacen su descripción semántica, lo que en este trabajo sí se busca. El tener un modelo facilita el análisis y desarrollo del problema. Sin embargo, su creación no es inmediata debido a la gran cantidad de características particulares que influyen en cada situación. El modelo propuesto es considerado una iniciativa para el Web Semántico ya que no hace mucho que empezó el desarrollo de éste (1999) y aún hay varias cosas por hacer para que llegue a un estado idóneo. En lo particular creo que en algún momento será necesario hacer modelos y esquematizar todas las operaciones que se realizan en el Web conforme a su semántica. Así, este trabajo sólo se convierte en una colaboración para la evolución de esta tecnología.

El Control de Acceso sólo es uno de los casos más importantes que se deben tomar en consideración para el acceso a los recursos. En particular se presenta aquí una aplicación que hace uso del modelo obtenido en la sección 2.2.4 en un caso real. El objetivo se ha alcanzado al observar cómo funciona el sistema: no solamente muestra una aplicación del Web Semántico, también añade características al Control de Acceso.

Uno de los resultados obtenidos del sistema presentado en el Capítulo 4 trata sobre la asignación de nuevos roles. Por ejemplo, al agregar a una persona con el rol de Editor en Jefe, ésta puede ver las ediciones que han sido publicadas (claro, está a cargo de eso), pero no puede ver los detalles de ellas ya que la creación de esos ejemplares no estuvieron a su cargo (sólo sobre los ejemplares de los que está al mando podrá tener algún acceso).

El Web Semántico nos permite expresar de mejor manera lo que deseamos

del sistema. Un ejemplo de esto se refleja principalmente en los permisos. Estos, para el Control de Acceso común, se colocan en tablas o listas, pero al expresarlos semánticamente se hacen explícitas las asociaciones entre las entidades (como los objetos y los roles) que forman parte del permiso.

Un resultado interesante es referente al uso de la ontología FOAF para la descripción de una persona. Con el simple hecho de tratar tal concepto de dicha forma, es posible obtener la información necesaria de un documento de este tipo, haciendo así más sencillo su proceso.

5.3. Algunos problemas importantes que se presentaron

A veces, la implementación de un sistema basado en información semántica necesita hacer una gran abstracción de los datos. Resultó difícil separar algunas características particulares del problema, dadas las relaciones entre todos los datos. Más aún, algunos aspectos pertenecientes exclusivamente al sistema presentado en el Capítulo 4 deben ser codificados, como es el caso de las acciones. Para estos problemas es necesaria una máquina de procesamiento mucho más compleja y, sin embargo, aun así pueden existir características muy ligadas a las paricularidades del problema (o a una forma específica para manipularla), que resulta en un trabajo mucho más elaborado que el presentado.

Uno de los mayores problemas, pero también el más sencillo de resolver, fue el referente a la descripción semántica. La mayoría de las ontologías estaban definidas conforme al significado que el programador les había dado. Sin embargo, hay que recordar que parte del poder del Web Semántico está en la reutilización de definiciones hechas anteriormente. Así que resultaba ser muy importante tomar las definiciones ya realizadas previamente, porque en caso contrario, la semántica se limitaba al contexto del programador. De ahí la importancia de agregar las descripciones FOAF para el concepto de *persona*.

Hacer especificaciones semánticas no es sencillo. Algunos conflictos surgen hasta que se trata de realizar alguna operación. Es por eso que muchas veces se recomienda trabajar por periodos en cuestiones de este tipo, para tener siempre una mejor perspectiva al respecto. Uno de los problemas presentados sucedió al intentar hacer las operaciones de los administradores. No todas las operaciones son posibles de realizar sin modificar de forma considerable otras categorías. Una de ellas es la creación de permisos, la cual implicaba agregar acciones (las que pueden ser inconsistentes con el resto de ellas si no se tiene cuidado). Para lograr tal consistencia era necesario agregar una máquina de razonamiento particular a la aplicación. Así, no sólo hay elementos que tienen un costo más alto de elaborar, sino también que se vuelven demasiado dependientes de la aplicación.

Los nodos en blanco son recursos que representan conceptos derivados de otros más generales, o que representan un concepto específico con ciertas características. El punto crítico es saber cómo identificar tales nodos: si por sus características o por un nombre (como es el caso con el concepto de *persona*). Reconocerlos por nombre no es correcto dado que no cuentan con uno solo. Sin embargo, conocerlos por sus características tiene como responsabilidad elegir las propiedades correctas para cada tipo (para una persona sería preguntarse ¿cómo identifico a una persona en el Web?), además de cuidar que el *ejemplar* (*instance*) de un concepto realmente cumpla con al menos los datos necesarios para distinguirlo. Desafortunadamente este aspecto es uno de varios que aún faltan por aclarar porque la importancia de un nodo en blanco puede ser alta pero no se le puede nombrar por su naturaleza.

5.4. Una breve perspectiva del futuro

Es importante tener un modelo (para cada problema) que describa las relaciones entre los datos, con el fin de contar con un mejor manejo y explotación de ellos. Aquí se propone uno (en el Capítulo 2), el cual une un modelo del Control de Acceso Basado en Roles con el paradigma del Web Semántico. Así, no sólo se pretende dar una manera de agregar la semántica al Control de Acceso, sino también se espera que sirva para atacar de forma similar otro tipo de problemas. La integración de estos dos temas da ventajas a ambas áreas. Para el Control de Acceso se presenta una opción de superar algunas de las dificultades que enfrenta, mientras que para el Web Semántico, este trabajo resulta una de sus aplicaciones más importantes.

Un aspecto que se observó durante la elaboración de esta tesis, fue la relevancia que tomaron todas las características particulares del problema. Afortunadamente hay varios esfuerzos de distintos grupos que se dedican completamente a esta materia ([23], [22], [10], [2]) y de los cuales se pueden rescatar aquellos elementos que no se tomaron en cuenta aquí.

Por ejemplo, algunos modelos de presentación de políticas de acceso son Ponder, Rei y KAoS ([23]). En particular, Rei, es un lenguaje para especificación de políticas expresado en OWL-Lite[1], pero también permite la descripción de ellas usando RDF, DAML+OIL[2] y OWL ([10]). Además, cada una de estas tecnologías cuenta con mecanismos de razonamiento sobre sus políticas. Sin embargo, hay que destacar que KAoS y Ponder cuentan con una interfaz gráfica para la creación de ellas, mientras que Rei no tiene tal ventaja. No obstante, Tonti *et*

[1]OWL-Lite es una versión simplificada de OWL que permite hacer clasificaciones jerárquicas y establecer restricciones sencillas. Para más información consultar: http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[2]Lenguaje para hacer clasificaciones más sofisticadas y dar propiedades a los recursos que RDFS. Además, es el antecesor de OWL. Para más información ver http://www.daml.org/.

al. indican que Rei es el modelo que más ha tenido éxito por su descripción en RDF ([23]).

Es interesante notar que la forma en que se describe una política en Ponder, Rei y KAoS es muy similar. En estas tres tecnologías, cada política consta de: las condiciones iniciales en las que se debe ejecutar, el permiso que ofrece y el estado final que resulta de aplicarla. En comparación con este trabajo, la diferencia se encuentra en la falta de una especificación de un resultado final. Así, una posible extensión al sistema presentado en el Capítulo 4 es añadir esta característica.

En particular, Rei ha sido utilizado en una tecnología llamada OWL-S. Esto se tratará con más amplitud a continuación.

5.4.1. OWL-S

OWL-S[3] es un lenguaje para especificar las restricciones y capacidades de los Servicios Web ([22]) y consta de tres partes. La primera se utiliza para describir el tipo de servicio que proporciona un Servicio Web; esto con el fin de hacer más fácil su publicación y descubrimiento en el Web (a esta sección se conoce como *service profile*). La siguiente parte provee el vocabulario para señalar la forma en que opera el servicio (conocida como *process model*). La última fracción permite especificar los detalles de interoperación con otro programa o agente para tener acceso a él (llamado *grounding*).

OWL-S provee, como una extensión a los servicios de seguridad, una forma de representación de políticas. Esto lo hace por medio de una combinación de dos ontologías: Rei y OWL-S, con las cuales permite detallar la privacidad, autenticación y confidencialidad de un servicio ([22]). Esto influye principalmente durante el proceso de descubrimiento de servicios, ya que para seleccionar el mejor proveedor se debe verificar la compatibilidad de sus políticas con las del solicitante. Para realizar esto, se integra una máquina de razonamiento llamada OWL-S Matchmaker y que utiliza la máquina de razonamiento de Rei.

La herramienta que realiza todos los pasos para relacionar la petición del cliente con el servidor es una máquina virtual llamada OWL-S VM. Ésta depende de la descripción del modelo de proceso (*process model*) y la especificación de cómo se debe interactuar con el servicio (*grounding*) para automatizar la interacción entre los Servicios Web minimizando la intervención humana.

5.4.2. Extensión del sistema con estas tecnologías

OWL está mostrándose como un mejor modelo de presentación de datos (en algunos aspectos) que RDF ([14] y [9]). Así, una opción para extender en un futuro este trabajo, es haciendo el cambio a este vocabulario. Las complicaciones

[3]OWL-S es parte del proyecto *DARPA Agent Markup Languaje* y fue originalmente conocido como DAML-S ([2]).

que surgen al hacer un cambio en las definiciones son mínimas (como se muestra en la sección 4.2.5). Sin embargo, el proceso de inferencia se vería afectado ya que el cambio a OWL establece nuevas asociaciones entre los conceptos.

Hay una gran diferencia con respecto al sistema presentado aquí y la tecnología OWL-S a pesar de que tienen el mismo propósito. Esto se debe a que, para permitir la interacción entre distintas entidades, OWL-S no sólo hace la descripción del servicio y sus políticas, sino también especifica la manera en que se debe establecer una comunicación entre ellas (por medio del *grounding*). Si se deseara utilizar la tecnología de OWL-S en el sistema mostrado en este trabajo, habría que definir todas las políticas nuevamente. Además, se debería agregar la descripción del servicio y el flujo del modelo utilizando el vocabulario de OWL. En resumen, no se podría recuperar nada *del programa* del sistema actual. Sin embargo, con el cambio a esa tecnología, se podría ganar una forma mucho más completa de descripción sobre las características del servicio, así como tener como ventaja el uso de una máquina de razonamiento general (OWL-S Matchmaker).

5.5. Otras extensiones técnicas

Hay dos problemas técnicos que deben ser resueltos para las siguientes versiones del sistema. La versión de Jena es una limitante importante para el desarrollo del sistema. Una consecuencia de usar la versión 1.5.0 es que algunas acciones, como la lectura de archivos RDF estándar, no es posible (lo cual repercutió en la lectura de archivos en forma de N-Triple en vez de RDF/XML). El cambio a versiones más actuales implica volver a escribir el sistema que ya estaba hecho, sobre todo porque las nuevas versiones de Jena han cambiado mucho y que hay algunas modificaciones sobre su código fuente para obtener algunas de sus estructuras internas. Hacer el cambio entre estas dos tecnologías tiene un gran costo sobre el tiempo de programación, así que decidí terminar de construir todas las funcionalidades del sistema inicial y presentar la idea del Web Semántico. De todo esto se mantiene la lógica de la aplicación y la información, que se encuentra en RDF/XML.

Finalmente, el segundo problema es derivado del anterior. Trata sobre la lectura del archivo RDF/XML en la creación de un usuario del sistema. El estándar actual de escritura de un RDF no coincide con el que maneja la versión 1.5.0 de Jena, pero sí reconoce el formato de N-Triple. Es por eso que se tiene que hacer la conversión entre estos dos formatos. Sin embargo, todo esto se puede llevar a cabo en una siguiente versión de este sistema.

Un aspecto que se puede desarrollar en un futuro para el sistema es la evolución del proceso de autenticación. En general, cualquier usuario puede acceder al sistema desde computadoras distintas, pero es probable que no se desee este tipo de comportamiento (o al menos se espera que se notifique al usuario del

uso de su rol por otra entidad), así, hay que tener un control sobre este tipo de casos. También, hay que vigilar la sincronización de los recursos cuando hay más de un rol trabajando con ellos.

Con estas observaciones concluyo este trabajo.

Concédeme, hijo de Latona, éste es mi ruego,
el gozar de mis trabajos en buena salud y con sano juicio,
sin afligirme con una vejez ajena al dulce canto de las musas.

- Horacio, *Odas, I*

Apéndice A

Manual para el uso del sistema

A.1. Objetivo del sistema

El sistema maneja los movimientos que se realizan dentro de un periódico. Para esto, el periódico consta de un conjunto de personas que pueden fungir como: *Editor en Jefe*, *Editor de Edición* (o *de Emisión*), *Editor de un Artículo*, *Revisor de un Artículo* y *Autor de un Artículo*. La asignación de cada uno de estos roles es realizada por otros roles, como se verá en este manual. A continuación se describen las operaciones que cada rol puede realizar desde la interfaz gráfica y la interfaz en texto, en ese orden.

A.2. Interfaz gráfica

La información general que se muestra en la aplicación Web se describe a continuación. La presentación de varias pantallas es muy similar, por lo que en la sección A.3 se indican las características generales que tienen. Luego se encontrarán las secciones que hacen una descripción más detallada de lo que permite hacer la aplicación.

A.3. En general

En todas las pantallas se encuentra un título de bienvenida para el usuario. Este título consta del *nombre de usuario* y el *rol* que desempeña en ese momento dentro del sistema. Seguidos de estos datos se despliegan las siguientes opciones: *Cambiar Rol* y *Salir*. El elegir la opción de *Cambiar Rol* hará que se muestre la primera pantalla que el usuario vio al entrar al sistema (que es la lista de sus roles, ver sección A.3.2). Si el usuario resulta ser un Administrador entonces la primera pantalla que se mostrará será la de las acciones que puede ejecutar.

En caso de que el usuario sea un empleado del Periódico entonces la pantalla mostrará los roles que se pueden ejercer en el sistema. La opción *Salir* hará que se presente la pantalla de identificación del sistema; para más información ver la sección A.3.1.

Además de estas características que presentan la mayoría de las pantallas, hay 5 formas distintas de presentar o pedir información en el sistema. A continuación están descritas cada una de las posibles pantallas que el usuario podría ver.

A.3.1. Nombre y Contraseña

La pantalla inicial (de identificación) pregunta por los datos que autentifican al usuario; como son el *nombre de usuario* y *contraseña*. Adicionalmente, se pregunta por el idioma que se desea utilizar en el sistema para la presentación de los datos. Si la información proporcionada para ingresar al sistema es incorrecta, se mostrará de nuevo la pantalla de identificación y se presentará un mensaje de error. Ver la Figura A.1.

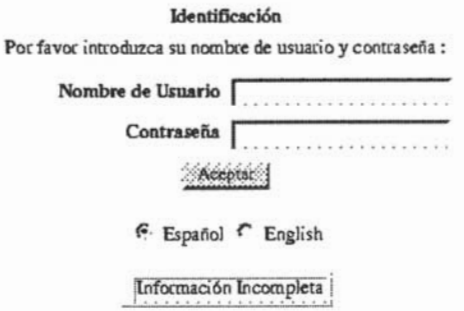


Figura A.1: Vista de la pantalla de identificación

A.3.2. Roles o acciones que ejerce un usuario

Cuando un usuario común entra al sistema se muestra una lista de los roles que se le han asignado (Figura A.2). Esta lista indica el rol que puede ejercer y el objeto del que es responsable. Para elegir alguno de estos roles, basta elegir una de las opciones que aparecen en la lista. En el caso de los administradores, se muestra una lista de las acciones que tienen permitidas hacer. Para ver más información sobre las acciones que puede realizar un administrador, ver la sección A.6.

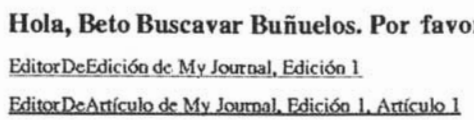


Figura A.2: Lista los roles de un usuario

A.3.3. Presentación de la información de un objeto

La información referente a un objeto se presenta en tres secciones. La primera sección (y la que se encuentra al inicio de la página) contiene los *datos generales* del objeto; por ejemplo, uno de estos datos podría ser el nombre del objeto. La segunda sección comprende las *acciones* que puede realizar el rol, que juega el usuario en ese momento, sobre el objeto. Finalmente, la última sección comprende los *elementos* que están contenidos en el objeto; por ejemplo, un Editor en Jefe puede ver las ediciones del periódico, mientras que un Editor de un Artículo podría ver la sucesión de los borradores que le han enviado y de las revisiones que se han hecho al artículo del que es responsable. Ver Figura A.3.

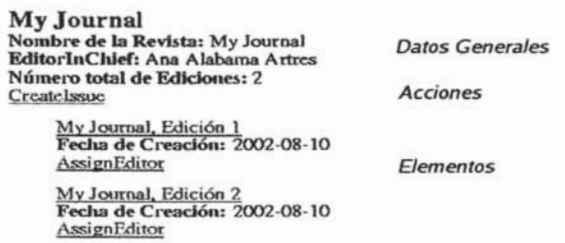


Figura A.3: Las secciones de las que se compone un página

A.3.4. Resultado de una acción

La ejecución de una acción puede resultar en uno de los dos siguientes casos:

1. Pedir algunos datos para completar su ejecución.
2. Mostrar el resultado de la ejecución de una acción.

La pantalla que se muestra en el punto 2 consta de dos partes. La primera sección tiene un mensaje con el resultado de la operación (mostrando su éxito o fracaso). La segunda parte muestra tres opciones a elegir, que son *Regresar*, *Cambiar Rol* y *Salir*. La opción *Regresar* se asemeja a oprimir un botón que se encargue de "*cancelar la operación*" y regresar a la pantalla anterior. La opción *Cambiar Rol* hace que se muestre la lista de los roles disponibles para el usuario

(ver sección A.3.2). Por último, la opción *Salir* hace que el usuario termine su sesión y regrese a la pantalla de autenticación (ver sección A.3.1).

La pantalla para el caso 1 también se divide en dos partes. La primera se explica en la sección A.3.5. En cambio, la segunda parte presenta 4 posibles actividades a seguir (ver Figura A.4). La primera es oprimir un botón con la leyenda *Enviar*, el cual enviará los datos al servidor y ejecutará la acción que se desee realizar. Las siguientes tres opciones, coinciden con las que aparecen en el punto 2 (*Regresar*, *Cambiar Rol* y *Salir*).

Figura A.4: Las cuatro opciones que se presentan al ejecutar una acción

A.3.5. Pantallas de captura de datos

Estas pantallas piden los datos que son necesarios para ejecutar una acción. Si se tienen los permisos adecuados, entonces en el sector de las *acciones* (ver sección A.3.3) se encontrarán las ligas hacia este tipo de pantallas. Para este tipo de casos sólo hay que proporcionar los datos que pide la aplicación y para enviarlos al sistema hay que oprimir el botón con la leyenda *Enviar*, que aparece al final de la página, como se explica en la sección A.3.4; las otras opciones que aparecen en la forma son las de *Regresar*, *Cambiar Rol* y *Salir* que se explican en esa misma sección.

A continuación, se proporciona la descripción sobre las diversas operaciones que se pueden ejecutar en el sistema. Primero se explica la forma en que se ingresa a la aplicación y luego se expone el resto de la funcionalidad con base en los roles que puede desempeñar el usuario común. Sin embargo, la descripción de las pantallas presentadas para los administradores se hará con base en las acciones que pueden ejecutar.

A.4. Identificación

La primera pantalla que se muestra del sistema (en la aplicación Web), y que es común para todos los usuarios, es la de identificación. Aquí, el usuario (ya sea el administrador o el empleado del periódico) debe de ingresar su *nombre de usuario* y *contraseña*. Adicionalmente, el usuario puede seleccionar el idioma en que se puede presentar la información. En este caso sólo existen

dos posibilidades: el inglés y el español (ver Figura A.1). Cuando la información porporcionada en la identificación es incorrecta, se muestra un mensaje indicando el error.

A.5. Los usuarios comunes

Lo que mostrará cada pantalla que vea el usuario depende del rol que haya elegido jugar. Después de pasar la pantalla de identificación se hace una lista de los roles que tiene el empleado del periódico (ver sección A.3.2). Así, las posibles pantallas que se presentan a continuación dependen de los roles que existen. Ver Figura A.2.

A.5.1. Editor en Jefe

El Editor en Jefe es el editor principal del periódico. También es el que se encuentra en la parte más alta de la jerarquía del sistema y tiene los siguientes permisos:

- Ver la información general del periódico. Entre esta información se encuentra lo referente a las ediciones y los datos públicos sobre los artículos.
- Crear una edición en el periódico.
- Asignar un editor a una edición.
- Ver la información detallada de las ediciones que creó.
- Ver la información general de los artículos de cada edición.
- Ver información general de las ediciones que crearon los otros Jefes de Edición.

La página que se muestra al Editor en Jefe está dividida en tres partes, como se describe en la sección A.3.3. A continuación se describe con más detalle lo que contiene cada una las secciones.

La *información general* del periódico se muestra en las primeras líneas de la página. Ahí se muestran datos como el Nombre del Periódico, el Editor en Jefe y el Número de Ediciones (o emisiones) de las cuales consta. Ver Figura A.5.

Después de la *información general* del periódico aparece una posible *acción* a realizar, que es la de crear una edición nueva (*Create Issue*). Al elegirla, aparecerá una nueva página que muestra el resultado de la operación, si tuvo éxito o hubo algún problema; esta página es como se describe en la sección A.3.4. Al regresar a la página original, se mostrará una nueva edición en la parte donde se muestran los *elementos* (ver la sección A.3.3) del objeto.

Figura A.5: Vista para un Editor en Jefe

La sección de los *elementos* que tiene un Periódico contiene una lista de las ediciones que corresponden al periódico. Cada elemento de lista contiene el número de edición, su fecha de creación y la posibilidad de asignar un editor para esa edición. Sin embargo, la asignación de un editor a una edición se restringe a que el Editor en Jefe actual sea el creador de tal edición.

Información sobre una Edición

Para ver la información de cada edición basta elegir, en la lista de los *elementos* (ver la sección A.3.3) del periódico, la edición en la cual se esté interesado. Esto mostrará una nueva pantalla con la información general o específica de la edición, dependiendo si el rol actual tiene permiso de ver o no los detalles de esa información.

En particular, un Editor de Edición vería la misma información que se muestra al elegir esta opción, si el rol actual fue el creador de ella. Para ver más detalles consultar la sección A.5.2.

Asignar un Editor de Edición

Para asignar un Editor de Edición hay que elegir la opción *Assign Editor*; que hará que se muestre una pantalla con la lista de las personas a las que se les puede asignar el rol. Algunas de ellas podrán venir acompañadas de su descripción en FOAF. Las acciones que se pueden realizar en esta pantalla son las que se indican en la sección A.3.4. Con esto, el nuevo rol aparecerá en la lista de los roles de la persona que fue elegida.

A.5.2. Editor de Edición

La pantalla que muestra la información correspondiente a un Editor de Edición está organizada como se describe en la sección A.3.3. Al inicio de la página se muestran los *datos generales*, los cuales son el número de emisión de la revista, la fecha de su creación, el (los) nombre (s) del (de los) editor (es) y el número total de artículos que conforman la edición. Ver Figura A.6.

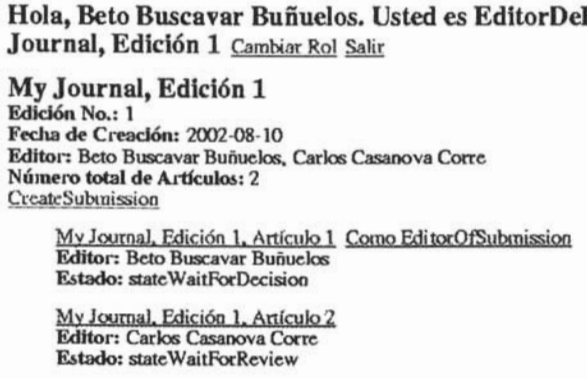


Figura A.6: Vista para un Editor de Edición

En general las operaciones que un Editor de Edición puede ejecutar son:

- Crear un nuevo artículo en la edición.
- Leer la información general sobre los artículos de la edición.

Información sobre un Artículo

La lectura de la información general sobre los artículos se puede hacer eligiendo alguna de las opciones mostradas en los *elementos* de la página para el rol Editor de Edición (ver sección A.3.3). Sin embargo, los datos que se mostrarán en esta sección serán sobre la información general como el estado del artículo y su editor.

Para crear un nuevo artículo se muestra la opción *Create Submission* en la parte de *acciones* (ver sección A.3.3) de la página. Al seleccionarla, la aplicación mostrará una página del tipo descrito en la sección A.3.5. A continuación se hace una pequeña descripción sobre la creación de este tipo de objetos.

Crear un nuevo Artículo, asignar un Autor de un Artículo y Editor de Artículo

Crear un nuevo artículo implica asignar un Autor al artículo y asignar un Editor de Artículo automáticamente, como se explica a continuación.

Si se elige crear un nuevo Artículo (*Create Submission*) entonces se mostrará una pantalla con los nombres de las personas que están registradas en el sistema y a las cuales se les puede asignar el rol de Autor. Adicionalmente se pedirá que se especifique el título del artículo. Para finalizar la operación, se elige alguna de las opciones que aparecen al final de la página y que se describieron en la sección A.3.5. De forma semejante a la asignación de un Editor de Edición podrá aparecer la descripción FOAF de las personas (si el sistema cuenta con una). Ver Figura A.7.

Al regresar a la pantalla inicial que se muestra para un Editor de Edición, aparecerá un nuevo artículo, en donde se mostrarán los nombres de los editores asignados y el estado del artículo (que será en un estado de espera por el artículo, ya que el autor del artículo no pudo haberlo enviado aún).

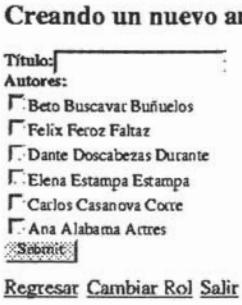


Figura A.7: Vista para la creación de un artículo

Al crear un artículo automáticamente, a la(s) persona(s) que se le(s) asignó el rol, tendrá(n) en su lista de roles el nuevo que se les acaba de crear. Además, al crear un artículo, se asigna un nuevo rol a su creador: Editor de Artículo (*Editor Of Submission*), el cual ahora es el responsable de ver las acciones que se realizan sobre el artículo (como seguir las revisiones que crean los Revisores o los borradores que manda el Autor). La información que se muestra cuando se elige jugar el rol de Editor de Artículo (*Editor Of Submission*) se describe en la sección A.5.3.

A.5.3. Editor de Artículo

Los *datos generales* (ver la sección A.3.3) de la pantalla para un Editor de Artículo se muestran los datos sobre:

- El título del artículo que está controlando.
- El número del artículo creado para la edición.
- Fecha de Creación del artículo.
- La lista de editores del artículo.
- Los nombres de los encargados de hacer la revisión del artículo.
- El número total de versiones del artículo.
- El estado en que se encuentra el artículo (si está en espera de que el autor envíe el documento, si lo están revisando o si está en espera de una decisión para determinar si será aceptado o rechazado).

En la sección de las *acciones* que se pueden llevar a cabo, se encuentra la de cambiar el estado del artículo y asignar un revisor para el artículo (ver Figura A.8). Estas acciones se describen a continuación.

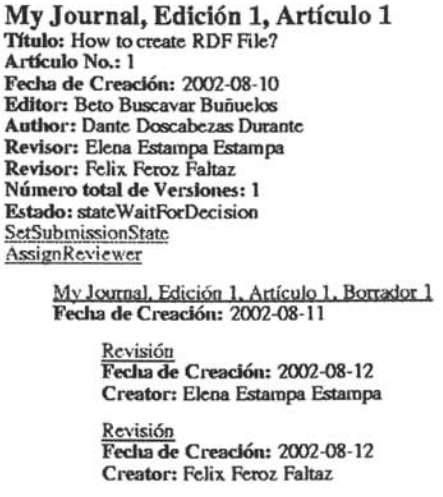


Figura A.8: Vista de un Editor de Artículo

Cambiar el estado de un Artículo

Al elegir la opción de cambiar el estado de un artículo aparecerá una pantalla como se describe en la sección A.3.5 (ver Figura A.9). Como opciones aparecerán los posibles estados en que puede aparecer un artículo y que se mencionan a continuación:

1. Estado Esperar por Documento.

2. Estado Esperar por Revisión.
3. Estado Esperar por Decisión.
4. Estado Aceptado.
5. Estado Rechazado.

Cada uno de estos estados representa un estado del documento y una acción que se puede tomar. El punto 1 es el estado en que se está esperando a que el Autor mande el documento, ya sea por primera vez o como corrección de alguna de las revisiones que se le hicieron. El estado Esperar por Revisión representa que el Revisor está en periodo de revisión del artículo y aún no envía el resultado. El punto 3 está en un estado de espera de decisión por parte del Editor de Edición, para ver si el artículo es aprobado o no. El estado final de un artículo puede ser aceptado o rechazado.

Figura A.9: Vista para la modificación del estado de un artículo

Algunos de estos puntos implican otras acciones, como agregar la acción de *envía documento* para un Autor, si es que se cambia el estado del Artículo al indicado en el punto 1. También, el estado *esperar por la revisión* hace que el Revisor tenga activada la acción de *enviar revisión.*

El ejecutar una de estas acciones hace que al regresar a la pantalla anterior se muestre el nuevo estado que se propuso en la sección de información.

Elementos que forman un artículo en una edición

Los *elementos* (ver sección A.3.3) de la página para un Editor de Artículo son los borradores y las revisiones que se han hecho para los artículos. Ver Figura A.8. Además, para cada borrador se hace un anexo de las revisiones y respuestas por las que ha pasado, se muestra la fecha de creación y se hace una anotación sobre quién elaboró el documento. Si se desea conocer más datos sobre un artículo, se elige la liga que lleva a su información particular. Los datos que aparecerán son la fecha de creación del elemento y su creador; además, si se anexaran los documentos reales en esta aplicación, entonces se mostraría en este momento la información. Ver Figura A.10.

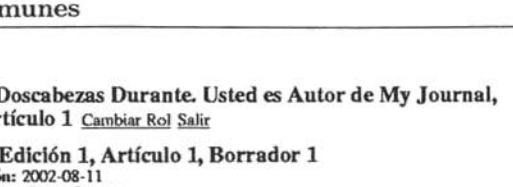


Figura A.10: Vista de un borrador enviado por un Autor

A.5.4. Autor de un Artículo

La presentación de la información para este rol es como se describe en la sección A.3.3. La *información general* que se presenta es: el título del artículo, su número (dentro de la edición), su fecha de creación, el editor del artículo, el número total de versiones y el estado en que se encuentra.

Si un artículo debe ser enviado por el autor, entonces se muestra la opción de enviar borrador (*Submit Paper*). Otra acción que puede realizar un Autor es la de enviar la respuesta a las revisiones que se realizó algún Revisor; así que en vez de aparecer la opción *Submit Paper*, aparecerá *Submit Response*. Ver Figura A.11.

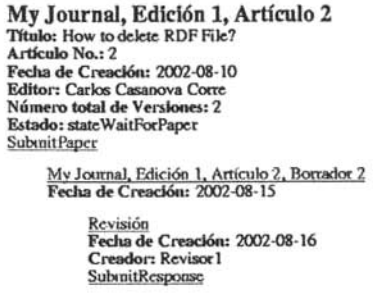


Figura A.11: Vista para un Autor de un artículo

A.5.5. Revisor de un Artículo

Al igual que en las pantallas anteriores, un Revisor de Artículo ve la información como se describe en la sección A.3.3. Los *datos generales* sobre el Artículo para el cual se está ejecutando el rol de Revisor, tiene los siguientes datos: título del artículo, número de artículo (dentro de la edición), su fecha de creación, el editor del artículo, el número total de versiones y el estado en que se encuentra.

En este caso, el revisor no tiene otras acciones que las de enviar la revisión para cada borrador que un autor le envíe. Por lo que la acción estará colocada

dentro de cada *elemento* (ver sección A.3.3) de la lista, ver Figura A.12. Dentro de cada *elemento* se muestran las ligas a las respuestas y las revisiones que se han hecho para el artículo, las cuales tendrán un aspecto similar a como se muestra en la Figura A.10.

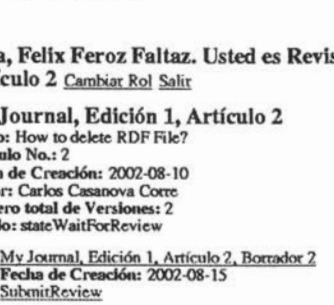


Figura A.12: Vista para un Revisor

A.6. Los administradores

El ingreso de un administrador al sistema es de forma similar a la de una persona común; sin embargo, si la identificación coincide con la de un administrador entonces se mostrará la pantalla con las acciones que su rol tiene permitido realizar. Hay dos tipos de administradores en el sistema: el *Administrador en Jefe* y el *Administrador General* o *Administrador*. Cada uno de ellos tiene distintos permisos y son los que se mostrarán en la lista. Las acciones que puede realizar cada administrador se describen a continuación.

A.6.1. Asignar un rol a un usuario

Al elegir esta opción aparecerá una pantalla pidiendo el rol que se desea asignar a un usuario, la identificación del usuario y el objeto sobre el cual se le asignarán los permisos, en ese orden (ver Figura A.13). La identificación del usuario al que se le desea asignar un nuevo rol deberá ser indicada por medio de su correo electrónico. Mientras tanto, el nombre del rol que se le desea asignar y el objeto sobre el cual actuará deben ser especificados con los URI que se utilizan en el sistema.

En este caso sólo hay dos roles disponibles. Estos roles son:

- Editor en Jefe del periódico (*Editor In Chief*).
- Revisor de un artículo (*Reviewer*).

Para dar de alta los cambios se debe oprimir el botón que tiene la leyenda *Enviar*. Esta acción hará que en la lista de roles del usuario aparezca una nueva

opción, a saber, la que se acaba de dar de alta. Si se desea cancelar la acción se elige la opción de *Regresar*.

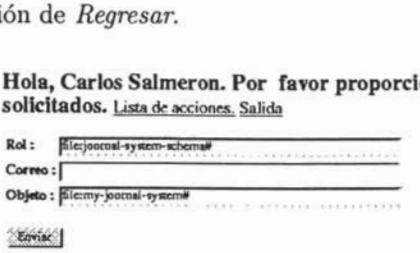


Figura A.13: Vista para asignar un rol

A.6.2. Agregar una persona al sistema

Al elegir esta opción aparece una página con dos formas para agregar a una persona al sistema. La primera parte de la página pide el nombre de la persona y el correo electrónico de ésta. Al oprimir el botón con la leyenda *Enviar* después de proporcionar los datos anteriores, el sistema registrará a un nuevo usuario, pero que aún no tiene un *nombre de usuario* ni *contraseña* asignados.

La otra opción para registrar a una persona (colocada en la parte inferior de la página) es agregando el archivo FOAF de ésta. El archivo FOAF debe estar en un formato de N-Triple y para crear un archivo de este formato se indican algunas ligas que pueden ser de utilidad. El procedimiento para obtener un archivo del formato adecuado es el siguiente:

1. Abrir la página del validador de la W3C.
2. Copiar el archivo FOAF en RDF/XML en la sección indicada para la transformación.
3. Elegir la conversión en el formato N-Triple y oprimir el botón con la leyenda *Enviar*.
4. Copiar resultado generado en un archivo con extensión *nt* y listo.

El archivo generado por la W3C tendrá el formato que se requiere para esta aplicación. Para indicar el archivo FOAF que se debe de agregar, hay que señalar la ruta de la ubicación del archivo con ayuda del botón con la leyenda *Buscar...*. Una vez mostrado el archivo en cuestión se oprime el botón con la leyenda *Enviar*. La ventaja de esta segunda forma de operar es que se puede agregar la información que se tenga en el archivo FOAF. Ver Figura A.14. Como resultado de esta acción aparecerá una nueva pantalla en donde muestra el éxito o fracaso de la operación con las características que se especificaron en la sección A.3.4.

Una vez realizada esta operación, es indispensable la creación de una identificación para que el usuario pueda ingresar al sistema. Esta operación se logra asignando una identificación al usuario (para más detalles vea las instrucciones de la sección A.6.3).

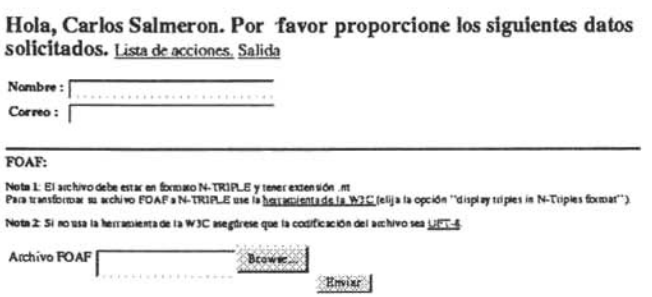


Figura A.14: Vista para agregar una persona al sistema

A.6.3. Crear la identificación para una persona

Al elegir esta opción, se muestra una pantalla que pide los siguientes datos:

- *Nombre de usuario* que será asignado,
- *contraseña* que será asignada al nombre de usuario y
- *correo electrónico* de la persona a la que se desea agregar la identificación.

El *nombre de usuario* que será asignado al usuario es el identificador por el cual el usuario será reconocido en el sistema. La *contraseña* es el identificador secreto que es asignado al usuario para ingresar al sistema. Finalmente se indica el *correo electrónico* de la persona a la cual se le desea asignar los datos que se acaban de indicar. Ver Figura A.15.

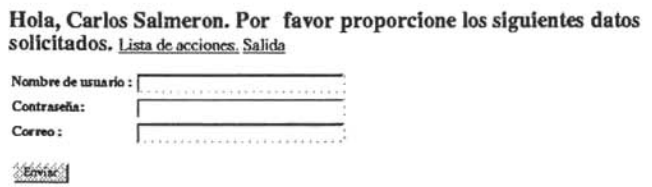


Figura A.15: Vista para crear la identificación de una persona

Después de oprimir el botón con la leyenda *Enviar* se muestra una nueva pantalla que indica el resultado de la operación como se describe en la sección A.3.4.

A.6.4. La lista de las reglas en el sistema

Al elegir esta opción aparece el comentario de cada una de las reglas que tiene el sistema. En el encabezado de esta pantalla aparece la opción de *Salir* del sistema o regresar a la pantalla inicial (*Lista acciones*). Ver Figura A.16.

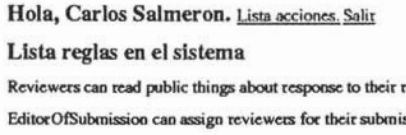


Figura A.16: Vista de las reglas del sistema

A.6.5. Obtener la información detallada de una regla

Al elegir esta opción se muestra una lista de las reglas (es decir, se presentan las reglas como se indica en la sección A.6.4). Cada regla se muestra como una liga, lo cual hace que al elegir una de ellas, se muestre otra pantalla que tenga la descripción completa de la regla. Ver Figura A.17.

El *predicado* indica el tipo de permiso que la regla mostrada trata de inferir. Mientras tanto, el *query* decribe el enunciado en RDQL que se emplea para obtener el permiso. La interpretación a la regla de la Figura A.17 es: *si el revisor X es el creador de la respuesta Y entonces X puede leer los atributos públicos de Y.*

Figura A.17: Vista de los detalles de una regla

A.6.6. Borrar una regla

Al elegir esta opción se muestra una lista como se indica en la sección A.6.4 con la diferencia que debajo de cada regla aparece la opción *Borrar esta regla.*

Al ejecutar esta acción aparece la lista de las reglas que quedan en el sistema. En el encabezado de la página aparecen las opciones de regresar a las acciones que tiene permitidas el rol (*Lista acciones*) y la de salida del sistema (*Salir*). Ver Figura A.18.

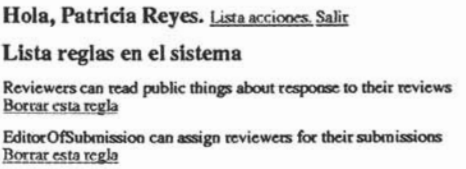


Figura A.18: Vista para borrar reglas

A.6.7. Agregar una regla al sistema

Para agregar una regla al sistema se tiene que elegir esta opción y llenar los datos que se requieren. Estos datos son los siguientes:

- Un comentario para la regla.
- Indicar quién hizo la definición.
- Una definición para la regla, la cual consta de:
  - El sujeto para el cual se actuará.
  - El predicado de la regla.
  - El objeto sobre el cual se actuará.
  - El query que expresa las relaciones entre el objeto, sujeto y el predicado descrito en RDQL.

Para crear una regla se puede tomar como ejemplo alguna de las reglas creadas dentro del sistema y que se pueden ver conforme se indica en la sección A.6.5. La parte final de la página tiene un botón con la leyenda *Enviar*, el cual, al ser oprimido, se encarga de mandar los datos otorgados. Al realizar esta acción aparecerá en la lista de las reglas la nueva regla agregada. Ver Figura A.19.

A.7. Interfaz de texto

Además de la interfaz gráfica se presenta una interfaz en texto en donde se pueden ejecutar las mismas operaciones que en la aplicación Web. A continuación se da una breve descripción del uso de esta interfaz.

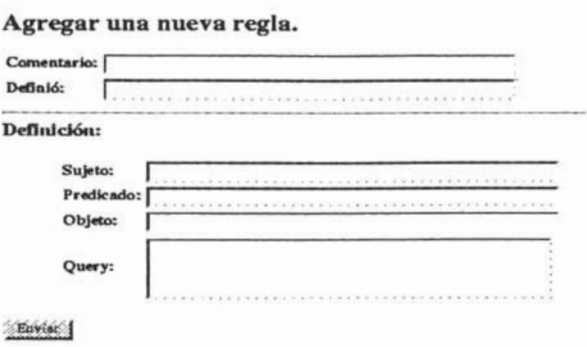


Figura A.19: Vista para agregar reglas

A.7.1. En general

Para la interfaz de texto se tienen las mismas operaciones que se indican en la interfaz gráfica; sin embargo, las acciones se realizan por medio de una lista de comandos.

Identificación

Para ingresar al sistema es necesario escribir el *nombre de usuario* y la *contraseña*. Al hacer esto aparece una línea de comandos que indica que se ha entrado al sistema como se ve en seguida:

```
Por favor escriba la instrucción {a:ayuda s:salir}:
```

Si no sucede esto, significa que los datos ingresados son incorrectos o que aún el usuario no está registrado en el sistema.

A.7.2. Usuario Común

Las acciones que un usuario común puede ejecutar son las que se indican en la ayuda para la interfaz. La línea de comandos aparece con un par de instrucciones básicas para poder entrar al sistema, que son:

- la salida del sistema y
- ayuda.

Con la *salida* del sistema se regresa a la parte de la identificación mientas que con la *ayuda* se obtiene la explicación que se muestra en seguida:

La ayuda para el usuario común

Para poder ejecutar alguna de las acciones que se realizan en la interfaz gráfica se tienen las siguientes posibilidades:

```
Ayuda:
  a                              -- da esta página
Modificar información:
  sp <persona>                   -- modifica la persona actual
  sr <rol>                       -- modifica el rol actual
Listar información:
  l                              -- lista la persona y rol actual
  lp                             -- lista todas las personas
  lr                             -- lista todos los roles de la persona
                                    actual
Genera vistas:
  v <recurso>                    -- genera una vista del <recurso>
Acciones:
  wci                            -- crea una nueva edición
  wae <edición><person>          -- asigna una <persona>como editor de
                                    <edición>
  wcs <edición>título <persona1>[<persona2>...]
                                 -- crea un nuevo artículo en la
                                    <edición>
                                 -- coloca el título y el autor es la
                                    <persona?>
  war <artículo><persona>        -- asigna una <persona>como revisor
                                    del <artículo>
  wss <artículo><estado>         -- coloco el <artículo>en el <estado>
  wsp <artículo>                    -- envia el nuevo documento como
                                    <artículo>
  wsre <artículo><borrador>      -- envía una revisión como <borrador>
                                    de un <artículo>
  wsrp <artículo><revisión>      -- envía una respuesta de <revisión>
                                    para el <artículo>

Todos las cosas entre <>deben ser URIs.
El espacio de nombres de un URI puede ser omitido si es
file:my-journal-system#
```

A.7.3. Administrador

Las acciones que un administrador puede ejecutar son las que se indican en la ayuda para la interfaz. La línea de comandos aparece con un par de instrucciones básicas para poder entrar al sistema, que son:

- la salida del sistema y
- ayuda.

Con la *salida* del sistema se regresa a la parte de la identificación mientras que la *ayuda* se obtiene la explicación que se muestra en seguida:

```
Por favor escriba la instrucción {a:ayuda s:salir}:
```

Si no sucede esto significa que los datos ingresados son incorrectos o que aún el usuario no está registrado en el sistema.

La ayuda para el Administrador

Si el usuario es un administrador, puede ejecutar alguna de las acciones que se realizan en la interfaz gráfica con los siguientes comandos:

```
Ayuda:
  a                                  -- da esta página
Información
  l                                  -- lista el rol y persona actual
Operaciones:
  cp <nombre><correo>                -- crea una persona
                                        los espacios deben ser
                                        reemplazados con _
  ca <usuario><contraseña><email>    -- crea la autenticación
  arp <rol><correo><objeto>          -- asigna un rol a una persona
Manejador de reglas:
  rl                                 -- lista todas las reglas
  rv <regla>                         -- da los detalles de la <regla>
  ra nombre_archivo                  -- agrega la regla definida en el
                                        archivo
  rd <regla>                         -- borra <regla>

Todos las cosas entre <>deben ser URIs.
El espacio de nombres de un URI puede ser omitido si es
file:my-journal-system#
```

A.8. Glosario

A

Acciones Las acciones que se pueden ejecutar en el sistema.

Administrador El rol que se encarga de realizar algunas acciones sobre la administración del sistema.
Artículo Es el documento que se pretende presentar en una edición. La creación de una artículo comprende la elaboración de borradores y revisiones.
Asignar Editor Es una operación en donde se asigna el rol de editor a una persona.
Assign Editor En inglés quiere decir asignar editor. Ver Asignar Editor.
Autor El rol que juega una persona encargado de crear un artículo.

B

Borrador El artículo elaborado por un Autor que aún no ha sido aceptado por el Editor de Artículo.

C

Create Issue En inglés quiere decir que se crea una edición en el periódico.
Create Submission En inglés quiere decir Crear un Artículo.
Contraseña El identificador secreto que tiene el usuario para ingresar al sistema.

E

Edición Es uno de los ejemplares que saca, cada cierto periodo, la revista. Por ejemplo, la Gaceta UNAM, sale dos veces por semana. En inglés se conoce por *issue*.
Editor de Artículo También conocido en inglés como Editor of Submission es el encargado de darle seguimiento al artículo.
Editor de Edición Es el editor de una edición en el periódico.
Editor en Jefe El editor principal encargado de la administración del periódico. Es el responsable de las ediciones realizadas en el periódico.
Editor In Chief Es la traducción en inglés de Editor en Jefe. Ver Editor en Jefe.
Editor Of Submission En inglés indica al Editor de Artículo.
Editor Of Issue Es la descripción en inglés de un editor de edición. Ver Editor de Edición.
Emisión Ver edición.
Enviar Es la acción de mandar la información a un destino particular.

F

FOAF Es el acrónimo de Friend of a Friend (*El Amigo de un Amigo*) y que representa un vocabulario que describe las características generales de las per-

sonas y las relaciones entre ellas.

I

Identificación Es la parte en que se pide la identificación del usuario para poder ingresar al sistema.
Issue Es la palabra en inglés para edición. Ver edición.

L

Login Ver nombre de usuario.

N

Nombre de usuario Es el identificador del usuario para poder ingresar al sistema. En inglés es conocido como login.

P

Password Ver contraseña.

R

Revisión Es el documento que elaboró un revisor como resultado de revisar un documento de tipo borrador.
Revisor Ver Revisor de un Artículo.
Revisor de un Artículo Es el rol encargado de hacer las revisiones de un artículo.
Rol Es el papel que juega una persona en el sistema. Se relaciona con una actividad o responsabilidad que tiene asignada en una organización.

S

Submission Es la traducción en inglés de artículo. Ver artículo.
Submit Es la traducción en inglés de enviar. Ver enviar.

U

Usuario común Es el empleado del periódico y que no es un administrador.
Usuario Ver usuario común.

Bibliografía

[1] Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, 1945.

[2] The OWL Services Coalition. Owl-s: Semantic markup for web services. http://www.daml.org/services/owl-s/1.0/owl-s.html, 2004.

[3] Hewlett-Packard Development Company. Rdql - rdf data query language. http://www.hpl.hp.com/semweb/rdql.htm, 2004.

[4] World Wide Web Consortium. Semantic web activity. http://www.w3.org/2001/sw/, 2004.

[5] Mariemma Inmaculada Yagüe del Valle, Antonio Mana, Javier Lopez, and José M. Troya. Applying the semantic web layers to access control. In *Proceedings of the 14th International Workshop on Database and Expert systems Applications*, pages 622–626. IEEE Computer Society, septiembre 2003.

[6] Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential.* The MIT Press, 2003.

[7] José Galaviz and Arturo Magidin. *Introducción a al Criptología.* Vínculos Matemáticos No.15, 2002.

[8] FOAF Group. The FOAF project. http://www.foaf-project.org/, 2004.

[9] Ian Horrocks and Peter F. Patel-Schneider. Three theses of representation in the semantic web. In *The Twelfth International World Wide Web Conference*, pages 39–47, 2003.

[10] Lalana Kagal, Tim Finin, and Anupam Joshi. Declarative policies for describing web service capabilities and constraints. In *W3C Workshop on Constraints and Capabilities for Web Services*, Octubre, 2004. http://ebiquity.umbc.edu/v2.1/paper/html/id/193/.

[11] Frank Manola and Eric Miller. Rdf primer, w3c recommendation 10 february 2004. http://www.w3.org/TR/rdf-primer/.

[12] John Naughton. *A Brief History of the Future, From the Radio Days to Internet Years in a Lifetime.* The Overlook Press, 2000.

[13] Bijan Parsia and Peter F. Patel-Schneider. Meaning and the semantic web. In *Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters*, pages 306–307, Mayo 2004.

[14] Peter F. Patel-Schneider. What is owl (and why should i care)? In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference.*, pages 735–737. AAAI Press, 2004.

[15] Apache HTTP Server Documentation Project. Authentication, authorization, and access control. http://httpd.apache.org/docs/howto/auth.html, 2004.

[16] Li Qin and Vijayalakshmi Atluri. Concept-level access control for the semantic web. In *Proceedings of the 2003 ACM workshop on XML security*, pages 94–103. ACM Press, 2003.

[17] Sergio Rajsbaum and Fengzhou Zheng. Exploring the use of semantic information in designing intelligent systems. Manuscrito, 2003.

[18] Pierangela Samarati and Sabrina De Capitani di Vimercati. Accesss control: Policies, models and mechanisms. In *FOSAD*, volume 2171 of *Series Lecture Notes in Computer Science*, pages 137–196. Springer-Verlag, 2001.

[19] Ravi Sandhu. What is rbac? http://csrc.nist.gov/nissc/1999/program/sandhu/, 1999.

[20] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer Society*, 29(2):38–47, Febrero, 1996.

[21] Juha Savolainen. The role of ontology in software architecture. A Position Paper for OOPSLA Workshop on How to Use Ontologies and Modularization to Explicitly Describe the Concept Model of a Software Systems Architecture. Nokia Research Center, 2003. Manuscrito.

[22] Katia Sycara, David Martin, Deborah L. McGuinness, Sheila McIlraith, and Massimo Paolucci. Owl-s technology for representing constraints and capabilities of web services. In *W3C Workshop on Constraints and Capabilities for Web Services*, Octubre 2004. http://www.w3.org/2004/08/ws-cc/dmowls-20040904.

[23] Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok. Semantic web languajes for policy representation and reasoning: A comparison of kaos, rei, and ponder. In *The Semantic Web - ISWC 2003, Second International Semantic Web Conference*, volume 2870 of *Series Lecture Notes in Computer Science*, pages 419–437. Springer-Verlag, 2003.