Actas del Taller de Trabajo Zoco'09 / JISBD

Integración de Aplicaciones e Información Empresarial
XIV Jornadas de Ingeniería del Software y Bases de Datos
San Sebastián, 8 de septiembre de 2009

http://www.tdg-seville.info/cfp/zoco/

Organizadores

José L. Álvarez, José L. Arjona, Iñaki Fernández de Viana (Universidad de Huelva)
Rafael Corchuelo, David Ruiz, Carlos Rivero, Hassan A. Sleiman, Inmaculada Hernández (Universidad de Sevilla)

Ponentes

Eduardo Martín Rojo y Vicente Luque Centeno (Universidad Carlos III de Madrid) | | Hassan A. Sleiman (Universidad de Sevilla) Patricia Jiménez (Universidad de Huelva) Carlos G. Figuerola, José Luis Alonso Berrocal y Angel Zazo Rodríguez (Universidad de Salamanca) Inma Hernández (Universidad de Sevilla) Paula Montoto, Alberto Pan, Juan Raposo, Fernando Bellas y Javier López (Universidade da Coruña) | | M. Mercedes Martínez-González, Beatriz Pérez-León (Universidad de Valladolid) y M. Luisa Alvite-Díez (Universidad de León) Ana Flores Cuadrado (Telefónica Investigación y Desarrollo), Eduardo Villoslada de la Torre (Telefónica Investigación y Desarrollo, Universidad de Valladolid) y Alberto Peláez Gutiérrez (Telefónica Soluciones) | | Ismael Navas-Delgado, Amine Kerzazi y José F. Aldana-Montes (Universidad de Málaga) María Pérez, Ismael Sanz, María José Aramburu y Rafael Berlanga (Universitat Jaume I de Castelló) Ismael Caballero, Mª Ángeles Moraga y Coral Calero (Universidad de Castilla–La Mancha) | | Juan A. Fraile, Javier Bajo (Universidad Pontificia de Salamanca) y Juan M. Corchado (Universidad de Salamanca) | | Francisco J. Garijo (Telefónica I+D), Juan Pavón, Carlos Rodríguez y Damiano Spina (Universidad Complutense de Madrid)

Agradecimiento

Financiación Proyecto IntegraWeb (TIN2007-64119, P07-TIC-02602, P08-TIC-4100)

Índice

Prólogo del taller

José L. Álvarez[1], José L. Arjona[1], Rafael Corchuelo[2] y David Ruiz[2]

[1] Universidad de Huelva
Dep. de Tecnologías de la Información
Escuela Politécnica Superior. Ctra. Huelva-La Rábida. Palos de la Frontera 21071
{alvarez,jose.arjona}@diesia.uhu.es
[2] Universidad de Sevilla
Dep. de Lenguajes y Sistemas Informáticos
ETSI Informática. Avda. Reina Mercedes, s/n. Sevilla 41012
{corchu,druiz}@us.es

1 Motivación

Nuestro interés principal es estudiar los problemas relacionados con la integración de aplicaciones web que tan sólo ofrecen una interfaz de usuario. Estas aplicaciones suelen ser fuentes de datos muy valiosas, pero no resulta fácil aprovecharlas a gran escala debido a las dificultades que supone integrarlas dentro de procesos de negocio automatizados.

Desde el punto de vista de la investigación, se trata de un tema que está atrayendo a diversas comunidades: la de Base de Datos, por ejemplo, está prestando mucha atención al desarrollo de lenguajes de consulta estructurados específicos para la Web; la de Inteligencia Artificial, está más centrada en el diseño de herramientas que permitan extraer la información de las páginas web y dotarla de un significado bien definido usando ontologías; la de Minería de Datos también ha mostrado interés por este problema y centra su esfuerzo en el desarrollo de técnicas de verificación de información; etcétera. Sin duda alguna, creemos que una de las comunidades que más puede beneficiarse de todos estos avances multidisciplinares es la de los Servicios Web, ya que las aplicaciones que tan sólo ofrecen una interfaz de usuario suelen suponer una complicación en muchos casos insalvable a la hora de diseñar un proceso de negocio basado en estándares como, por ejemplo, BPMN/BPEL. Para la comunidad de Ingeniería del Software, este tipo de aplicaciones también supone un reto interesante ya que hasta el momento no existe ninguna metodología adaptada a este tipo de problemas.

Zoco'09 continúa una andadura que comenzó en 2001 con el objetivo de ofrecer un foro apropiado a los investigadores interesados en el desarrollo de aplicaciones de negocio en la Web. En esta edición ha pretendido ser un punto de encuentro multidisciplinar en el que diversas comunidades hayan podido discutir propuestas relacionadas con la integración de aplicaciones web.

2 Contribuciones

En total han sido trece las contribuciones aceptadas en esta edición del taller. En su conjunto creemos que proporcionan una visión bastante amplia del trabajo

que se está realizando en el campo de la integración en distintos grupos de investigación y, en algunos casos, en empresas de nuestro entorno.

1. El primer trabajo, de Eduardo Martín Rojo y Vicente Luque Centeno, de la Universidad Carlos III de Madrid, presenta un modelo de anotación para sitios de la Web Profunda que pueda ser utilizado para la extracción de información. Estas anotaciones permitirán la creación de Wrappers Web más adaptables a los posibles cambios estructurales del sitio web accedido.
2. Hassan A. Sleiman, de la Universidad de Sevilla, presenta un framework de comparación de extractores de información de la web semi-estructurada, este tipo de frameworks reduce los costes de análisis y de construcción de nuevos extractores, facilitando la integración de información de la web en los procesos de negocio.
3. Patricia Jiménez, de la Universidad de Huelva, presenta un conjunto de mejoras para aplicar al algoritmo FOIL cuando se emplea en extractores de información de páginas web semi-estructuradas.
4. Carlos G. Figuerola, José Luis Alonso Berrocal y Angel Zazo Rodríguez, de la Universidad de Salamanca, describen algunas de las actividades del grupo de investigación REINA en torno a la recuperación de información web. Estas actividades se han centrado en probar la capacidad de recuperación que puede esperarse de diversos elementos informativos presentes en las páginas web, además del texto que el usuario visualiza normalmente en su navegador.
5. El cuarto trabajo presentado por Inma Hernández, de la Universidad de Sevilla, estudia las diferentes propuestas que existen actualmente para navegar por páginas web de forma automática, además, se presenta un estudio experimental que demuestra que las técnicas actuales son ineficientes a la hora de navegar.
6. Paula Montoto, Alberto Pan, Juan Raposo, Fernando Bellas y Javier López, de la Universidade da Coruña, proponen una serie de técnicas para tratar la navegación web en aplicaciones complejas basadas en AJAX. Aunque las herramientas existentes tratan páginas web con AJAX, dichas herramientas presentan limitaciones significativas en su usabilidad y la habilidad para tratar con páginas web complejas.
7. M. Mercedes Martínez-González, Beatriz Pérez-León, de la Universidad de Valladolid, y M. Luisa Alvite-Díez, de la Universidad de León, afirman que el uso de estándares asociados a la Web Semántica no es suficiente para garantizar totalmente la integración de las herramientas conceptuales utilizadas. Demuestran dicha afirmación en el caso de los sistemas de información jurídica.
8. Ana Flores Cuadrado, de Telefónica Investigación y Desarrollo, Eduardo Villoslada de la Torre, de Telefónica Investigación y Desarrollo y la Universidad de Valladolid, y Alberto Peláez Gutiérrez, de Telefónica Soluciones, definen un algoritmo que permite materializar la estructura conceptual de la Wikipedia y sus relaciones para construir un tesauro.
9. Ismael Navas-Delgado, Amine Kerzazi y José F. Aldana-Montes, de la Universidad de Málaga, abordan el diseño e implementación de una extensión

de WSMO Studio para permitir el diseño de anotaciones de servicios mediante OWL-S. De esta forma, los desarrolladores de servicios Web Semánticos pueden hacer uso de un mismo entorno para producir servicios anotados con ambas propuestas.

10. María Pérez, Ismael Sanz, María José Aramburu y Rafael Berlanga, de la Universitat Jaume I de Castelló, presentan una técnica para seleccionar métodos de ranking apropiados para fuentes de datos complejas usando transformaciones de modelos.
11. Ismael Caballero, Mª Ángeles Moraga y Coral Calero, de la Universidad de Castilla-La Mancha, plantean algunos de los retos de investigación que están afrontando, como por ejemplo la posibilidad de aplicar los principios de LinkedData para anotaciones de calidad de datos, o cómo aplicar los resultados que obtenidos a los datos almacenados en bases de datos heredadas.
12. Juan A. Fraile, Javier Bajo, de la Universidad Pontificia de Salamanca, y Juan M. Corchado, de la Universidad de Salamanca, presentan un sistema multiagente que facilita la realización de tareas diarias para personas en un entorno con conocimiento del contexto.
13. Francisco J. Garijo, de Telefónica I+D, Juan Pavón, Carlos Rodríguez y Damiano Spina, de la Universidad Complutense de Madrid, presentan las experiencias y resultados obtenidos para ofrecer sofisticados Business Support Systems en pymes.

3 Comités

4 Programa

A continuación aparecen los nombres y afiliaciones del comité de programa y de organización de este taller. Damos las gracias a todos ellos por su apoyo y contribución desinteresada para conseguir que esta edición sea un éxito.

4.1 Académico

Joaquín Adiego, Universidad de Valladolid
José F. Aldana, Universidad de Málaga
César J. Acuña, Universidad Rey Juan Carlos
Rafael Berlanga, Universidad Jaume I
Fernando Bellas, Universidad de la Coruña
Ismael Caballero, Universidad de Castilla-La Mancha
Juan M. Corchado, Universidad de Salamanca
Carlos G. Figuerola, Universidad de Salamanca
Vicente Luque, Universidad Carlos III de Madrid
Mercedes Martínez, Universidad de Valladolid
Juan Pavón, Universidad Complutense de Madrid
Miguel Toro, Universidad de Sevilla

4.2 Empresa

Pablo Adanero, Indevia
Ana Flores Cuadrado, Telefónica I+D
Javier López Mato, Denodo Technologies
Alberto Pan, Denodo Technologies
Antonio J. Sáenz, Isotrol

5 Organización

José L. Álvarez, Universidad de Huelva
José L. Arjona, Universidad de Huelva
Rafael Corchuelo, Universidad de Sevilla
Iñaki Fernández de Viana, Universidad de Huelva
Inma Hernández, Universidad de Sevilla
Carlos Rivero, Universidad de Sevilla
David Ruiz, Universidad de Sevilla
Hassan A. Sleiman, Universidad de Sevilla

Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 3, No. 6, 2009

Extracción de Datos de Sitios de la Web Profunda Anotados Semánticamente

Eduardo Martín Rojo, Vicente Luque Centeno

Universidad Carlos III de Madrid
Av. Universidad 30, 28911
Leganés (Madrid), España
{emartin,vlc}@it.uc3m.es

Resumen La navegación y recolección de información de forma automática de páginas de la Web Profunda requiere de la utilización de Wrappers Web que permitan simular la interacción realizada por usuarios humanos; sin embargo, este tipo de aplicaciones poseen algunos problemas debido a que su implementación posee una fuerte dependencia con la estructura del sitio web accedido y a que su código fuente requiere de un mantenimiento constante.

En este trabajo proponemos un modelo de anotación para sitios de la Web Profunda que pueda ser utilizado para la extracción de información. Las anotaciones realizadas con nuestro modelo se expresan desde el punto de vista del cliente permitiendo a usuarios finales y a terceras partes el desarrollo de sus propias anotaciones para cualquier sitio web. Esta anotaciones permitirán la creación de Wrappers Web más adaptables a los posibles cambios estructurales del sitio web accedido.

1. Introducción

Actualmente los sitios Web más populares proporcionan a sus usuarios de herramientas de desarrollo que permiten la creación de aplicaciones capaces de acceder a algunas de sus funcionalidades (por ejemplo, *eBay Developers Program*[1]); y frecuentemente dichas funcionalidades son ofrecidas por medio de servicios Web que permiten ser accedidos con aplicaciones para *mashups* como *Google Mashups*[2], *Yahoo Pipes*[3] or *Microsoft Popfly*[4]. Sin embargo, la inmensa mayoría de sitios Web están enfocados a ser manejados únicamente por usuarios humanos, con lo que no proveen de este tipo de herramientas. El acceso a este tipo de sitios se realiza por medio de los llamados *Wrappers Web*[5].

Algunos de los principales problemas de los Wrappers Web son: en primer lugar, requieren que los desarrolladores posean un alto nivel de conocimiento de la

[1] http://developer.ebay.com/
[2] http://www.googlemashups.com/
[3] http://pipes.yahoo.com
[4] http://www.popfly.com

estructura del sitio Web accedido dado que los Wrappers son soluciones específicas para cada sitio Web concreto; y en segundo lugar, los Wrappers Web requieren de un contante mantenimiento que les permita ser capaces de adaptarse a los cambios futuros. La evolución de las herramientas de desarrollo para Wrappers Web se ha dirigido a intentar resolver estas debilidades y al mismo tiempo hacer posible una integración de datos Web de fuentes heterogéneas de una manera más fácil. Algunos ejemplos de herramientas de desarrollo de Wrappers Web son las herramientas para creación de soluciones específicas como *GreaseMonkey*[5] y *Ubiquity*[6], las herramientas de usuario final que permiten la utilización de ciertos elementos de lenguaje natural como *Chickenfoot*[7], o también herramientas gráficas como *OpenKapow RoboMaker*[8].

La principal característica común a todas las soluciones de desarrollo de los Wrappers Web en la actualidad es que, aunque proveen de un manejo sencillo, siempre implican un tipo de desarrollo con fuerte dependencia respecto de la estructura del sitio Web.

En el trabajo que presentamos a continuación hemos definido una formalización que permite anotar semánticamente la estructura de un sitio Web para representar la navegación a través del sitio. Estas anotaciones poseen a su vez referencias que permiten localizar los fragmentos de información más importantes del sitio Web durante la navegación; utilizando las actuales herramientas de desarrollo de Wrappers Web nuestras anotaciones permitirán la implementación de Wrappers basados en el modelo del sitio web que eviten el solapado con la estructura del sitio. Además, el mantenimiento requerido por los cambios producidos en el sitio Web quedarán aislados dentro de la capa del modelo.

En nuestro trabajo hemos elegido utilizar el punto de vista del cliente para la elaboración de los modelos de navegación de los sitios Web debido a que:

- El punto de vista del cliente permite a terceras partes y usuarios finales participar y colaborar en la creación de anotaciones sobre cualquier sitio Web, ya que no requieren acceder al servidor.
- No es intrusivo; no requiere cambiar el sitio Web que está siendo anotado como lo requerirían otros métodos de anotación como RDFa[9] o Microformats[8].

2. Trabajos Relacionados y Contribución

La generación del modelo de navegación de Web Profunda es tratado por [10], donde los auotres generan un modelo de navegación donde utilizan búsqueda de palabras clave para identificar las diferentes páginas de la Web Profunda. Otro trabajo, más enfocado a la interacción con sitios de la Web Profunda es *Trascendence*[1], un sistema que permite a los usuarios generalizar sus consultas sobre un formulario de búsqueda para poder expandir su espacio de búsqueda.

[5] http://www.greasespot.net/
[6] http://ubiquity.mozilla.com/
[7] http://groups.csail.mit.edu/uid/chickenfoot/
[8] http://openkapow.com/

Trascendence permite además definir patrones para extracción de datos, lo que permite combinar los datos obtenidos con otras fuentes de datos.

Uno de los principales problemas de la Web Profunda reside en que una página no siempre puede ser representada de forma unívoca por medio de una URL[9]. El problema de referenciar páginas de la Web Profunda es mencionado en [6], donde los autores presentan una solución para crear *bookmarks* a páginas de la Web Profunda utilizando una secuencia de pasos especificada por medio de scripts *Chickenfoot* que simulan la interacción requerida por parte del usuario para poder alcanzar la página marcada.

Extraer datos semánticos de sitios Web es un problema que ha sido afrontado con anterioridad en *Marmite*[13], un sistema que proporciona una interfaz al usuario final que le permite diseñar el flujo de procesos necesario para extraer datos; o también en *PiggyBank*[7], un sistema donde el usuario puede hacer uso de scripts llamados *Screen Scrapers* para convertir HTML en datos semánticos RDF.

En nuestro sistema demo hacemos uso de *Chickenfoot* como herramienta de desarrollo de Wrappers. Las principales características de esta herramienta son presentadas en [2] y [3]. *Chickenfoot* permite la especificación de interacciones Web del lado del cliente por medio de una extensión del lenguaje Javascript.

3. Modelo de Anotación de la Web Profunda

El objetivo de un cliente Web es alcanzar un determinado estado por medio de interactuar con la Web. Nuestro modelo representa los estados y transiciones que componen el grafo de navegación que refleja todas las posibles situaciones que pueden ocurrrir en un sitio Web desde el punto de vista del cliente. En el grafo, los vértices representan todos los posibles estados lógicos en que se puede encontrar la aplicación durante la interacción producida por el usuario, mientras que las aristas del grafo representan las acciones producidas que permiten las transiciones entre estados.

Cada estado puede ser dividido en elementos llamados fragmentos, y estos fragmentos a su vez pueden ser dividios en nuevos fragmentos. Un fragmento se identifica por medio de una expresión XPath dentro de un estado, con lo que cada fragmento representa un tipo de contenido semántico que puede ser extraído al utilizar dicha expresión sobre el estado al que pertenece. La figura 1 muestra un ejemplo de división de estados en diferentes fragmentos. Los fragmentos permitirán extraer información semántica a partir de páginas de la Web Profunda si conocemos la localización del estado dentro del grafo de navegación del sitio Web.

Una transición representa las acciones disponibles para el usuario a partir de un estado origen. Estas acciones permitirán cambiar entre diferentes estados en el sitio Web. Las transiciones están compuestas de, en primer lugar, una secuencia ordenada de interacciones que originan el cambio de estado; y en segundo lugar, una lista de todos los posibles destinos que pueden ser alcanzables

[9] Uniform Resource Locator, defined on http://tools.ietf.org/html/rfc1738

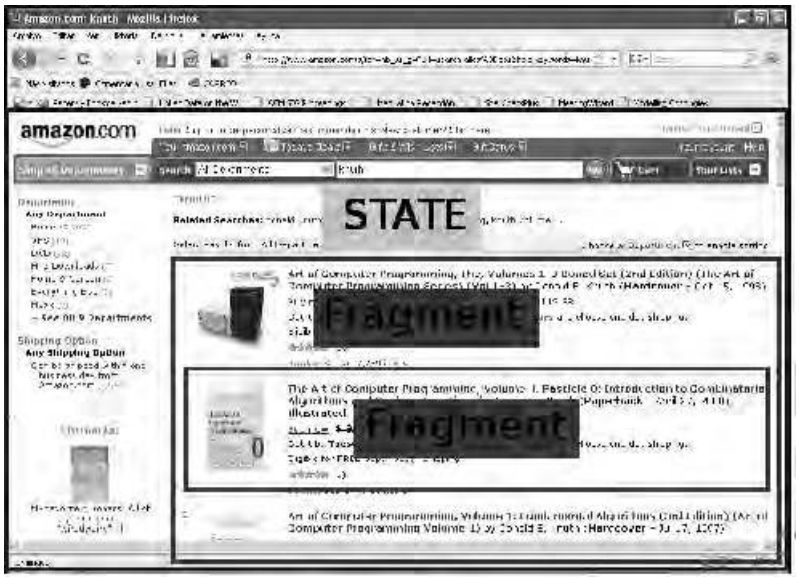

Figura 1. Estados y Fragmentos

por la utilización de estas transiciones. Nuestro grafo de estados y transiciones es **no determinista**, es decir, dado un origen determinado y una transición, pueden existir varios posibles destinos debido a que un sitio Web podría actuar de diferentes formas no predecibles para el cliente. Esto puede producirse como consecuencia de factores dinámicos como el estado de la plataforma que ofrece el servicio, el histórico de interacciones, la fecha y hora, etc.

Nuestro modelo de estados y transiciones para el grafo de navegación está representado por los siguientes tipos de elementos:

1. PageState: un estado unívoco identificable que representa una página de la Web Profunda. Un estado contiene fragmentos.
2. Fragment: representa un elemento dentro de un PageState o de otro Fragment. Esta clase se define como dominio de las siguientes propiedades:
   a) fragmentOf: Define este fragmento como parte de otro fragmento o parte de un PageState dentro de una jerarquía.
   b) locatedBy: una expresión XPath que identifica la posición del fragmento dentro de la representación XHTML del PageState o Fragment con que se enlaza por medio de la propiedad fragmentOf.
   c) semantic: conjunto de triplas RDF que representa el conocimiento referenciado por este Fragment. Esta propiedad está definida también para las clases Input y Action.
3. Transition: representa un conjunto de acciones que permiten el cambio de estado entre un conjunto de estados de origen y un conjunto de posibles estados de destino. Tiene las siguientes propiedades:

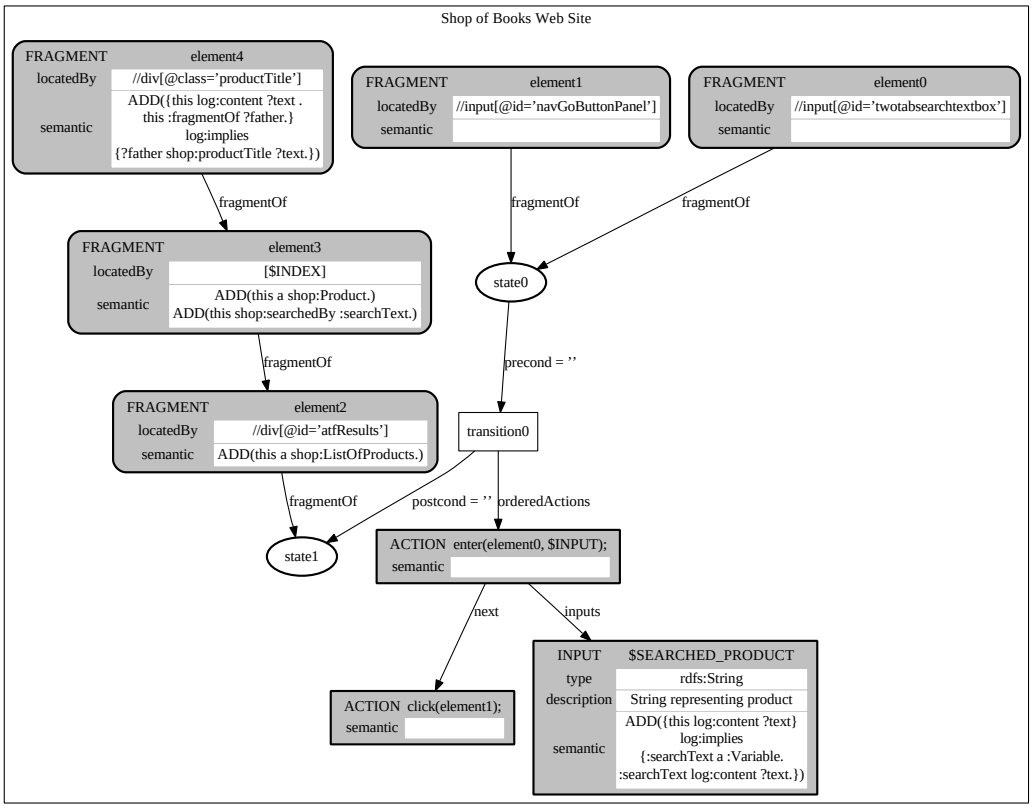

Figura 2. Modelo de navegación anotado para la realización de búsquedas en una supuesta tienda de libros

   a) actions: una transición es originada por una secuencia ordenada de acciones (instancias de la clase Action). Esta propiedad contiene dicha lista de acciones.
   b) sources y destinations: todos los posibles estados que pueden ser fuente o destino de esta transición respectivamente.

4. Action: una interacción que puede ser llevada a cabo sobre un fragmento. Contiene las siguientes propiedades:
   a) hasType: tipo de interacción (clic sobre un elemento, selección de elemento, introducción de datos textuales...). Se representa por medio de un comando *Chickenfoot* en las anotaciones de nuestra demo.
   b) inputs: todos los valores de formulario necesarios para llevar a cabo la interacción son referidos por esta propiedad como una lista que contiene elementos de tipo Input. Para cada entrada Input debe proporcionarse un nombre, un tipo de dato de XML Schema, y una descripción.

Hemos representado esta formalización por medio de una ontología en OWL[10] que puede ser accedida desde [12].

En la figura 2 hemos representado un ejemplo de anotación para una supuesta Web que permite buscar y comprar libros. Desde la página inicial de este sitio representada por el nodo state0, existen dos fragmentos: element0 (una caja de texto para búsquedas) y element1 (un botón que inicia la tarea de búsqueda). Desde state0, es posible ir a state1 a través de la transición transition0. Esta transición requiere desarrollar dos acciones (representadas por la lista de acciones orderedActions): la primera acción es llevada a cabo al insertar una cadena de texto dentro de la caja de texto element0 mientras que la segunda acción se lleva a cabo por medio de la realización de un clic de ratón sobre el botón element1. En el estado state1, existen tres fragmentos que pueden ser accedidos: element2, element3 y element4. Ya que cada uno de ellos es una parte de otro fragmento, la expresión XPath expresada en la propiedad locatedBy es relativa al fragmento o estado padre. Por ejemplo, element4 es construido con las expresiones XPath de element3 y element2, y por eso la localización XPath resultante es //div[@id='atfResults'][$INDEX]//div[@class='productPrice'].

En esta figura hemos utilizado una ontología hipotética para anotar conceptos relacionados con el escenario del ejemplo, pero el modelo de anotación presentado puede ser combinado con cualquier ontología para definir el contenido semántico dentro de los Fragments, Actions o Inputs del modelo. Este conocimiento puede ser proporcionado en formato RDF dentro de la propiedad semantic de estos elementos.

Todo PageState se compone de Fragments y puede ser alcanzado por medio de la realización de acciones (representadas por Action) que pueden utilizar entradas (Inputs). Todos estos elementos poseen un contenido semántico que definen el propio contenido semántico del estado PageState. No obstante, este contenido también se ve influido por la semántica de todas las transiciones que se han seguido y de todos los estados por los que se ha pasado para alcanzar el estado actual. Este conocimiento puede ser transformado durante las transiciones, y nuevo conocimiento pude ser añadido o eliminado del conjunto de aserciones almacenadas en la memoria de trabajo del cliente, dependiendo de las interacciones del cliente con el sitio de la Web Profunda. Es, por tanto, un sistema similar a los denominados Basados en Reglas o Sistemas de Producción en que el conocimiento se va modificando según las acciones anteriores.

Como se puede ver en la figura 2, Fragment, Input y Action han definido la propiedad semantic que indica qué conocimiento (representado como triplas RDF o reglas de inferencia) es añadido (ADD) o borrado (DEL) de la memoria de trabajo. Para añadir una tripla RDF sólo es necesario añadirla directamente a la memoria de trabajo, mientras que añadir una regla lógica en RDF requiere de la ejecución de dicha regla sobre todas la triplas de la memoria de trabajo cada vez que nuevas triplas RDF son añadidas. El hecho de borrar una tripla o una regla simplemente la elimina de la memoria de trabajo.

[10] http://www.w3.org/TR/owl-features/

El efecto de añadir o borrar datos RDF de la memoria de trabajo del cliente sigue las siguientes reglas:

1. Si el cliente se encuentra sobre un estado PageState que contiene fragmentos, la propiedad semantic de los fragmentos es procesada dentro de la memoria de trabajo siguiendo la propiedad fragmentOf que los enlaza entre sí por medio de una jerarquía. Los fragmentos que se encuentran en el mismo nivel de jerarquía pueden ser procesados en cualquier orden (por ejemplo, todos los fragmentos que se encuentran directamente enlazados con un PageState se pueden procesar en cualquier orden).
2. Si el cliente ha viajado a través de una transición Transition que contiene una lista ordenada de acciones, la propiedad semantic de las acciones es procesada siguiendo el orden en que se definió la secuencia de acciones.
3. Si el cliente utiliza una entrada definida en un fragmento de un estado PageState o en una acción de una transición Transition, la propiedad semántica de la entrada es procesada después de todas las otras propiedades semánticas del resto de PageStates y Transitions.

La información semántica se encuentra asociada a cada instancia de las clases Fragment, Action e Input por medio de la utilización de la propiedad semantic. El usuario que está anotando un sitio Web indica qué tipo de información representa el fragmento, acción o entrada por medio de esta propiedad. Su contenido está expresado en RDF.

Para definir información semántica, en la figura 2 utilizamos las siguientes propiedades y conceptos basados en las funciones disponibles para el razonador CWM[11].

Con el objeto de facilitar la compartición y reutilización de anotaciones, es necesario el uso de ontologías para el modelado de este contenido semántico.

4. Extracción de Datos utilizando Anotaciones

El contenido de la propiedad semantic permite localizar una información particular dentro del mapa de navegación para lleva a cabo una consulta que especifica las condiciones en las cuales la información puede ser localizada dentro de la memoria de trabajo. Como nuestro modelo trata con conocimiento expresado por medio de triplas RDF, hemos seleccionado SPARQL[12] como lenguaje de consulta. SPARQL es un lenguaje que permite indicar el *Named Graph*[4] al que un conjunto de condiciones de la consulta se refiere. Hacemos uso de esta funcionalidad para relacionar anotaciones de diferentes sitios de la Web Profunda con el objeto de llevar a cabo una consulta distribuida entre sus grafos de navegación anotados. La figura 3 muestra un ejemplo de consulta que utiliza dos grafos diferentes. La consulta SPARQL requiere el precio de un libro que cumple las siguientes condiciones expresadas en la parte WHERE de la consulta:

[11] http://www.w3.org/2000/10/swap/doc/CwmBuiltins.html
[12] http://www.w3.org/TR/rdf-sparql-query/

1. Seleccionar desde un RSS cualquier elemento identificado como Product que tenga definido un título.
2. Seleccionar desde el sitio web de la tienda de libros cualquier elemento identificado como Product que haya sido buscado en el sitio web utilizando la cadena que representa el título del producto obtenida desde el RSS previamente accedido.

Un conjunto de condiciones expresadas por una consulta SPARQL puede ser satisfecho por una lista de transiciones. Las acciones asociadas a estas transiciones pueden requerir que el cliente proporcione determinadas entradas. Estas entradas a su vez pueden ser necesarias para acceder a fragmentos específicos dentro de un estado. En el ejemplo de la figura 3, el mapa RSS requiere una entrada llamada INDEX para acceder a un fragmento dentro de rss_state0, y el grafo de la tienda de libros requiere de la entrada SEARCHED_PRODUCT para llevar a cabo las acciones de transition0. Las Inputs son los puntos en los que la consulta SPARQL puede relacionar datos entre diferentes mapas de navegación. Debido a que las entradas requieren que se suministre cierta información, la consulta SPARQL puede enfrentarse a las siguientes posibles situaciones:

- Si las condiciones de la consulta especifican el valor de una entrada, entonces utilizar dicho valor.
- Si las condiciones indican que la información requerida por la entrada debe ser obtenida de otro grafo, entonces la consulta debe primero llevar a cabo sus acciones en el otro grafo. Esto ocurre en el ejemplo con el mapa de la tienda de libros en la condición ?product2 shop:searchedBy ?title, ya que el título es referenciado por el mapa RSS.
- Si la información requerida no puede ser inferida, entonces la consulta SPARQL debe utilizar todas las posibles informaciones que podrían ser utilizadas para la entrada. En el ejemplo esto ocurre con la entrada INDEX, que no es proporcionada, con lo que la consulta debe iterar entre todos los posibles valores de entradad.

Hemos desarrollado un sistema demo que puede ser accedido a traves de [11]. Nuestra demo genera scripts de Wrappers Web compuestos de comandos *Chickenfoot*[13] que pueden ser ejecutados en un navegador *Firefox* que tenga instalado el plugin de *Chickenfoot*. *Chickenfoot* [2] es una extensión del lenguaje Javascript que proporciona un conjunto de funciones especiales para la realización de tareas Web tales como realizar clic en un enlace, introducir datos en un formulario HTML, etc.

Los scripts Chickenfoot son generados a partir de consultas SPARQL utilizando las anotaciones expresadas en la formalización que hemos presentado en este artículo. Esta formalización está accesible en OWL en la URL [12]. Aunque con SPARQL no podemos definir tareas demasiado complejas, pensamos que es un buén punto de partida para definir lenguajes más potentes que puedan ser utilizados en un desarrollo de Wrappers Web orientado a semántica.

[13] http://groups.csail.mit.edu/uid/chickenfoot/

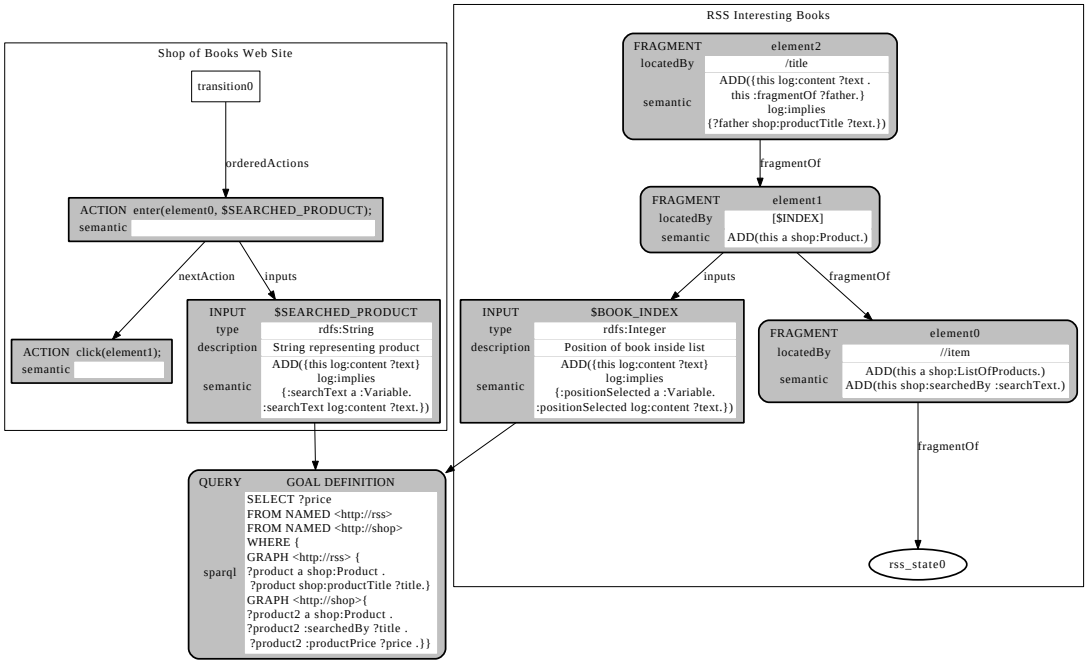

Figura 3. Ejemplo de realización de consultas entre mapas diferentes

5. Trabajos Futuros

Acceder e integrar datos procedentes de fuentes Web heterogéneas no estructuradas es un problema que actualmente se resuelve por medio de la realización de Wrappers Web a pesar de sus carencias. Los Wrappers Web son herramientas que pueden ser utilizadas para estructurar la información obtenida de fuentes no estructuradas y ponerla a disposición de la Web de Datos. Pensamos que el desarrollo de wrappers debe ser adaptado a este nuevo entorno de datos enlazados utilizando definiciones de Wrappers orientados a semántica, e implementaciones no enfocadas al sitio Web concreto. Para lograr esto, estamos interesados en la investigación de lenguajes más potentes y herramientas de desarrollo para esta nueva aproximación de desarrollo de Wrappers semánticos.

Estamos interesados en continuar trabajando en la integración de ontologías con las implementaciones de Wrappers semánticos con el objeto de aprovechar los conceptos comunes entre diferentes sitios web. La misma implementación de Wrapper semántico puede ser utilizada en cualquier sitio web que ha sido anotado utilizando la misma ontología.

6. Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Educación y Ciencia, proyecto ITACA No.TSI2007-65393-C02-01.

Referencias

1. Jeffrey P. Bigham, Anna C. Cavender, Ryan S. Kaminsky, Craig M. Prince, and Tyler S. Robison. Transcendence: enabling a personal view of the deep web. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 169–178, New York, NY, USA, 2008. ACM.
2. Michael Bolin and Robert C. Miller. Naming page elements in end-user web automation. *SIGSOFT Softw. Eng. Notes*, 30(4):1–5, 2005.
3. Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and customization of rendered web pages. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 163–172, New York, NY, USA, 2005. ACM.
4. Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.
5. Sudarshan Chawathe, Hector Garcia-molina, Joachim Hammer, Kelly Irel, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The tsimmis project: Integration of heterogeneous information sources. In *In Proceedings of IPSJ Conference*, pages 7–18, 1994.
6. Darris Hupp and Robert C. Miller. Smart bookmarks: automatic retroactive macro recording on the web. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 81–90, New York, NY, USA, 2007. ACM.
7. David Huynh, Stefano Mazzocchi, and David Karger. *Piggy Bank: Experience the Semantic Web inside your web browser*, volume 5, pages 16–27. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 2007.
8. Rohit Khare and Tantek Celik. Microformats: a pragmatic path to the semantic web. pages 865–866, 2006.
9. W3C. Rdfa primer. W3C Working Group Note 14 October 2008, October 2008.
10. Yang Wang and Thomas Hornung. Deep web navigation by example. In Tomasz Kaczmarek Marek Kowalkiewicz Tadhg Nagle Jonny Parkes Dominik Flejter, Slawomir Grzonkowski, editor, *BIS 2008 Workshop Proceedings, Inssbruck, Austria, 6-7 May 2008*, pages 131–140. Department of Information Systems, Pozna, University of Economics, 2008.
11. WebTLab. Site annotation demo. http://corelli.gast.it.uc3m.es/siteannotation.
12. WebTLab. Site annotation ontology. http://corelli.gast.it.uc3m.es/siteannotation/ontology.owl.
13. Jeffrey Wong and Jason I. Hong. Making mashups with marmite: towards end-user programming for the web. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444, New York, NY, USA, 2007. ACM.

Information extraction from the World Wide Web ⋆

Hassan A. Sleiman

Universidad de Sevilla
ETSI Informática
Avda. Reina Mercedes, s/n. Sevilla 41012 Spain
hassansleiman@us.es

Abstract. The World Wide Web is an enormous and a growing source of information presented in a human friendly language called Html. Unfortunately, querying and accessing this information by software agents is not an easy task, so web information extractors are used. Currently, there is a variety of algorithms to build web information extractors, but none of them is universally applicable. There is not a common software framework to develop them. This has resulted in proposals that range in complexity, precision and recall, but having diverging interfaces, which makes it difficult to reuse or integrate them. As a result, few side-by-side comparisons are available, but none of them is complete. We argue that the key is the absence of a unifying framework in which researchers can develop their proposals so that they can be assessed properly. Devising and implementing such a framework would be an ultimate tool to help reduce costs at integrating web information into automatic business processes. In this paper we report on our first version of this framework for information extractors.

Key words: Information extraction, Enterprise Information Integration.

1 Introduction

The World Wide Web is a growing database in which a great amount of information is present. Unfortunately, querying and accessing these data by software agents is not a simple task since data present in web pages is provided using human friendly formats. This type of codification makes it easier for humans to understand and browse the Web, but makes the incorporation and the use of this data by automated processes very difficult. Some solutions for this problem can be the use of the semantic web [22], which is still a vision, or the use of web services [3], but till these proposals are taken into consideration by all the web sites and are completely specified, one solution is to use information extractors to fill the gap and help us to transform data present in the web to completely structured data usable by automated processes.

There exists a great number of proposals for information extraction algorithms, but unfortunately none of them can be considered as a perfect solution. Information extractors such as [15], [1], [6], [4], [10], [12] and [24], are usually designed and built providing distinct interfaces, thus complicating the task of integration of these algorithms inside enterprise applications.

⋆ Partially funded by the Spanish National R&D&I Plan under grants TIN2008-04718, and the Andalusian Local Government under grants P07-TIC-02602, and P08-TIC-4100.

Suppose we wish to integrate web information into a business process of ours, then one of these tools should be integrated into our enterprise application, but before integrating it, we would like to evaluate existing algorithms to choose the one that fits our case better. One of these three tasks should be performed: study existing surveys that compare information extractors, ask authors for these tools and then work on integrating them into our system, or even develop these algorithms to test them over the web sites we are going to work over. At the end of any of the previous tasks, an information extractor is chosen and integrated, but all of these tasks are tedious and expensive.

The variety of information extractors motivated the existence of many surveys to compare these extraction algorithms and tools such as: [2], [11], [20] and [8]. However, these surveys use taxonomies do not provide new information about the algorithms, and although effectiveness and efficiency are studied, they is not a side-by-side comparison. Besides, the results presented in algorithms' proposals are performed over distinct web sites, so these results are not comparable between each other.

Since no decision can be taken based on the comparison presented by surveys, using information extraction tools for comparison should be considered. We need the tools and the implementations of all the extraction algorithms to test and compare their efficiency and effectiveness. Existing tools are developed using different technologies and different libraries, thus the results of our tests are not reliable since technology conditions are distinct. Also, these tools provide different interfaces and integrating the chosen one into our enterprise application could be time and resource costly. Implementing all extraction algorithms is a very high costing task, user should study deeply a great number of algorithms, and develop them since there is not any software framework to facilitate this job.

We think that the solution to this problem is the creation of an information extraction framework where researchers can develop their proposals and where users can compare these proposals and select the most adequate information extraction algorithm for their case. Our proposal is to create an information extraction framework for semi-structured web pages. This framework reduces the costs of the integration of web information into business processes, reduces costs for developing new extraction algorithms and for testing and comparing these algorithms.

This paper is organised as follows. After showing the motivation in the introduction Section 1, Section 2 introduces information extractors for semi-structured web pages and then section 3 describes our information extraction comparison framework. Section 4 presents the conclusions and our future work.

2 Types and Techniques in Web Information Extraction

We can differentiate between three types of web pages: non-structured, semi-structured and structured web pages. Non structured web pages are those in which the info of interest is within free text. A common property for these types of pages is that given a set of similar non-structured web pages, no common pattern can be induced. In other words, free texts have no common pattern or rule that may help us to extract information from them. For this type of web pages, special information extractors are needed such as [23], [18] and [21]. These information extractors analyse and preprocess text, such as

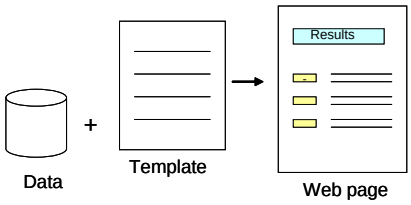

Fig. 1. Semi-structured web page creation.

performing Part of Speech Tagging (POS Tagging) and gaseeters or Entity recognisers, and then identify verbs, adjectives, actions, subjects, places etc.

The second type of web pages is the one that is usually used to show a list of results as a response to a search such as a list of books at amazon.com or any other web store. This type of web pages is typically generated by using a template and a set of data, see Figure 1. In this case, given a set of semi-structured web pages, we can infer one or more patterns that may be used to extract information from this type of web pages. A common objective of many information extractors is to discover the template used to generate these web pages, and use it to detect the data and extract it.

The last type of web pages is the structured type. A structured web page is a web page that besides the presenting information in Html for human browsing, offers structured data that can be processed automatically by machines and are easily integrated into business processes.

Our study focuses on information extraction from semi-structured web pages. As we mentioned before, given a set of semi-structured web pages, information extractors for this type of pages try to extract useful data which is the result returned from the data set before presenting it. Information Extractors apply many techniques in order to separate data from html tags and useless content, such as advertisements. These techniques usually try to detect one or more patterns to identify and extract interesting data from these web pages.

Methods used by information extractors to learn patterns or to extract directly from web pages vary too much. Some information extractors consider web pages as a set of tokens and try to detect repeating patterns using string techniques (for example: [12] and [6]), others use DOM trees and try to detect repeating branches and children in these trees [5], some create and use statistical information to identify repeating zones and extract them directly [16] while others learns prefixes and suffixes for interesting data marked by user to apply them them over new web pages [15].

Besides classifying information extractors by input features, we can consider the the degree of automation for learning as another feature for classification. Information extractors usually learn a set of rules to use them later to extract information. We identified four types of learning algorithms: handcrafted, supervised, semi-supervised and unsupervised. Handcrafted algorithms are written manually for a special type of web pages and can only be used for similar pages having the same structure. Crafting such an information extractor is a hard task and any change in a web page may lead the information extractor to failure. Some information extraction algorithms need the user to

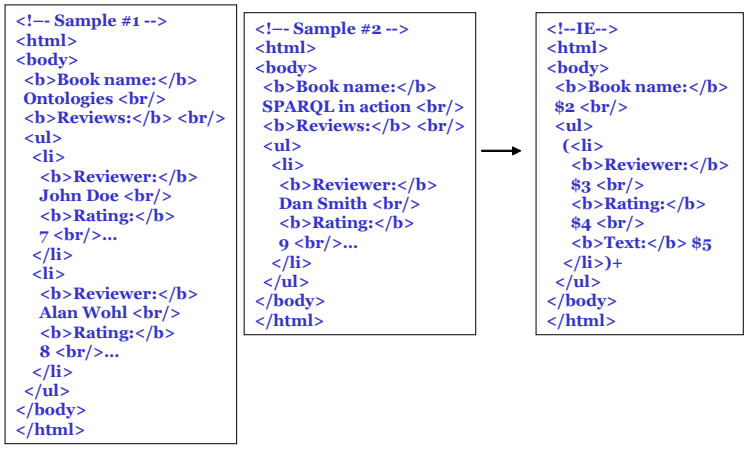

Fig. 2. RoadRunner: an unsupervised information extractor.

mark the important data on some example web pages that are then used in the learning process. It is supervised if the user is required to mark the data to be extracted on sample pages; few authors distinguish between semi-supervised method in which few marking is required, and supervised, in which more marking is required. The last type contains the unsupervised information extractors, in this case, given one or more web pages, the learner automatically and without any user interference, identifies and extracts important information from these webs. An example of an unsupervised information extractor can be seen at figure **??** that induces extraction rules without and user interference.

3 Framework

Here we present the first approximation to our information extraction framework. We identified several modules, each one has a determined task. Some tools are integrated into our framework and are used to prepare web pages for learning extraction rules or even for extracting information from input web pages. These tools are: a tokeniser and an annotating tool [13]. Our tokeniser allows the user to define the tokenisation necessary for his or her extraction algorithm, this gives flexibility to integrate new algorithms into our framework and to define new ones. The annotating tool can be used to mark entities inside a web page and assign them a class or a relation in a predefined ontology. User can configure these tools according to the case of use, this provides high flexibility for developing and using new extraction algorithms. Here we describe the the most important elements:

– Preprocessor: Many information extraction algorithms preprocess web pages before learning or even before extracting information. We have identified two types of preprocessors: Transformer and Segmenter. A Transformer cleans Html code or performs part of speech tagging in case of free text information extractors, for this purpose HtmlTidy [17] is usually used. The segmenter depends on the extraction

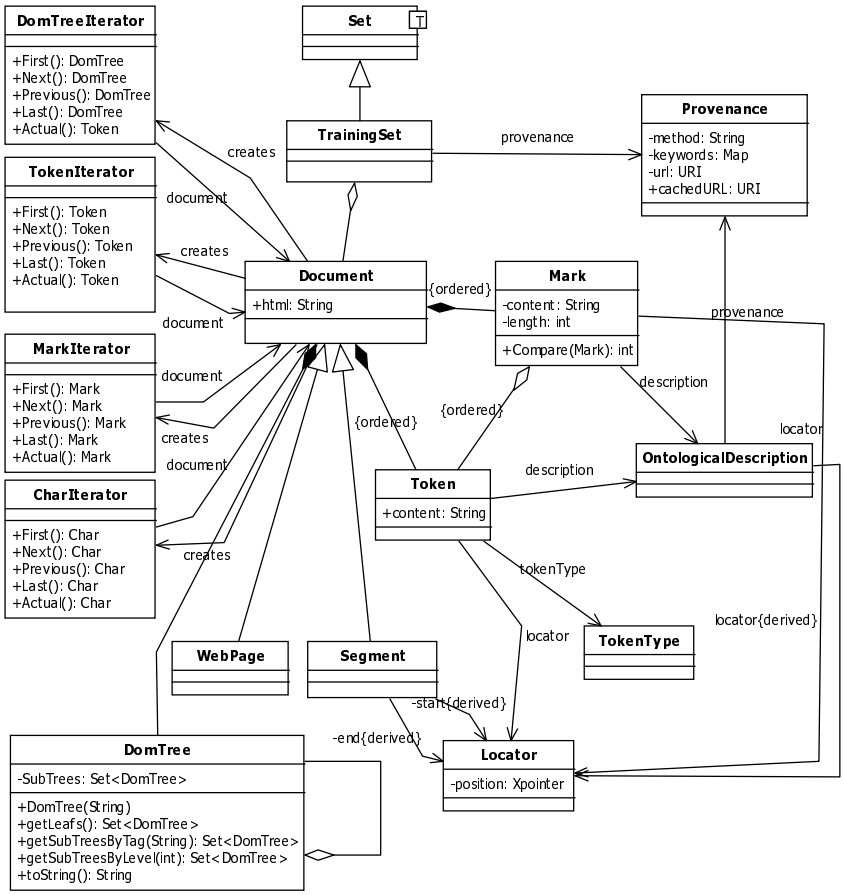

Fig. 3. TrainingSet is composed of a set of Documents.

algorithm. For example, some algorithms create clusters from web pages [14] as a first step, others separate DOM trees and remove unnecessary trees using [7].

– TrainingSet: This structure is the one that contains the web pages used to train information extractors creating extraction rules. See 3. A TrainingSet is a set of Documents which is the main class in this module. A Document is a web page or a segment of a web page. If an algorithm segments a web page before processing it, it creates new segments and adds them to the TrainingSet. The DOM tree of each document is obtained and the DOMTree structure is built. Before working with a Document, tokenisation should be performed using our previously mention tokeniser, creating and adding the sequence of tokens to each document. Besides tokenisation, the user should provide the Marks in the case the information extractor needs them to induce the extraction rules. These marks are added to the document creating instances of the class Mark whose OntologicalDescription is

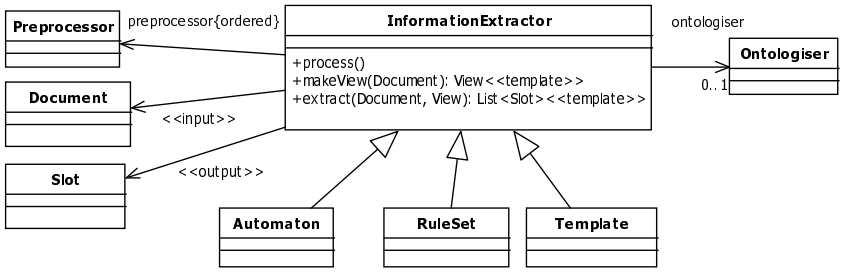

Fig. 4. Information Extractor.

assigned reading the marks ontology description provided by our annotation tool. Once a Document is marked and tokenised, the Learner Module can start working over the training set to discover extraction rules.

The ontologicalDescription class is used to indicate where the ontology that describes an item is, and which class inside this ontology or even which property inside a class this item represents. The Provenance of a web page indicates the URI of an Html file, and the way used to reach and download this web page. The Locator indicates the position of an item inside a Document or inside an ontology, Xpointer is for this purpose.

A segment has a Locator that indicates its start and end position in a given Document. The Locator also indicates which class or property inside an ontology provides semantics to a given mark. The TokenType is provided by our tokeniser and returns the type of the token depending on the it's configuration file. Iterators are used in order to get documents of a training set, marks or tokens of a certain document or even DOM subtrees of a given Document.

– Learner: Information extractors that learn rules use this module to deduce extraction rules. It provides to learners of extraction algorithms the capability of defining an ordered set of preprocessing steps before starting to extract from the input documents, and a set of classes with utilities such as a class to create clusters, generaliser or specialiser for regular expressions, also a String and a Tree aligner classes. Besides, this module contains template classes where predefined algorithms, such as Branch and Bound, are defined. Researchers should only give a body for some template methods in order to implement their proposals.

  The result of this module is an object of the class InformationExtractor, see Figure 4. An information extractor has an ordered set of preprocessors that should be executed over input documents before starting the information extraction, besides, it contains some template methods that should be defined by each extraction algorithm. Once extraction is performed, an information extractor creates slots that contain the extracted data and uses an ontologiser to identify and label extracted data. We have identified three types of information extractors: Those that work as an automaton such as [11] and [9], others identify local patterns for each element to be extracted, for example the two information extractores: [19] and [15]. The third

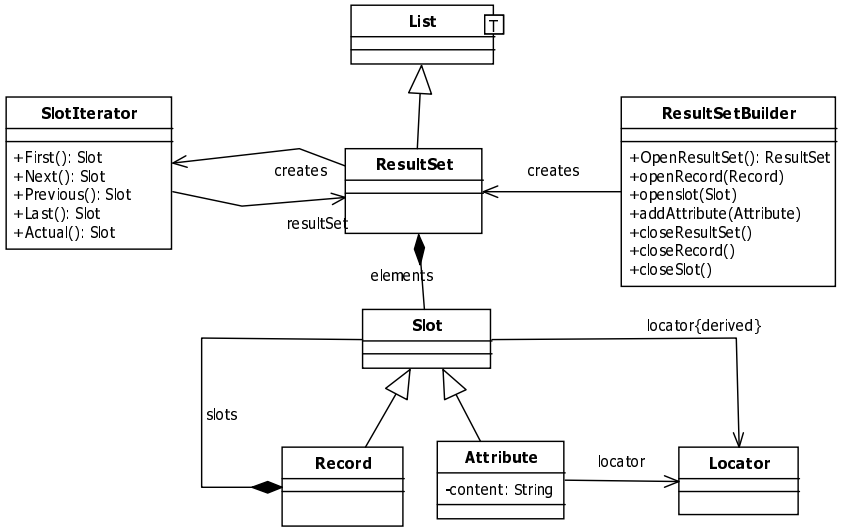

Fig. 5. ResultSet.

type of extractors include those that identify a template for the complete web page such as [24] and [14].

In the case of information extractors that work as an automaton, the created InformationExtractor is an automaton with a set of states and transitions with rules. A transition is performed when its rule can be applied, in this case, a command can be executed. This command indicates the action that should be executed when data is extracted, for example slots containing extracted data can be created and saved. InformationExtractor that learns patterns to identify some elements in a web page create a set of rules. These rules are noniterating rules, since each one is executed over all the document, extracting the zones where it matches, for example, a pattern to detect book titles matches 3 times in a web page and returns three titles. The third type identifies a template and is executed once over the input document, it identifies the element that should be extracted and returns a set of slots.

– WorkingSet: It contains a set of Documents, an InformationExtractor and a ResultSet. When an InformationExtractor is executed, the learned rules in a previous phase are executed over the set of Documents creating a ResultSet that is composed of a set of slots. To build a result set, ResultSetBuilder is used. Then, it uses the ontologiser to identify the classes and the relations between these slots, creating attributes and records then adding them to the ReultSet. At the end of the extraction, the ResultSet contains the extracted entities, each containing its attributes.

4 Conclusions and future work

Although there is a great number of information extraction techniques, none of them is applicable for all cases of information extraction. Users interested in these techniques should make a deep study of all systems before considering which information extraction technique should be used for their purpose. This study is tedious and is a resource consuming task. Besides, once the user has decided which information extractor use, integrating this tool into his business applications needs more effort increasing the web information integration costs.

Our framework contains common parts between the different information extraction algorithm, thus it supports reusability. New extraction algorithms can be implemented using our framework by reusing the developed modules and simply focusing on developing the main algorithm, so it reduces costs to develop new proposals. Testing and comparing information extractors are tasks that could be done using our framework since conditions are equal for all the tested algorithms. Using our information extraction framework, the effort used to compare different proposals is reduced since all algorithms are easily integrated, tested and compared between each others. Definitively, our framework can be considered as an important platform to reduce web information integration costs.

Once the framework is completed, the second step is to create a taxonomy for classifying and comparing information extractors, but this time, more features will be comparable since the recall and precision will be calculated using our framework, under the same conditions. Also, the framework will be used to define new information extractors and compare them with the existing ones.

References

1. Altigran S. da Silva Alberto H. F. Laender, Berthier Ribeiro-Neto. Debye - data extraction by example. *Data Knowl. Eng., Vol. 40*, 2001.
2. Altogran S. da Silva Juliana S. Teixeira Alberto H.F. Laender, Berthier A. Ribeiro-Neto. A brief survey of web data extraction tools. *SIGMOD*, 2002.
3. Casati F. Kuno H.-Machiraju V. Alonso, G. *Web Services Concepts, Architectures and Applications*. 2004.
4. Charles Schafer Andrew Carlson. Bootstrappiing information extraction from semi-structured web pages. 2008.
5. MA David Karger Andrew Hogue. Thresher: automating the unwrapping of semantic content from the world wide web. *ACM*, 2005.
6. Hector Garcia-Molina Arvind Arasu. Extracting structured data from web pages. *SIGMOD*, 2003.
7. Yanhong Zhai Bing Liu, Robert Grossman. Mining data records in web pages. *SIGKDD*, 2003.
8. Moheb Ramzy Girgis Khaled F. Shaalan Chia-Hui Chang, Mohammed Kayed. A survey of web information extraction systems. *IEEE*, 2006.
9. Ming-Tzung Dung Chun-Nan Hsu. Generating finite-state transducers for semi-structured data extraction from the web. 1998.
10. Dawn G. Gregg. Exploring information extraction resilience. *Journal of Universal Computer Science*, 2008.

11. Craig Knoblock Ion Muslea, Steve Minton. A hierarchical approach to wrapper induction. *SIGGRAPH*, 1999.
12. Fred H. Lochovsky Jiying Wang. Data extraction and label assignment for web databases. *Journal of Universal Computer Science*, 2003.
13. Agustín Domínguez Nicolás Griñolo José L.Alvarez, José L. Arjona. Annotator: Herramienta para la anotación semántica de islas de datos amigables en la web. *JISBD*, 2009.
14. Khaled Shaalan Moheb Ramzy Girgis Mohammed Kayed, Chia-Hui Chang. Fivatech: Page-level web data extraction from template pages. *IEEE*, 2007.
15. Robert Doorenbos Nicholas Kushmerick, Daniel S. Weld. Wrapper induction for information extraction. *IJCAI*, 1997.
16. Dimitrios Skoutas Nikolaos K. Papadakis. Stavies: A system for information extraction from unknown web data sources through automatic web wrapper generation using clustering techniques. *IEEE*, 2005.
17. Dave Raggett. *Clean up your Web pages with HP's HTML Tidy*. Elsevier B.V., 1999.
18. Ellen Riloff. Automatically generating extraction patterns from untagged text. *AAAI96*, 1996.
19. Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning, vol. 34*, 1999.
20. Ross Tredwell Stefan Kuhlins. Toolkits for generation wrappers. *Net.ObjectDays*, 2002.
21. Jonathan Aseltine Wendy Lehnert Stephen Soderland, David Fisher. Crystal: Inducing a conceptual dictionary. 1995.
22. Ora Lassila Tim Berners Lee, James Hendler. The semantic web. *Scientific American*, 2001.
23. Sriram Raghavan Shivakumar Vaithyanathan Huaiyu Zhu T.S. Jayram, Rajasekar Krishnamurthy. Avatar information extraction system. *IEEE*, 2006.
24. Paolo Merialdo Valter Crescenzi, Giansalvatore Mecca. Roadrunner: Towards automatic data extraction from large web sites. *VLDB*, 2001.

Optimizando FOIL para la Extracción de Información de la Web*

P. Jiménez, J.L. Arjona, J.L. Álvarez

Universidad de Huelva, Dep. de Tecnologías de la Información,
Escuela Politécnica Superior. Crta. Huelva - La Rábida. Palos de la Frontera 21071
{patricia.jimenez,jose.arjona,alvarez}@dti.uhu.es

Abstract. Para abaratar los costes de las soluciones de Integración de aplicaciones web amigables necesitamos sistemas automáticos que permitan navegar hasta la información de interés, extraerla, estructurarla y verificarla. Los extractores de información son los elementos que permiten extraer la información de la web, y existe mucho trabajo en investigación que tiene como objetivo automatizar su construcción a partir de técnicas de aprendizaje automático. El algoritmo FOIL, ha demostrado ser una buena solución al problema anterior, sin embargo, su ineficiencia en este contexto, imposibilita su uso desde un punto de vista ingenieril.
En este artículo presentamos una serie de ideas originales que tienen como objetivo la optimización del algoritmo FOIL, lo que permitirá su uso, abaratando la construcción de extractores de información usados en soluciones de integración.

Key words: extracción de información, EAI, FOIL

1 Introducción

Las soluciones de Integración de Aplicaciones Empresariales permiten la sincronización y/o colaboración de las distintas aplicaciones que gestionan los procesos de negocio de una empresa. El objetivo de estas soluciones es incrementar los activos de la empresa proporcionando valor sobre la información y procesos ya definidos. Por otra parte, los servicios Web nos proporcionan la tecnología necesaria para llevar a cabo soluciones de integración, siempre y cuando dispongamos de APIs que nos permita acceder a las distintas aplicaciones a integrar. El problema se plantea cuando los costes asociados al proceso de reingeniería inversa para construir dichas APIs no son permisibles (aplicaciones no desmantelables).

En el contexto de la Web, las aplicaciones son concebidas para ser utilizadas por seres humanos, y es habitual que no proporcionen una interfaz programática para acceder a una vista estructurada de la información que gestionan, ni lo que es peor, desmantelarlas. Para integrar estas aplicaciones con otras aplicaciones empresariales, es necesario disponer de un conjunto de APIs que ofrezcan la

* Este trabajo esta parcialmente financiado por el proyecto IntegraWeb - Wrapping (P08-TIC-4100)

posibilidad de interactuar con la aplicación como si de un ser humano se tratara. Estas APIs deberían de aceptar una serie de consultas definidas en un lenguaje estructurado de alto nivel, como SQL o SPARQL[1], rellenar formularios a partir de ellas [1, 2], ejecutarlos y navegar a través de los resultados [2], devolver las páginas que contengan la información de interés, extraer dicha información [4] y dotarla de estructura de acuerdo con la ontología que estemos utilizando [5] y, por último, comprobar que la información obtenida es correcta [6, 7].

Los extractores de información juegan un rol de vital importancia en las soluciones anteriores y este es el motivo, por el que cada vez más, están apareciendo herramientas basadas en algoritmos de aprendizaje automático que tienen como objetivo ayudar a los ingenieros en el desarrollo de soluciones de integración de aplicaciones web amigables. No obstante, no todos los algoritmos de aprendizaje son de utilidad para ser incorporados en herramientas de apoyo al proceso de ingeniería. Algunos, como CLS [9], ID3 [10], C4.5 [11] y CART [12], utilizan lenguajes de especificación poco expresivos a la hora de describir el problema. Otros, como FOIL [13], FFOIL [13] y PROGOL [14], utilizan lenguajes de una gran capacidad expresiva pero presentan problemas de eficiencia que desaconsejan su uso desde un punto de vista ingenieril.

En este artículo, presentamos una propuesta en la que apostamos por la expresividad que nos ofrece FOIL y mejoramos su eficiencia de cara a ser utilizado en soluciones de integración de aplicaciones empresariales. En concreto, presentamos 12 optimizaciones, de las que 6 pueden ser vistas como optimizaciones independientes de su dominio de aplicación, las restantes 6 están centradas en el dominio concreto de la extracción de información en la Web.

El resto de este artículo se organiza de la siguiente manera: la siguiente sección analiza el trabajo relacionado, tanto en lo referente a métodos de aprendizaje inductivo, como a los sistemas de extracción de información de la Web. La sección 3 describe el algoritmo FOIL, la sección 4 presenta las optimizaciones propuestas. Finalmente, la sección 5 presenta las conclusiones y trabajos futuros.

2 Trabajo relacionado

2.1 Métodos de aprendizaje inductivo

En el aprendizaje inductivo se parte de un conjunto de entidades cuyas clases son previamente conocidas y se trata de obtener, a partir de las mismas, una caracterización de cada clase. El aprendizaje en estos sistemas es, a grandes rasgos, el siguiente: dado un conjunto de entrenamiento formado por entidades cuyas clases son conocidas, encontrar una serie de reglas que permitan predecir la clase de una entidad desconocida, en función de los valores de sus atributos. Existen dos diferentes aproximaciones a este proceso.

Los sistemas como CLS, ID3, C4.5 y CART representan estas reglas con un árbol de decisión. Este árbol se construye a partir de un conjunto de entrenamiento mediante un proceso reiterativo. Cada nodo representa un test sobre

[1] http://www.w3.org/TR/rdf-sparql-query/

un atributo, las aristas se determinan en función de los diferentes valores del atributo. El proceso finaliza cuando en un nodo todas las entidades son de la misma clase; en este caso, se convierte en un nodo hoja etiquetado con esa clase. La decisión del atributo a seleccionar en cada test es diferente para cada algoritmo. En CLS un experto determina el atributo que debe usarse en el test. En ID3 se utiliza el concepto de ganancia de información. En C4.5 se usa el concepto de ganancia de información normaliza y en CART se elige al atributo que es capaz de diferenciar en mayor medida a las diferentes entidades. Otros algoritmos de inducción, como los de la familia de algoritmos AQ [16], representan cada clase mediante una expresión lógica disyuntiva. Esta expresión se construye mediante nuevas conjunciones que tratan de caracterizar sólo a las entidades de la clase en cuestión. El proceso termina cuando todas las entidades de un clase se pueden obtener a partir de la expresión lógica disyuntiva.

Existen una serie de algoritmos, como FOIL, FFOIL o PROGOL, que combinan aspectos de las dos aproximaciones anteriores, ofreciendo una mayor expresividad. FOIL es un algoritmo de aprendizaje, derivado de AQ e ID3, cuyo objetivo es inferir un conjunto de reglas, a partir de un conjunto de predicados y de ejemplos utilizando una estrategia top-down. Para una relación objetivo R, FOIL encuentra las cláusulas de una en una, de forma similar a AQ, eliminando los ejemplos cubiertos por la nueva cláusula, antes de inducir la siguiente. Como ID3, FOIL usa una heurística basada en la ganancia de información para guiar esta búsqueda. FFOIL es un variante de FOIL pensada para trabajar con relaciones funcionales. Una relación funcional es aquella donde uno o más parámetros se consideran parámetros de salida y quedan inequívocamente determinados por el resto de parámetros. PROLOG hace frente al mismo problema que FFOIL pero con una estrategia down-up.

2.2 Extracción de información de la Web

Podemos definir un extractor de información como un algoritmo genérico que está configurado con un conjunto de reglas que han sido obtenidas aplicando técnicas de aprendizaje automático. Las reglas permiten identificar, de forma unívoca, fragmentos de información de interés de un sitio web. Existen dos estrategias distintas a la hora de plantear el aprendizaje de las reglas: supervisado y no supervisado.

En los sistemas no supervisados, partimos de un conjunto lo suficientemente representativo de páginas webs no etiquetadas. El sistema trata de identificar de forma automática los elementos comunes entre las diferentes páginas webs y construir, a partir de ellos, las reglas de extracción. Su principal ventaja es la rapidez con que son capaces de aprender; sin embargo, no son capaces de trabajar con páginas que presenten la información en diferentes formatos. Algunos sistemas de este tipo son: Roadrunner [17], DEPTA [18] y EXALGA [19].

En los sistemas supervisados, partimos de un conjunto representativo de páginas web, pero, en este caso, la información de interés está etiquetada. Esos metadatos son usados para inducir las reglas que permiten extraer la información. Aunque no cuentan con la limitación de los sistemas no supervisados,

en lo referente a la alternancia de formatos, son sistemas con un proceso de aprendizaje lento y costoso. Algunos sistemas de este tipo son: SRV, STALKER [20] y WIEN [7].

3 FOIL

FOIL[10, 11, 13] es un algoritmo de aprendizaje de reglas de lógica de primer orden para explicar un predicado objetivo P, a partir de un conjunto de ejemplos positivos y predicados de soporte. Para ello, usa una estrategia muy similar a la de los algorimos de cobertura secuencial y learn-one-rule [23] donde aprende una regla en cada iteración, eliminando las tuplas positivas del conjunto de entrenamiento cubiertas y repitiendo el mismo proceso, hasta que no queden tuplas positivas por cubrir.

Para aprender cada regla, FOIL emplea una estrategia TOP-DOWN, comenzando por la regla más general, la cabecera de la regla, y especializándola mediante la adición de predicados, hasta que la regla no cubra ejemplos negativos. Los posibles predicados se ajustan a una de las siguientes formas: $Q(X_1, X_2, ..., X_k)$ y $\neg Q(X_1, X_2, ..., X_k)$, donde al menos, una de las variables X_i debe existir en la regla; $X_i = C$ y $X_i \neq, C$, donde X_i es una variable ya existente y C una constante; $X_i = X_j$ y $X_i \neq X_j$, donde X_i y X_j son variables ya existentes; por último, $X_i \leq t$ y $X_i > t$, donde X_i es una variable numérica ya existente y t un umbral.

En la elección del siguiente literal a añadir, FOIL se guía de una estrategia *greddy*. En cada iteración, explora el espacio de los literales candidatos de forma casi-exhaustiva para evaluarlos mediante una heurística basada en la ganancia de información. La ganancia de información mide la mejora que se produce sobre la cobertura al añadir un predicado P a la regla en construcción R.

Aunque a priori FOIL es un algoritmo con un gran potencial, adolece de un problema importante: la generación de nuevos predicados supone un gran costo. Por ejemplo, para un predicado de la forma $P(X_1, X_2, ..., X_n)$ con n variables, el número de nuevos predicados a explorar viene determinado por $n*(\binom{n}{m} + n! * m)$, donde m es el número de variables existentes en la regla. Para que FOIL sea más eficiente, debemos centrar nuestros esfuerzos en reducir en gran medida este número.

3.1 Caso de Estudio

Si estamos interesado en extraer los coautores de una página web de la base de datos de publicaciones DBLP (`http://dblp.uni-trier.de`), la salida de un extractor de información construido con FOIL sobre la página de la figura 1 sería: $Coauthor(x) :< Krista\ Lagus,\ Samuel\ Kaski,\ Panu\ Somervuo >$.

Los predicados seleccionados para el ejemplos son los siguientes: $h(x)$: indica que x se encuentra dentro de una etiqueta <h>; $a(x)$: indica que x se encuentra dentro de una etiqueta <a>; $td(x)$: indica que x se encuentra dentro de una

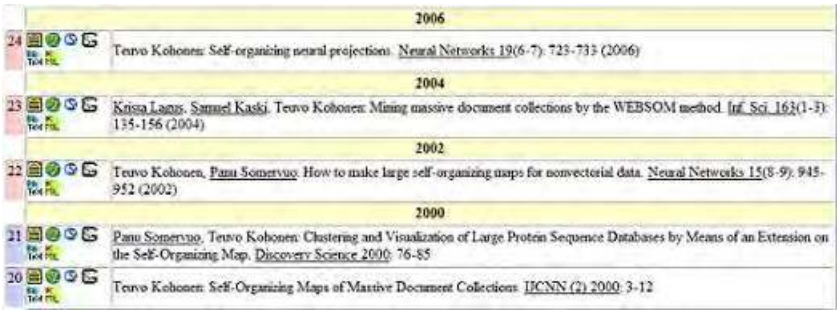

Fig. 1. Página web obtenida de DBLP

celda de una tabla; $previous(x, y)$: indica que el anterior de x es y; por último, $next(x, y)$: indica que el siguiente de x es y.

Además se han definido un conjunto de ejemplos positivos que contienen coautores válidos, y ejemplos negativos con elementos que no son coautores:
$Positivos: \ Krista\ Lagus,\ SamuelKaski,\ Panu\ Somervuo,\ \ldots$
$Negativos: \ Self-organizing\ neural\ projections,\ Neural\ Networks,\ \ldots$

La ejecución de FOIL en forma de árbol para determinar reglas que extraigan los coautores de una página web puede observarse en la figura 2. El número de nodos generados viene determinado por el número de predicados de soporte definidos, mientras que el coste de calcular la ganancia de cada predicado está determinado por el tamaño del conjunto de entrenamiento.

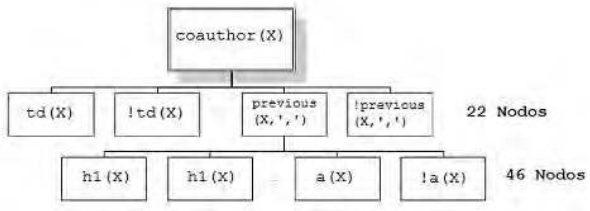

Fig. 2. Árbol de ejecución de FOIL

4 Optimizaciones

Las optimizaciones aplicables a la versión actual de FOIL, pueden clasificarse en dos tipos: independientes del dominio y en el contexto concreto de la Extracción de Información en la Web.

4.1 Independiente del dominio

Test T-Students pareado. Un test T-Students pareado es una fórmula sencilla, que determina si dos nodos estadísticamente pueden considerarse iguales. Dos ramas dentro de un árbol, aunque sean completamente distintas para un experto humano, estadísticamente pueden ser iguales.

Se trata de reducir la expansión de nodos a clases de equivalencia, de manera que, a cada clase pertenecieran todos aquellos nodos que estadísticamente parecen iguales. De esta forma, sólo calculamos la ganancia de un único nodo por clase. El número de literales explorados se reduce al número de clases encontradas.

Asignación de semántica a los predicados. La búsqueda de una definición en FOIL es ciega, de modo que no tiene en cuenta la semántica de los predicados. Existen casos en los que no tendría sentido explorar ciertos literales. Por ejemplo, el literal $previous(X, X)$ es correcto, sin embargo, para un observador humano está claro que no tiene interés ya que es imposible ser el antecesor de sí mismo.

Podemos así añadir para cada predicado de soporte información semántica que permita prescindir de literales que, aunque son válidos, carecen de sentido práctico. Por ejemplo, podemos indicar que los predicados $previous(X, Y)$ y $next(X, Y)$, no tienen sentido cuando las variables X e Y son iguales.

Exploración de n predicados. Una de las formas más simples es ampliar la filosofía greddy a la creación de los nuevos predicados a evaluar. Nuestra propuesta es reducir este número. Se establece así una profundidad límite p, un número de caminos c, y un número de predicados a explorar n. Se llevan de forma simultánea los c mejores caminos de todos los posibles. De cada camino se obtienen los n nuevos predicados y de entre todos ellos se eligen los $c - 1$ mejores y del resto de literales descartados se elige uno al azar para garantizar una cierta capacidad de exploración. Estos predicados constituirán los nuevos caminos a explorar. Una vez que se ha alcanzado una profundidad p (se han añadido p predicados) el algoritmo se centrará sólo en el mejor de todos los caminos que se ha estado considerando.

Selección del primer predicado cuya ganancia supere un umbral. Esta propuesta se basa en generar todos los posibles nuevos predicados y seleccionar el primero con una ganancia que supere cierto umbral, dejando de evaluar el resto de predicados candidatos en ese nivel del árbol de exploración. El problema de esta optimización es la generación de reglas poco expresivas. No obstante, nuestro objetivo es desarrollar herramientas que hagan transparentes las reglas a los usuarios.

Catálogo completo de predicados de soporte. Se trata de disponer de un conjunto útil de predicados de soporte, catalogados por dominio, para ayudarnos

a la hora de enfrentarnos a un nuevo sitio web. Además, dando prioridad a predicados que tras un análisis previo, ya sea experimentalmente o proporcionados por un experto en el dominio se sepan que pueden alcanzar una mayor precisión si se incluyen en la regla.

Determinar si una cadena de predicados está generada por azar. En Lerman [21], se usa el test de hipótesis de Popoulis[22] donde mediante detección de regularidades, es capaz de determinar si una secuencia de texto es fruto o no del azar. Sería interesante incorporar este test a FOIL, ya que así podríamos ser capaces de detectar, antes de evaluar, si la regla en construcción es aleatoria o no, y descartarla en caso afirmativo para centrarnos en otras más prometedoras.

4.2 Dependiente del dominio de Extracción de Información

Uso de los predicados Left y Right. El objetivo de esta optimización es hacer uso de predicados que caractericen el contexto por la derecha y después por la izquierda (o al revés) de la información a extraer. Se pretende explotar la estructura de las páginas Web. A este tipo de extractores de información se les denominan landmark [21].

Un ejemplo de este tipo de extractores son los de Kushmerick[8]. Por tanto, si la información se puede extraer a partir de su contexto, es francamente una buena idea darle prioridad a predicados que tienen como objetivo caracterizar el contexto por la derecha (p.e predicado *next*) y una vez que la ganancia no sea significativa, dar prioridad a predicados que tengan por objetivo fijar el contexto por la izquierda (p.e *previous*).

Chequear si se pueden emplear este tipo de reglas landmark para extraer información de un sitio web, es tan fácil como tomar n tokens por la izquierda y n tokens por la derecha de la información a extraer y aplicar un algoritmo tipo *longest common subsequence* [2]

Priorización de Constantes. Durante la fase de etiquetado de páginas Webs que servirán de entrenamiento al sistema, el experto humano puede observar patrones de tokens que aparecen a izquierda y derecha de la información a extraer. Si esto ocurre, podríamos ordenar las clases de tokens en función de la distancia en la que aparezcan de la información relevante, siendo prioritarias aquellas clases más cercanas a dicha información. De esta manera, en reglas en las que se hayan añadido literales del tipo $next(X, Y)$ o $previous(X, Y)$ los literales más prometedores para evaluar en la siguiente iteración son los predicados predefinidos del tipo $Y = C$ siendo C un token de clase preferente. Incluso si el literal más prometedor fuera $next(X, Y)$, añadir el siguiente más prometedor del tipo $Y = C$ en un solo paso. En el resto de predicados de soporte habría que estudiar su utilidad.

[2] http://en.wikipedia.org/wiki/Longest_common_subsequence_problem/

Predicados que implementan modelos de características de la información. Además de las reglas de tipo landmark, disponemos de reglas de tipo content-based [21]. En este tipo de reglas se explota la estructura de la propia información a extraer. La idea es usar predicados de soporte que hagan referencia a las características de la información a extraer.

Las características pueden clasificarse en dos grandes grupos, características numéricas o características categóricas. Para el ejemplo de la figura 1 una posible regla obtenida por FOIL para extraer todos los coautores es: $coauthor(X) : -a(X), digitDensity(X, 0)$ donde el predicado de soporte $digitDensity$ se refiere a una características numérica de la información a extraer, para el ejemplo de la figura 1 puede generar una regla del tipo $coauthor(X) : -a(X), nounPhrase(X)$ donde el predicado de soporte $nounPhrase(X)$ se refiere a una características lingüistica de la información a extraer. Las ventajas que ofrecen estas reglas es que son menos susceptibles a cambio que las landmark, el problema está en evaluar las características, ya que supone un mayor coste computacional.

Concatenar tokens next y previous. En FOIL, una variable en un predicado puede instanciarse a un token. Si en vez de instanciarse a un sólo token puede instanciarse a un conjunto de ellos, estamos englobando varias iteraciones en una sola. Sería como seleccionar más de un literal candidato en una misma iteración.

Supongamos que los ejemplos positivos que quedan por cubrir son los coautores $Krista\ Lagus$ y $Samuel\ Kaski$. Empezaríamos a generar una nueva regla para cubrir estos ejemplos. Una posible regla de salida que extrayera esta información sin extraer ningún ejemplo negativo sería

$$coauthor(X) : -next(X,' ,'), next(' ,' , Z), next(Z,' ,')$$

Si permitimos la concatenación de tokens, se podría generar una regla del tipo $coauthor(X) : -next(X,' ,' Z' ,')$.

Inducción de patrones. Inducir prefijos y sufijos de la información relevante, puede ser muy útil cuando trabajamos con los predicados de soporte next y previous. Por ejemplo, si disponemos de la tupla ¡P, X, S¿ donde P es el prefijo de X inducido, S es el sufijo de X inducido y X es la información relevante, es apropiado dar asignar prioridad a predicados más específicos del tipo $next(X, Y), Y = S$ o $previous(X, Y), Y = P$ o, de forma simplificada $next(X, S)$ y $previous(X, P)$ respectivamente, porque aportarán una ganancia más alta. Tanto P como S hacen referencia a un token o clase de tokens.

Transferencia de predicados a expresiones regulares. Una vez tenemos las reglas de extracción, acceder a la información puede plantear importantes problemas de eficiencia. En el peor de los casos, tenemos que tokenizar toda la página y a continuación, para cada token, comprobar si se satisfacen todos los predicados de las reglas. En reglas de tipo landmark, este problema queda resuelto traduciendo la regla a una expresión regular, para una página web, podríamos

identificar la información a extraer mediante una expresión xpath. Por ejemplo, si tenemos la regla $coauthor(X) : -next(X, a), previous(X, b), previous(b, c)$ se podría traducir a la expresión regular: $*cb\$a*$, siendo * la clausura de kleene, y $ la información que queremos extraer.

5 Conclusiones

En este artículo hemos presentado 13 ideas originales que tienen por objetivo optimizar su eficiencia computacional de FOIL. Estas optimizaciones permiten la posibilidad de desarrollar APIs programáticas para integrar aplicaciones Web (no desmantelables), abaratando los costes del desarrollo de extractores de información para soluciones de integración.

De entre las optimizaciones propuestas, 6 son independientes del dominio y 6 dependen del contexto de Extracción de Información de la Web. En el primer caso, las optimizaciones sólo son aplicables al núcleo de FOIL, para reducir el número de nodos a evaluar. Mientras que las optimizaciones del segundo tipo, van desde una adecuada definición de la base de conocimiento, hasta la reducción del número de nodos a evaluar en el dominio de extracción de información y la adaptación del conjunto de reglas inducidas de salida.

Todas ellas, se pueden aplicar de forma independiente aunque nuestra propuesta es aplicarla de forma conjunta puesto que no son excluyentes. Para ello, estamos desarrollando un marco de trabajo teórico que nos permita realizar un estudio experimental exhaustivo que determine, en función de un problema, las optimizaciones a usar.

References

1. Raghavan, S., Garcia-Molina, H.: Crawling the hidden web. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2001) 129-138.
2. Pan, A., Raposo, J., Álvarez, M., Carneiro, V., Bellas, F.: Automatically maintaining navigation sequences for querying semi-structured web sources. In: Data Knowl. Eng. 63(3) (2007) 795-810.
3. Lage, J.P., da Silva, A.S., Golgher, P.B., Laender, A.H.F.: Automatic generation of agents for collecting hidden web pages for data extraction. In: Data Knowl. Eng. 49(2) (2004) 177-196.
4. Kayed, M., Shaalan, K.F.: A survey of web information extraction systems. In: IEEE Trans. on Knowl. and Data Eng. 18(10) (2006) 1411-1428.
5. Arjona, J.L., Corchuelo, R., Ruiz, D., Toro, M.: From wrapping to knowledge. In: IEEE Trans. Knowl. Data Eng. 19(2) (2007) 310-323.
6. Kushmerick, N.: Regression testing for wrapper maintenance. In: AAAI '99/IAAI'99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence (1999) 74-79.
7. Kushmerick, N.: Wrapper verification. In: World Wide Web 3(2) (2000) 79-94.

8. Kushmerick N.: Wrapper induction: Efficiency and expressiveness In: Artif. Intell. 118, 15-68, (2000).
9. Hunt, E., Marin, J., Stone P.: Experements in induction. In: New York: Academic Press. (1996).
10. Quinlan J.: Induction of decision trees. In: Machine Learning, 1, 81-106. (1986).
11. Quinlan, J.: C4.5: Programs for Machine Learning. In: Morgan Kaufmann Publishers, 1993.
12. Breiman, L., Friedman, J., Oslhen, R., Stone, C.: Classification and regression tress. In: Belmont: Wadswroth, (1984).
13. Quinlan J.: Learning First-Order Defininitions of Functions. In: Journal of Artificial Intelligence Research, 5, 139-161, (1996).
14. Muggleton, S.: Inverse Entailment and Progol. In: New Generation Computing, Special issue on Inductive Logic Programming, 13, 1995
15. Craven,D.,DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., Slattery, S.: Learning to Extract Symbolic Knowledge from the World Wide Web. In: Proceedings of the 15th National Conference on Artificial Intelligence, (2001).
16. Michalski, R., Mozetic, I., Hong, J., Lavrac, N.: The multipurpose incremental learning sustem AQ15 and its testing application to three medical domains. In: Proccedings of the Fith Nation Conference on Artificial Intelligence, Philadelphia, 1041-1045, (1986).
17. Crescenzi, V., Mecca, G., Merialdo, P.: RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In: Proceedings of 27th International Conference on Very Large Data Bases, 109-118, (2001).
18. Zhai, Y.,Liu, B.: Web data extraction based on partial tree alignment. In: International World Wide Web Conference, 76-85, (2005).
19. Ma, L., Goharian, N., Chowdhury, A., Chung, M.: Extracting unstructured data from template generated web documents. In: Conference on Information and Knowledge Management, 213-215, (2003).
20. Muslea I., Minton, S., Knoblock, C.: Knoblock: Hierarchical Wrapper Induction for Semistructured Information Sources In: Journal of Autonomous Agents and Multi-Agent Systems 4 (1/2), 93-114, (2001).
21. Lerman, K., Minton, S. Knoblock, C.: Wrapper Maintenance: A Machine Learning Approach. In: Journal of Artificial Intelligence Research, 2002.
22. Papoulis, A.: Probability and Statistics. In: Prentice Hall, Englewood Cliffs, NJ, (1990).
23. Mitchell, T.: Machine Learning. In: McGraw Hill, 1997.

Mejoras en la recuperación web combinando campos

Carlos G. Figuerola, José Luis Alonso Berrocal, y Angel Zazo Rodríguez

Grupo de Investigación REINA, Universidad de Salamanca
{figue, berrocal, zazo}@usal.es

Resumen. Este artículo describe algunas de las actividades del grupo de investigación REINA en torno a la recuperación de información web. Estas actividades se han centrado en probar la capacidad de recuperación que puede esperarse de diversos elementos informativos presentes en las páginas web, además del texto que el usuario visualiza normalmente en su navegador. Nuestro objetivo ha sido probar estrategias pre-recuperación de mezclar o combinar esos campos o elementos de información. Combinar términos de diversa procedencia en un único índice puede conseguirse, en sistemas basados en el modelo del espacio vectorial, operando sobre la fecuencia del término en el documento, si se aplica un esquema de pesado basado en $tf \times idf$. El campo `BODY` es, obviamente, el más potente desde el punto de vista de la recuperación; pero los `ANCHOR`s de los *backlinks* que reciben las páginas indizadas añaden una mejora considerable a los resultados de la recuperación. El contenido de las etiquetas `META`, sin embargo, contribuyen poco a la mejora en la recuperación.

Palabras clave: Recuperación de información, recuperación web, modelo vectorial

1 Introducción

Este artículo describe los trabajos efectuados por el grupo de Investigación REINA en torno a la posible mejora de los resultados en la recuperación de páginas web en base a la consideración de diversos campos o elementos de dichas páginas. La colección de documentos utilizada como base ha sido la conocida como EuroGov, utilizada ampliamente en trabajos experimentales de este tipo [10]. En realidad, nuestros trabajos han utilizado la parte de documentos que están en castellano; tanto los que pertencen al dominio .es, como los que, perteneciendo a otros dominios diferentes, están tambien en esta lengua.

De otro lado, muchos documentos pertenecientes al dominio .es están en otras lenguas. Además de las otras lenguas que también se hablan en España (catalán, euskera, gallego), muchas de las páginas tienen versiones o traduciones internacionales; especialmente en inglés, pero también en otras como francés o alemán. Adicionalmente, y por parecidas razones, páginas pertencientes a otros dominios de EuroGov tambiés estaban en castellano, por lo que se decidió explorar la colección entera recolectando todas las páginas cuyo idioma fuera el castellano.

Lamentablemente, las cabeceras de las páginas web ofrecen información muy poco fiable; tanto sobre el idioma de las mismas como sobre otras cosas. En muchas ocasiones esas cabeceras están vacías; mientras que en otros tienen contenidos que no se corresponden con la realidad. Así pues, tuvimos que recurrir a un detector de lenguaje para seleccionar las páginas en castellano existentes en la colección EuroGov. El detector utilizado es *TextCat* [7], un programa basado en la construcción de patrones de uso de *n-gramas* para cada lengua contemplada; y la posterior categorización del texto cuya lengua se desea averiguar [2].

Por lo que se refiere a las consultas, se utilizaron las elaboradas en castellano durante las campañas CLEF 2005 y 2006 [11], junto con los correspondientes juicios de relevancia.

Este artículo está organizado de la siguiente manera: en la próxima sección se describe el enfoque adoptado para abordar el problema; en la sección siguiente se exponen los problemas

y las opciones adoptadas en lo que se refiere a análisis léxico y extracción de términos de las páginas. A continuación, se discute acerca de los campos o elementos de información útiles en la indización y recuperación. Más adelante se muestran los experimentos llevados a cabo y sus resultados. Finalmente, se extraen unas conclusiones.

2 Nuestro enfoque

La estrategia básica seguida es: primero, encontrar las páginas más relevantes para cada consulta y, de acuerdo con el tipo de consulta, reordenar la lista de los documentos más relevantes recuperados. Así, nuestro trabajo tiene dos partes fundamentales: encontrar páginas relevantes y después ordenar de una forma útil esas páginas. La tarea de recuperación de las páginas relevantes para una consulta determinada puede abordarse mediante un sistema convencional de indización y recuperación, como los basados en el modelo del espacio vectorial [9].

Sin embargo, en una página web hay, además del texto que el usuario ve en la ventana de su navegador, otros elementos informativos que pueden resultar interesantes desde el punto de vista de la recuperación. Incluso dentro del propio texto podemos encontrar ciertas estructuras que pueden ayudar, junto con esos otros elementos informativos, a mejorar los resultados de la recuperación.

Por ejemplo, algunos de esos otros lementos mencionados son el campo `TITLE`, algunas etiquetas `META`, los anclas de los *backlinks*, etc. Dentro del campo `body`, que es lo que se visualiza en la ventana del navegador, podemos diferenciar partes que usan diferentes tipografías, por ejemplo. Así, tenemos diferentes fuentes de información que podemos fusionar. En este sentido, se han propuesto dos estrategias básicas de fusión: fusión o combinación pre-recuperación y post-recuperación [1], [6]. Ambas estrategias han sido probadas; la estrategia post-recuperación consistió en construir un índice para cada elemento de información a combinar. Cuando hay que resolver una consulta, ésta es ejecutada contra cada uno de los índices construídos; los resultados obtenidos con cada índice son fusionados después en una única lista de documentos recuperados.

También hemos probado la estrategia pre-recuperación, que se ha concretado el la elaboración de un único índice con los términos de todos los elementos o campos contemplados; pero pesado de manera diferente. Una vez construído este único índice, las consultas son ejecutadas normalmente contra él. Naturalmente, en la elaboración de este único índice podemos buscar dar más o menos valor a un término en función de que aparezca en un campo o en otro. Esto nos permite utilizar una combinación que, bien afinada, debería proporcionar buenos resultados en la recuperación.

La aplicación de diferente peso a los diferentes componentes de la mezcla, en nuestro caso, es fácil. Para la indización y recuperación utilizamos nuestro software *Karpanta* [4], basado en el conocido modelo vectorial, y, en consecuencia, no tenemos más que operar sobre la frecuencia de cada término en cada uno de los componentes de la combinación.

En este caso hemols aplicado un esquema de pesado `ATU` (slope=0.2) [12], pero la idea es similar para cualquier esquema de pesado basado en $tf \times idf$. Podemos aplicar un multiplicador a tf en función del campo en que aparezca el término, de manera que haga aumentar o disminuir el peso de dicho término, sin dejar de considerar la frecuencia del término y el IDF.

2.1 Análisis léxico y extracción de términos

La primera opreación, previa a cualquier otra, es la conversión de las páginas web a texto plano. Esto también era necesario para detectar la lengua de cada página, dado que se decició trabajar sólo con las que estaban en castellano.La obtención del texto plano no es

trivial, y no está exenta de problemas. Como se ha comentado antes, no se puede confiar que en que se sigan los estándares en todos los casos; ni siquiera es posible asegurar que el contenido sea HTML, aunque el documento comience con las etiquetas apropiadas. En algunas ocasiones podemos nos podemos encontrar directamento con código binario, PDFs y similares inmediatamente después de un `<HTML>`. Con las etiquetas `META` sucede lo mismo; incluso cuando están presentes, no siempre ofrecen información correcta.

De hecho, muchas páginas ni siquiera contienen texto; así que lo primero es determinar el tipo de contenido. El viejo y bien conocido comando *file*, bien afinado, puede ayudar a determinar el contenido real de cada página. Adicionalmente, nos informará sobre otra cuestión importante: el sistema de codificación de caracteres utilizado. Para páginas en castellano, lo habitual es *ISO 8859* o *UTF-8*; es necesario conocer esta información para tratar adecuadamente los caracteres especiales. La conversión a texto plano, cuando el documento parece ser realmente HTML, la efectuamos con *w3m*. Hay otros conversores, pero después de varias pruebas los mejores resultados parecen ser los de *w3m*.

Una vez obtenido el texto plano, podemos determinar el idioma mediante *TexCat*. Seleccionados los documentos en castellano, es preciso extraer términos y normalizarlos en alguna manera. Básicamente, los caracteres son convertidos a minúsculas, los acentos son eliminados, al igual que números, signos ortográficos y similares. Se eliminan también palabras vacías, de acuerdo con una lista estándar para el castellano; y los demás términos se pasan a través de un s-stemmer mejorado [5].

2.2 Campos utilizados

Son varios los elementos o fuentes de información que podemos considerar en una página web. La base es el campo `BODY`, obviamente, pero además podemos usar el campo `TITLE`, que parece claramente descriptivo, al igual que varias etiquetas `META` que habitualmente son utilizadas para estos propósitos, en especial `META content="Description"` and `META content="Keywords"`. Sin embargo, como ya hemops indicado en otro lugar ([3]), estos campos no están presentes de una manera uniforme en todas las páginas. Muchas no los contienen y otros tienen valores `META` generados de manera automática por los programas que han producido esas páginas web.

En otros casos, aunque los valores sean asignados de manera manual por los autores de las páginas, resultan poco operativos. Adicionalmente, podemos pensar que términos que aparecen con tipografía resaltada pueden ser más representativos. Las etiquetas o campos `H1`, `H2`, etc. son un ejemplo. Desgraciadamente, el uso de estas etiquetas tampoco es uniforme, y ha evolucionado hacia la definición de fuentes y tamaños específicos de letra, más complejos de procesar.

Otro elemento importante es el de los anclas de los *backlinks*, o enlaces desde otros lugares a la página en cuestión. De manera más o menos breve, estos anclas describen la página con la que conectan; esta descripción es importante, porque está hecha por alguien diferente del autor de la página que queremos indizar. Podemos pensar que esta descripción añade términos distintos de los usados en la propia página. No obstante, muchos de estos anclas pueden hacer referencia a partes muy concretas de la página apuntada; igualmente, hay páginas con muchos *backlinks* y anclas y otras con pocos o ninguno. Y, en cualquier caso, tenemos el problema de obtener esos anclas. En nuestro caso, hemos procesado la totalidad de la colección EuroGov para obtener todos los enlaces y sus correspondientes anclas que apuntan hacia páginas en castellano. Es evidente que fuera del la colección EuroGov hay más enlaces que apuntan hacia dichas páginas, pero no tenemos forma de obtenerlos.

Finalmente, hemos construido índices con los siguientes elementos: `BODY`, `TITLE`, `META content=Title`, `META content=description`, `META content=keywords`, `H1`, `H2`, `ANCHORS`.

3 Experimentos

Apoyándonos en las estimaciones de relevancia que forman parte de la colección de documentos, hemos efectuados diversas pruebas con varias combinaciones de elementos o campos en distintas proporciones. Hemos elegido como línea base para comparaciones los resultados con un índice formado exclusivamente por los términos que aparecen en el campo `BODY`.

`BODY`(fd=1)
`ANCHOR`(fd=1)
`TITLE` (fd=1.5)
`META-DESC` (fd=1.5)
`META-TITLE` (fd=1)
`META-KEY` (fd=0.5)
`H1` (fd=0.8)
`H2` (fd=0.8)

Tabla 1. Combinación ganadora

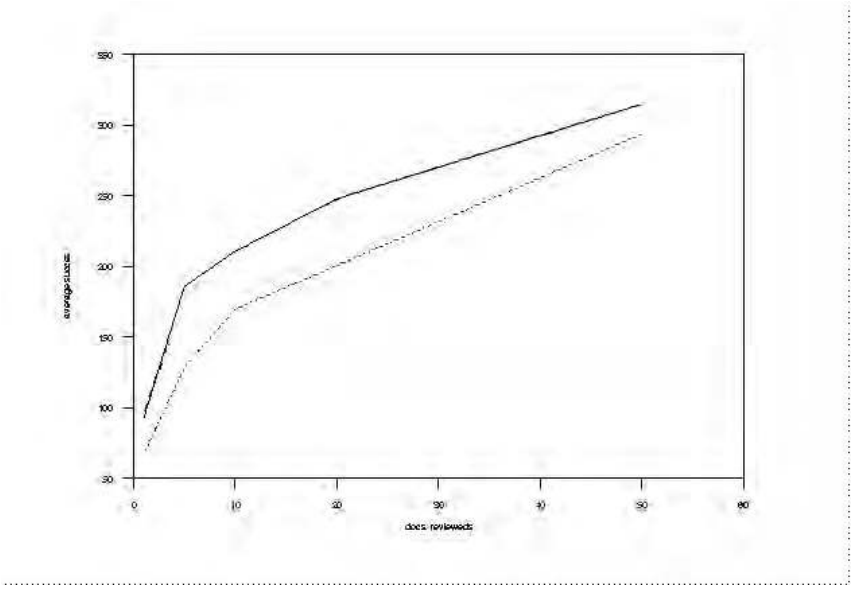

Fig. 1. Resultados con campo `BODY` y con Combinación Ganadora

Como era de esperar, el uso de cualquiera de esos elmentos en adición al campo `BODY` mejora los resultados de la recuperación. Pero algunos de esos elementos lo hacen de forma notable y otros no tanto.

También hemos efectuado pruebas con índices compuestos por uno solo de cada uno de los campos. Esto nos permite observar la utilidad, desde el punto de vista de la recuperación, de cada uno de esos campos. El gráfico muestra los resultados; cada curva representa los obtenidos tras ejecutar la batería de consultas contra un índice elaborado con los términos de cada campo o elemento de información ($fd = 1$). Cada curva se refiere a uno solo de esos campos, sin utilizar los términso del `BODY`. Los campos `H1` y `H2` no aparecen en el

gráfico porque producen resultados extremadamente pobres. Es evidente que muchas páginas carecen de uno o varios de estos campos, razón por la cual nunca podrán ser recuperados a partir de esos campos; pero es una de verificar por separado la capacidad de recuperación de esos campos.

Como era esperable, cada campo por separado produce peores resultados que `BODY`, lo cual es normal, pues este campo contiene la parte visualizable de la página web. Pero tras el campo `BODY` el campo que parece más interesante desde el punto de vista de la recuperación es el de los `ANCHOR`s de los *backlinks*, aunque en muchos casos tales anclas sean muy breves. Pero parece que, como se ha dicho, las descripciones que otros hacen de una página son muy eficaces para su recuperación.

Le sigue a corta distancia el campo `TITLE`, algo previsible. Sin embargo, los campos basados en etiquetas `META` ofrecen resultados bastante pobres, a bastante distancia de `ANCHOR` y `TITLE`; y eso a pesar de que son campos concebidos específicamente para la recuperación. Aunque hay poca diferencia entre los resultados de los tres `META` observados (`title`, `description` y `keywords`) es éste último, curiosamente, el que peores resultados produce de los tres.

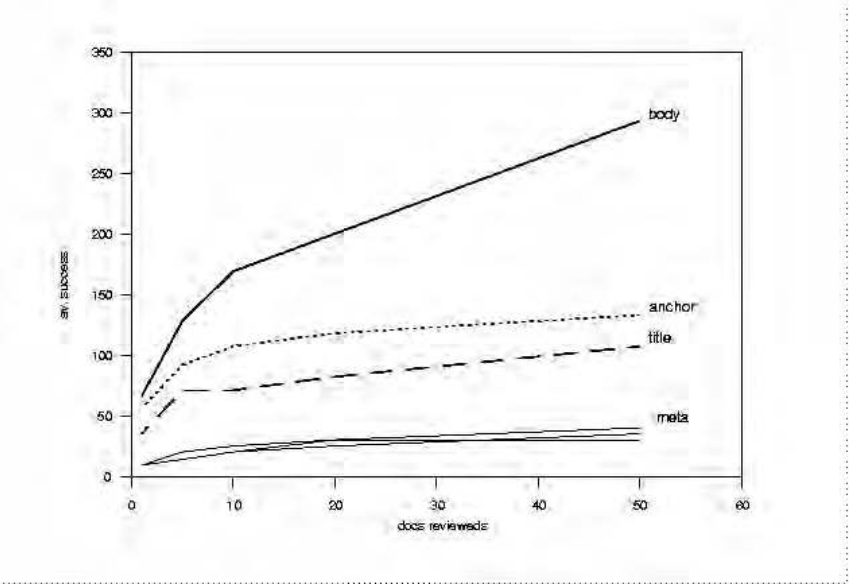

Fig. 2. Aportación de campos individuales

4 Conclusiones

Hemos descrito algunos de los experimentos realizados sobre la recuperación de páginas web a partir de determinados elementos. El uso de tales campos o elementos de información, además del texto dela zona `BODY` de las páginas web permite mejorar los resultados de la recuperación de información. Una forma de hacerlo es elaborar un único índice con los términos que aparecen en esos campos, junto con los que aparecen el el `BODY`; pero pesándolos de manera diferente, ajustable de forma empírica.

De todos esos campos, parece que el más eficaz es el de los `ANCHORS` de los *backlinks* que recibe cada página. El campo `TITLE` también contribuye de manera importante a la mejora

de la recuperación. El contenido de las etiquetas `META`, sin embargo, parece de utilidad reducida, desde el punto de vista de la recuperación.

Referencias

1. Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David Grossman, Ophir Frieder, and Nazli Goharian. On fusion of effective retrieval strategies in the same information retrieval system. *Journal of the American Society for Information Science and Technology (JASIST)*, 55(10):859–868, 2004.
2. William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval. April 11-13, 1994, Las Vegas, Nevada*, pages 161–175, 1994.
3. Carlos G. Figuerola, José L. Alonso Berrocal, Ángel F. Zazo Rodríguez, and Emilio Rodríguez. REINA at the WebCLEF task: Combining evidences and link analysis. In Peters [8].
4. Carlos G. Figuerola, José Luis A. Alonso Berrocal, Ángel F. Zazo Rodríguez, and Emilio Rodríguez Vázquez de Aldana. Herramientas para la investigación en recuperación de información: Karpanta, un motor de búsqueda experimental. *Scire*, 10(2):51–62, 2004.
5. Carlos G. Figuerola, Ángel F. Zazo, Emilio Rodríguez Vázquez de Aldana, and José Luis Alonso Berrocal. La recuperación de información en español y la normalización de términos. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 8(22):135–145, 2004.
6. Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In *The Second Text REtrieval Conference (TREC-2)*. NIST Special Publication 500-215, 1993.
7. Gertjan van Noord. Texcat language guesser.
8. Carol Peters, editor. *Results of the CLEF 2005 Cross-Language System Evaluation Campaign. Working notes for the CLEF 2005 Workshop, 21-23 September, Vienna, Austria*, 2005.
9. Gerard Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communication of the ACM*, 18:613–620, 1975.
10. B. Sigurbjornsson, J. Kamps, and M. De Rijke. EuroGOV: Engineering a multilingual Web corpus. *Lecture Notes in Computer Science*, 4022:825, 2006.
11. Börkur Sigurbjörnsson, Jaap Kamps, and Maarten de Rijke. Overview of webclef 2005. In Peters [8].
12. Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 18–22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*, pages 21–29. ACM, 1996.

Intelligent Web Navigation *

Inma Hernández

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
Avda. Reina Mercedes s/n
41012 Sevilla Spain
{inmahernandez}@us.es

Abstract. Virtual integration systems retrieve information from several web applications according to a user's interests. Being an online process, response time is a significant factor. Usually web pages contain a high number of links, some of them leading to interesting information, but most of them having other purposes, like advertising or internal site navigation. Traditional crawlers follow every link in each page, retrieving, analysing and classifying thousands of pages for every single site, which is a costly task. This problem can be solved with the combination of a web page classifier, to distinguish between interesting and irrelevant pages, and a link classifier, which automatically identifies links leading to interesting pages. This kind of navigation is more efficient and has a lower cost than traditional crawlers.

Key words: Enterprise Information Integration, Navigation, Information Retrieval, Virtual Integration

1 Motivation

There are two approaches to automated access to deep web information, namely Virtual Integration (also known as Metasearch) and Surfacing (also known as Crawling) [24]. In virtual integration, users define their interests using high-level structured queries, and the system retrieves information related to this query in response. This information is retrieved from many different web application sources, but it is presented uniformly to the users in a transparent way. This process is online, and therefore response time is an important issue. As opposed to virtual integration, surfacing is an off-line process that collects all pages behind a web form by submitting pre-computed queries, and not taking the user's requirements into account. In this case, minimising response time is not a priority.

Both previous approaches have a common phase: once the form has been submitted, a response page is retrieved and processed in order to check if it

* Partially funded by the Spanish National R&D&I Plan under grants TIN2008-04718, and the Andalusian Local Government under grants P07-TIC-02602, and P08-TIC-4100.

is a relevant page, and to use the information contained in it to reach further relevant pages. Traditional exhaustive crawlers use a blind approach to navigation, following every link in every page. This approach has a drawback: usually, web pages contain a large number of links, most of them non-relevant to the user (e.g., advertising or links to partner web sites). Too much time is wasted following all those links, making it less efficient.

As opposed to blind navigation, other approaches include some criteria to decide which links must be visited and which ones not, therefore reducing the number of irrelevant visited links. Focused crawlers, for instance, hold the criterion of avoiding pages not related to a certain topic (and therefore links leading to them). Relevancy criteria can be handcrafted by the user or automatically decided by the navigator, with the support of some reasoning process, usually in the form of a classifier. The latter is what we consider an automated intelligent navigator.

Figure 1 shows a virtual integration process with intelligent navigation, starting with a set of user queries which leads to a collection of response pages. Each one of these pages can be a Web page containing the answer to the query made by the user (detail page), or a paginated list of links to detail pages (hub page). In case the server lacks the answer to the query, the response page usually shows an informative message, and optionally some suggestions (no-results page). Finally, when some unexpected error arises, the response page may just contain some error description (error pages). Hence, a classifier is employed to distinguish between the different kinds of response pages, discarding both no-results and error pages and retrieving detail pages.

If a hub page is detected, it is further analysed in order to discriminate between the links it contains. A usual hub page has different types of links, including advertising, internal site navigation, links leading to detail pages, and links to the previous/following hub page(e.g., “More” or “Next” links). A link classifier, makes this distinction in order to follow only relevant links. Finally, when detail pages have been identified and retrieved, an information extractor is used to extract relevant structured data. This final step is beyond our scope.

The rest of the article is structured as follows. Section 2 describes related work in the navigation and web page classification areas; Section 3 shows the experimental study; Section 4 lists some of the conclusions extracted from the research and concludes the article.

2 Related Work

Next, we briefly describe and analyse the existing proposals in the Web Page Classification and Navigation areas.

2.1 Web Page Classification

Web page classification has been extensively researched, and several techniques have been applied with successful experimental results. In general, we classify

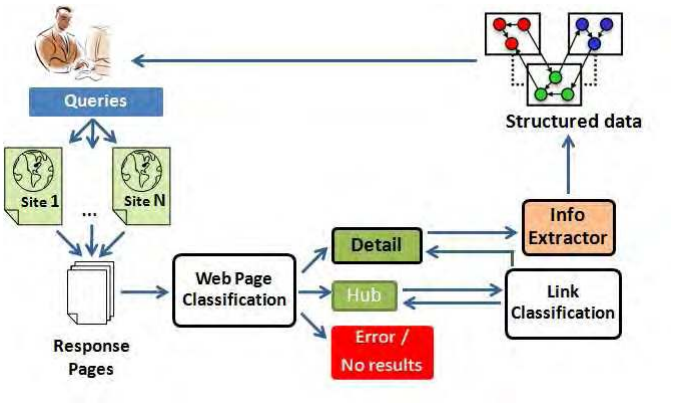

Fig. 1: Virtual Integration example.

them according to the type and location of the classification features. There are three main trends in feature types: content-based ([21], [32] and [34]), structure-based ([3], [4], [9], [15], [20], [18], [36] and [37]) and hybrid classifiers ([11] and [25]). There are also some abstract classification techniques that sort out pages on the basis of any given feature, e.g., PEBL [39]. As for feature location, most approaches obtain features from the page to be classified, whilst others get them from neighbouring pages. These proposals are content-based, and usually rely on features such as the link anchor text, the paragraph text surrounding the anchor [14], the headers preceding the anchor, or a combination of these [19].

2.2 Navigation

Traditional crawlers ([33]) navigate pages by following every link they find. This is useful for certain tasks, like indexing pages for a searching engine, but for virtual integration purposes it is not an efficient approach.

Recorders are also blind navigators, although they are more specific than crawlers, and less flexible. Anupam et. al. [2] present WebVCR, a recording tool with a VCR-like interface to record a user's navigation steps through a web site and store them in a file, in order to replay them automatically in future. Baumgartner et. al. [7] propose another recording system, based on Lixto, which includes an embedded browser in which the user can perform navigation steps. Pan et. al. [28] introduced the WARGO system, a wrapper creation tool, and a language to express navigation sequences called NSEQL. The system offers a browser-based interface to record the user navigation steps, which are transformed in sequences expressed in NSEQL. It is possible to leave some steps undefined until runtime, e.g., the number of clicks the system has to perform on a Next link depending on how many result pages the search returned. To solve this problem, NSEQL is endowed with some extraction capabilities.

The previous proposals interact directly with the web browser interface, so they do not deal with issues like scripts, posting forms, required authentication,

or sites that keep session information. They depend completely upon the user's knowledge, who is responsible for defining the navigation sequences, providing the values to fill in the forms and redefining the sequences whenever the target web site changes. Recording navigation steps makes the navigation inflexible, and error-prone.

Focused Crawling ([1], [17], [5], [6], [12], [13], [27], [29], [30] and [31]) is an improvement over traditional crawlers, in the sense that it combines a traditional crawler with a topical web page classifier, keeping the focus on pages with topics that have some interest for the user, and therefore reducing the number of useless retrieved pages.

However, focused crawlers improvement is limited to the reduction of useless pages retrieved, but the number of useless visited links is still an issue. Even when a high number of pages are discarded, they have to be visited and analysed previously. Moreover, focused crawlers do not classify the functionality of a page, but its topic. However, in a virtual integration system, every page should be processed differently according to the role it plays, like we mentioned before: hubs should be iterated, error pages should be discarded, and detail pages should be processed by an information extractor.

User-Defined Navigation is learned by the system in a supervised way, i.e., the user demonstrates how to obtain the interesting information, and the system is able to generalise this knowledge. EzBuilder [10] is a virtual integration tool used to create information extraction procedures. In order to learn navigation for a web site, the user needs to give an example, submitting forms and navigating to several of the desired pages, and specifying the structure of the data to be extracted. Then, the system builds a model of the user behaviour, that is employed to navigate automatically. This proposal relies on the user, from whom the system learns a navigation model for each site. This model cannot be reused in other sites automatically and the authors do not report any technique to update it.

Davulcu et. al. [16] present a technique to express navigation sequences using navigation maps. These are labelled directed graphs in which nodes represent web pages and edges represent actions that can be executed to navigate from one page to another (submitting a form or following a link). User navigation steps through the site, and form fields information is captured, and added to the navigation map. Some structural changes in web sites can be handled by the system automatically, whilst other changes makes it necessary to update the maps manually.

Other proposals take a workflow-based approach to represent navigation sequences, although the definition of these sequences is performed by the user. Montoto et. al. [26] model navigation sequences as activities in a work flow diagram. They detect some structural page patterns which frequently occur in web sites. However, they do not use classifiers to automatically distinguish between them, and furthermore the sequence of navigation steps to reach these pages is defined by the user; actually their technique is an extension of [28].

[38] is an example of a semi-automated navigator based on a content classifier. Navigation is defined as a sequence of pages, in which the last one is the goal, and the rest are intermediate steps. Each page belongs to a class, which the user defines using a set of keywords, and a set of actions. The system analyses the words in every page looking for class keywords. When a page belongs to an intermediate class, the actions mark the steps to get to the next page in the sequence. However, if the page belongs to a goal class, the actions are extraction rules, in order to extract desired information.

Automated navigators are similar to the former proposals, except for the fact that navigation patterns are automatically extracted from web sites, instead of learnt from the user. Liddle et. al. [23] propose a crawling tool, in which forms are submitted, but no specific information is filled in, in order to retrieve as many results as possible. Once the form is submitted, the system is able to automatically distinguish between an indexed list of results, a complete list of results, an informative page with a no-result message, and an error page. When a hub page is detected due to the presence of a Next link, this link is followed iteratively until the last page (or until a sufficiently large number of pages is found), and all retrieved pages are concatenated in a single result page.

This proposal is very simple, since it is only applicable if we wish to retrieve the summarised information contained in hub pages, but not the extended information in detail pages. Furthermore, this tool retrieves all pages, regardless of their relevance to the user's keywords.

For Palmieri et. al., [22], navigation patterns are a graph-based description of the different stages of a user navigation in a web site. They state that most web sites are designed according to the same navigation patterns, e.g., a start page with a link to an advanced search form, which is submitted to obtain a response page. They consider both HTML submission methods, POST and GET, which are expressed as different patterns. Once the form has been submitted, this approach analyses the response page to check if data objects of interest are present, in which case it is considered a hub page. Every hub page has a link leading to the next hub page, until all results are exposed to the user. Some heuristics are presented in order to detect and deal with these links.

However, this approach does not allow for the fact that that hub pages may only contain a brief summary of the data object and a link to a detail page. This proposal limits the possible navigation patterns to a couple of handmade samples, and therefore is not applicable to all web sites.

Vidal et. al. [36], consider that navigation patterns are automatically detected sequences of regular expression URLs that lead to relevant pages. This proposal receives a sample page, which represents the class of relevant pages, and it returns a navigation pattern, composed of the sequence of regular expressions that describe URLs that lead to the largest number of relevant pages. Just like the former approach, forms are submitted automatically, and the response page is analysed. If the page belongs to the same class as the one given as example (using a structural classifier), it is considered a relevant page, and it is then retrieved. Small pages are considered to be error pages and they are

not processed. Whenever a form is detected, it is submitted again, and process continues recursively. If a large number of links are detected, it is considered a hub page, and a further analysis is needed to select which links will be followed. Links are grouped by their URL similarity, and only one link from each group is followed to check if it leads to a relevant page.

This proposal is semi-supervised, since the process is automated, but the user has to provide a sample detail page, aside from filling in the parameter values for every form submission. This navigator can only reach a small number of possible results, for two reasons: it only keeps navigation patterns leading to the largest number of relevant pages, and besides, links leading to other hub pages are not considered, only links to detail pages, so only relevant pages referenced in the first hub page are retrieved. This navigator has to be combined with an iterator in order to retrieve all possible results. Another question to bear in mind is that navigation systems are ad-hoc. If the site changes its URL nomenclature, the system has to be trained again, thus requiring user intervention, to update navigation patterns.

This proposal has some technical problems to be solved, namely, the heuristic for the treatment of form pages can lead to an infinite loop, in cases in which the server includes the original form in every response page. It does not support form submission using the POST method. Finally, it does not support web sites that keep user session information.

[8] is another automated navigator. It is similar to the former: it relies on a structural-based clustering of pages, and it requires a sample relevant page. All pages are grouped in clusters, according to their structure, and links between pages are analysed in order to find relationships between clusters. Whenever a cluster has a high number of outgoing links to the cluster containing the sample, it is considered a hub cluster. This proposal is not designed to crawl pages behind forms, and therefore should be adapted in order to retrieve pages from the deep web.

3 Experimental Study

In order to justify the need for an intelligent navigator, we analysed a sample set of real web pages, and calculated the percentage of relevant links they included. Sample pages were extracted from Alexa Web Directory[1]. Alexa is supported by a traditional crawler, which collects publicly available web pages, rank them and offer additional information about them, like traffic statistics, demographic distribution of page viewers, keywords used to find that page in search engines, and related links. The first ten sites from each category were chosen, excluding Adults and Reference categories. Sites without hubs were discarded too, as well as sites that required user session information, or that enclosed relevant information in iframes (due to some conflicts with the Selenium XPath API), as they are not useful for our purposes. For every web site, a query was issued in order to

[1] `http://www.alexa.com/topsites`

obtain a response hub page. XPath expressions for relevant links in these pages were extracted by hand, including those that lead from one paginated hub to the next one. We developed a Java application using Selenium Core Reference Library[2], which automatically visited every page, calculated the number of relevant links defined by XPath expressions, as well as the total number of links in the page, and stored the page in a local repository. In order to analyse the

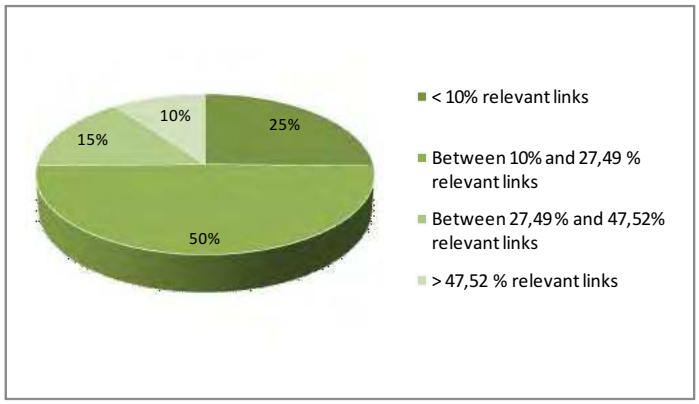

Fig. 2: Percentage Distribution

results, we calculated the percentage of relevant links in every page, and the average percentage of relevant links per category.

Figure 2 shows the percentage distribution we obtained with a sample set of 150 sites. Barely a 10% of the pages contained a percentage of relevant links higher than 47,52%. In fact, the quartile distribution was: (10,01%; 15,19%; 27,49%; 95,67%). The average percentage value of relevant links for all analysed pages was 22,35% with a standard deviation equal to 19,85. Figure 3 left shows the percentage of relevant links for every page in a random category (Arts). Figure 3 right shows the average percentage values by category. Values for all categories were similar, around 20%, except for Health category, which had the highest percentage value, with a 43%. These results confirm that average pages contain a high percentage of useless links, which should be avoided by the crawler in order to improve its efficiency.

4 Conclusions and Research Questions

The main problem posed in this paper is the development of an intelligent navigation system for virtual integration within the *Deep Web*, as opposed to the blind navigation performed by most of the previous works. In this context, users specify their interests by means of queries, and information is retrieved from several

[2] `http://seleniumhq.org/`

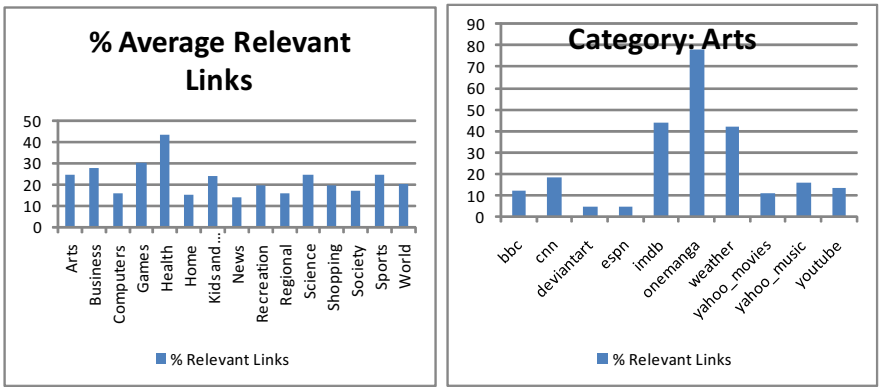

Fig. 3: Experimental Results

web applications. Response pages are analysed, navigating through their links and collecting only those pages that contain relevant information. In contrast, traditional crawlers follow every link and collect every possible page, wasting a lot of time and resources.

In order to support our thesis, we presented an experimental study, in which we estimated the average percentage of relevant links included in every page. We came to the conclusion that most pages in our sample set have a low percentage of relevant links. As we mentioned before, virtual integration is an online process, hence information should be presented to the user in a reasonable response time. Therefore, these systems will indeed find benefits in the application of an intelligent navigator, which avoids visiting useless pages by following only relevant links, reducing the cost and improving the efficiency of traditional crawling systems.

Summarising, these are some other research challenges concerning intelligent navigation:

1. Response pages have to be classified into the different roles that they play, in order to automatically deal with them.
2. Links in hub pages have to be classified before clicking on them, to identify those that lead to relevant pages.
3. Lazy link and web page classifiers work better in this context, i.e., classification model is built and updated progressively during the process.
4. Navigator should interact with the user as little as possible. Therefore, learning is unsupervised, or at least, very little supervised. Also, it should be as general as possible, not having to build an ad-hoc model for every site, but instead developing a general model that can adapt to most sites just by tuning some parameters.
5. Advanced post-filtering of hub pages is also a desirable feature. For example, when looking for products in an online store, a user may wish to retrieve only those products whose price lay within a certain range. This means that the navigator needs to be endowed with a lite extraction tool.

6. It is necessary to create a standard data set to evaluate classification and navigation proposals. Most proposals so not use a well-known data set in their experiments; instead, they collect their own set of pages by issuing queries to a search engine, or using a blind crawler. As the experiments are performed on different sets of pages, the experimental results cannot be compared. There are some well known data sets of already classified pages available for this kind of experiments, e.g., the WebKB collection[3], the Reuters-21578[4] news collection or the TEL-8 query interfaces collection[5]. [35] reports an attempt to collect a standard data set. Unfortunately, these collections are outdated and updating them is a costly task. We argue that more effort on archiving needs to be done so that new proposals can be compared from an empirical point of view.

References

1. C. C. Aggarwal et al. On the design of a learning crawler for topical resource discovery. *ACM Trans. Inf. Syst.*, 19(3), 2001.
2. V. Anupam et al. Automating web navigation with the webvcr. *Computer Networks*, 33(1-6), 2000.
3. A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD Conference*, 2003.
4. Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *WWW*, 2002.
5. L. Barbosa and J. Freire. Searching for hidden-web databases. In *WebDB*, 2005.
6. S. Batsakis et al. Improving the performance of focused web crawlers. *Data & Knowledge Engineering*, In Press, Corrected Proof, 2009.
7. R. Baumgartner et al. Deep web navigation in web data extraction. In *CIMCA/IAWTIC*, 2005.
8. L. Blanco et al. Efficiently locating collections of web pages to wrap. In *WEBIST*, 2005.
9. L. Blanco et al. Structure and semantics of data-intensiveweb pages: An experimental study on their relationships. *J. UCS*, 14(11), 2008.
10. J. Blythe et al. Information integration for the masses. *J. UCS*, 14(11), 2008.
11. J. Caverlee and L. Liu. Qa-pagelet: Data preparation techniques for large-scale data analysis of the deep web. *IEEE Trans. Knowl. Data Eng.*, 17(9), 2005.
12. S. Chakrabarti et al. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks*, 30(1-7), 1998.
13. S. Chakrabarti et al. Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16), 1999.
14. W. W. Cohen et al. Improving a page classifier with anchor extraction and link analysis. In *NIPS*, 2002.
15. V. Crescenzi et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, 2001.
16. H. Davulcu et al. A layered architecture for querying dynamic web content. In *SIGMOD Conference*, 1999.

[3] `http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/`
[4] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`
[5] `http://metaquerier.cs.uiuc.edu/repository/datasets/tel-8/`

17. G. T. de Assis et al. Exploiting genre in focused crawling. In *SPIRE*, 2007.
18. D. de Castro Reis et al. Automatic web news extraction using tree edit distance. In *WWW*, 2004.
19. J. Fürnkranz et al. Hyperlink ensembles: a case study in hypertext classification. *Information Fusion*, 3(4), 2002.
20. S. Grumbach and G. Mecca. In search of the lost schema. In *ICDT*, 1999.
21. A. Hotho et al. Ontology-based text document clustering. *KI*, 16(4), 2002.
22. J. P. Lage et al. Automatic generation of agents for collecting hidden web pages for data extraction. *Data Knowl. Eng.*, 49(2), 2004.
23. S. W. Liddle et al. Extracting data behind web forms. In *ER (Workshops)*, 2002.
24. J. Madhavan et al. Harnessing the deep web: Present and future. In *CIDR*, 2009.
25. A. Markov et al. The hybrid representation model for web document classification. *Int. J. Intell. Syst.*, 23(6), 2008.
26. P. Montoto et al. A workflow language for web automation. *J. UCS*, 14(11), 2008.
27. S. Mukherjea et al. Discovering and analyzing world wide web collections. *Knowl. Inf. Syst.*, 6(2), 2004.
28. A. Pan et al. Semi-automatic wrapper generation for commercial web sources. In *Engineering Information Systems in the Internet Context*, 2002.
29. G. Pant and P. Srinivasan. Learning to crawl: Comparing classification schemes. *ACM Trans. Inf. Syst.*, 23(4), 2005.
30. G. Pant and P. Srinivasan. Link contexts in classifier-guided topical crawlers. *IEEE Trans. Knowl. Data Eng.*, 18(1), 2006.
31. I. Partalas et al. Reinforcement learning with classifier selection for focused crawling. In *ECAI*, 2008.
32. J. M. Pierre et al. On the automated classification of web sites. *CoRR*, cs.IR/0102002, 2001.
33. S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *VLDB*, 2001.
34. A. Selamat and S. Omatu. Web page feature selection and classification using neural networks. *Inf. Sci.*, 158, 2004.
35. M. Sinka and D. Corne. A large benchmark dataset for web document clustering. In *Soft Computing Systems: Design, Management and Applications, Volume 87 of Frontiers in Artificial Intelligence and Applications*. 2002.
36. M. L. A. Vidal et al. Structure-based crawling in the hidden web. *J. UCS*, 14(11), 2008.
37. K. Vieira et al. A fast and robust method for web page template detection and removal. In *CIKM*, 2006.
38. Y. Wang and T. Hornung. Deep web navigation by example. In *BIS (Workshops)*, 2008.
39. H. Yu et al. Pebl: Web page classification without negative examples. *IEEE Trans. Knowl. Data Eng.*, 16(1), 2004.

Web Navigation Automation in AJAX Websites

Paula Montoto[1], Alberto Pan[1], Juan Raposo[1], Fernando Bellas[1] and Javier López[1],

[1] Department of Information and Communication Technologies, University of A Coruña
Facultad de Informática, Campus de Elviña s/n 15071 A Coruña, Spain
{pmontoto,apan,jrs,fbellas,jmato}@udc.es

Abstract. Web automation applications are widely used for different purposes such as B2B integration or automated testing of web applications. One crucial part in web automation applications is to allow easily generating and reproducing navigation sequences. Previous proposals in the literature assumed a navigation model today turned obsolete by the new breed of AJAX-based websites. Although some open-source and commercial tools have also addressed the problem, they show significant limitations either in usability or their ability to deal with complex websites. In this paper, we propose a set of new techniques to deal with this problem.

Keywords: Web automation, web integration, web wrappers.

1 Introduction

Web automation applications are widely used for different purposes such as B2B integration, web mashups, automated testing of web applications or business watch. One crucial part in web automation applications is to allow easily generating and reproducing navigation sequences. We can distinguish two stages in this process:

- Generation phase. In this stage, the user specifies the navigation sequence to reproduce. The most common approach, cf. [1,9], is using the 'recorder' metaphor: the user performs one example of the navigation sequence using a modified web browser, and the tool generates a specification which can be run by the execution component.
- Execution phase. In this stage, the sequence generated in the previous stage and the input parameters are provided as input to an automatic navigation component which is able to reproduce the sequence. The automatic navigation component can be developed by using the APIs of popular browsers (e.g. [11]). Other systems [1] use simplified custom browsers.

Most existing proposals assume a navigation model which is now obsolete: on one hand, the user actions that could be recorded were mainly restricted to clicking on elements and filling in form fields; on the other hand, it was assumed that almost every user action caused a request to the server for a new page.

Nevertheless, this is not enough for dealing with modern AJAX-based websites. These sites can respond to a much wider set of user actions (mouse over, keyboard strokes, drag and drop…) and they respond to those actions executing scripting code that manipulates the page at will (for instance, by creating new elements on the fly).

In this paper, we propose a set of new techniques to build an automatic web navigation system able to deal with all this complexity. In the generation phase, we also use the 'recorder' metaphor, but substantially modified to support recording a wider range of events; we also present new methods for identifying the elements participating in a navigation sequence in a change-resilient manner.

In the execution phase, we use the APIs of commercial web browsers to implement the automatic web navigation components (the techniques proposed for the recording phase have been implemented using Microsoft Internet Explorer (MSIE) and the execution phase has been implemented using both MSIE and Firefox). We take this option because the approach of creating a custom browser supporting technologies such as scripting code and AJAX requests is very vulnerable to small implementation differences that can make a web page to behave differently. In the execution phase, we also introduce a method to detect when the effects caused by a user action have finished. This is needed because one navigation step may require the effects of the previous ones to be completed before being executed.

2 Models

In this section we describe the models we use to characterize the components used for automated browsing. The main model we rely on is DOM Level 3 Events Model [2]. This model describes how browsers respond to user-performed actions on an HTML page currently loaded in the browser. Although the degree of implementation of this standard by real browsers is variable, the key assumptions our techniques rely on are verified in the most popular browsers (MSIE and Firefox).

2.1 DOM Level 3 Events Model

In the DOM Level 3 Events Model, a page is modelled as a tree. Each node in the tree can receive events produced (directly or indirectly) by the user actions. Event types exist for actions such as clicking on an element (*click*), moving the mouse cursor over it (*mouseover*) or specifying the value of a form field (*change*), to name a few. Each node can register a set of event listeners for each event type. A listener executes arbitrary code (typically written in a script language such as Javascript). Listeners have the entire page tree accessible and can perform actions such as modifying existing nodes, removing them, creating new ones or even launching new events.

The event processing lifecycle can be summarized as follows: The event is dispatched following a path from the root of the tree to the target node. It can be handled locally at the target node or at any target's ancestors in the tree. The event dispatching (also called event propagation) occurs in three phases and in the following order: capture (the event is dispatched to the target's ancestors from the root of the tree to the direct parent of the target node), target (the event is dispatched to the target node) and bubbling (the event is dispatched to the target's ancestors from the direct parent of the target node to the root of the tree). The listeners in a node can register to either the capture or the bubbling phase. In the target phase, the events registered for the capture phase are executed before the events executed for the

bubbling phase. This lifecycle is a compromise between the approaches historically used in major browsers (Microsoft IE using bubbling and Netscape using capture).

The order of execution between the listeners associated to an event type in the same node is registration order. The event model is re-entrant, meaning that the execution of a listener can generate new events. Those new events will be processed synchronously; that is, if l_i, l_{i+1} are two listeners registered to a certain event type in a given node in consecutive order, then all events caused by l_i execution will be processed (and, therefore, their associated listeners executed) before l_{i+1} is executed.

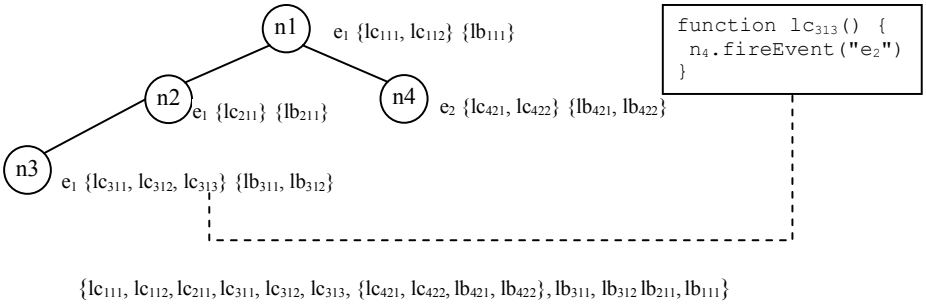

Fig. 1. Listeners Execution Example

Example 1: Fig. 1 shows an excerpt of a DOM tree and the listeners registered to the event types e_1 and e_2. The listeners in each node for each event type are listed in registration order (the listeners registered for the capture phase appear as l_{cxyz} and the ones registered for the bubbling phase appear as l_{bxyz}). The figure also shows what listeners and in which order would be executed in the case of receiving the event-type e_1 over the node n_3, assuming that the listener on the capture phase l_{c313} causes the event-type e_2 to be executed over the node n_4.

DOM Level 3 Events Model provides an API for programmatically registering new listeners and generating new events. Nevertheless, it does not provide an introspection API to obtain the listeners registered for an event type in a certain node. As we will see in section 3.1, this will have implications in the recording process in our system.

2.2 Asynchronous Functions and Scripts Execution Model

In this section we describe the model we use to represent how the browser executes the scripting code of the listeners associated to an event. This model is verified by the major commercial browsers.

The script engine used by the browser executes scripts sequentially in single-thread mode. The scripts are added to an execution queue in invocation order; the script engine works by sequentially executing the scripts in the order specified by the queue.

When an event is triggered, the browser obtains the listeners that will be triggered by the event and invokes its associated scripts, causing them to be added to the execution queue. Once all the scripts have been added, execution begins and the listeners are executed sequentially.

The complexity of this model is slightly increased because the code of a listener can execute asynchronous functions. An asynchronous function executes an action in a non-blocking form. The action will run on the background and a callback function provided as parameter in the asynchronous function invocation will be called when the action finishes.

The most popular type of asynchronous call is the so-called AJAX requests. An AJAX request is implemented by a script function (i.e. in Javascript, a commonly used one is *XMLHTTPRequest*) that launches an HTTP request in the background. When the server response is received, the callback function is invoked to process it.

Other popular asynchronous calls establish timers and the callback function is invoked when the timer expires. In this group, we find the Javascript functions *setTimeout(ms)* and *setInterval(ms)*. Both have associated cancellation functions.

It is important to notice that, from the described execution model, it is inferred the following property:

Property 1: The callback functions of the asynchronous calls launched by the listeners of an event will never be executed until all other scripts associated to that event have finished.

The explanation for this property is direct from the above points: all the listeners associated to an event are added to the execution queue first, and those listeners are the ones invoking the asynchronous functions; therefore, the callback functions will always be positioned after them in the execution queue even if the background action executed by the asynchronous call is instantaneous.

3 Description of the Solution

3.1 Recording User Events

The generation phase has the goal of recording a sequence of actions performed by the user to allow reproducing them later during the execution phase.

A user action (e.g. a click on a button) causes a list of events to be issued to the target node, triggering the invocation of the listeners registered for them in the node and its ancestors, according to the execution model described in the previous section. Notice that each user action usually generates several events. For instance, the events generated when the user clicks on a button include, among others, the mouseover event besides of the click event, since in order to click on an element it is previously required to place the mouse over it. Recording a user action consists in detecting which events are issued, and in locating the target node of those events.

In previous proposals, cf. [1,9], the user can record a navigation sequence by performing it in the browser in the same way as any other navigation. The method used to detect the user actions in these systems is typically as follows: the recording tool registers its own listeners for the most common events involved in navigations (mainly clicks and the events involved in filling in form fields) in anchors and form-related tags. This way, when a user action produces one of the monitored event-types e on one of the monitored nodes n, the listener for e in n is invoked, implicitly identifying the information to be recorded.

Nevertheless, the modern AJAX-based websites can respond to a much wider set of user actions (e.g. placing the mouse over an element, producing keyboard strokes, drag and drop…); in addition, virtually any HTML element, and not only traditional navigation-related elements, can respond to user actions: tables, images, texts, etc.

Extending the mentioned recording process to support AJAX-based sites would involve registering listeners for every event in every node of the DOM tree (or, alternatively, registering listeners for every event in the root node of the page, since the events execution model ensures that all events reach to the root). Registering listeners for every event has the important drawback that it would "flood" the system by recording unnecessary events (e.g. simply moving the mouse over the page would generate hundreds of *mouseover* and *mouseout* events); recall that, as mentioned in section 2, it is not possible to introspect what events a node has registered a listener for; therefore, it is not possible to use the approach of registering a listener for an event-type *e* only in the nodes that already have other listeners for *e*.

Therefore, we need a new method for recording user actions. Our proposal is letting the user explicitly specify each action by placing the mouse over the target element, clicking on the right mouse button, and choosing the desired action in the contextual menu (see Fig. 2). If the desired action involves providing input data into an input element or a selection list, then a pop-up window opens allowing the user to specify the desired value (see Fig. 2). Although in this method the navigation recording process is different from normal browsing, it is still fast and intuitive: the user simply changes the left mouse button for the right mouse button and fills in the value of certain form fields in a pop-up window instead of in the field itself.

This way, we do not need to add any new listener: we know the target element by capturing the coordinates where the mouse pointer is placed when the action is specified, and using browser APIs to know what node the coordinates correspond to. The events recorded are implicitly identified by the selected action.

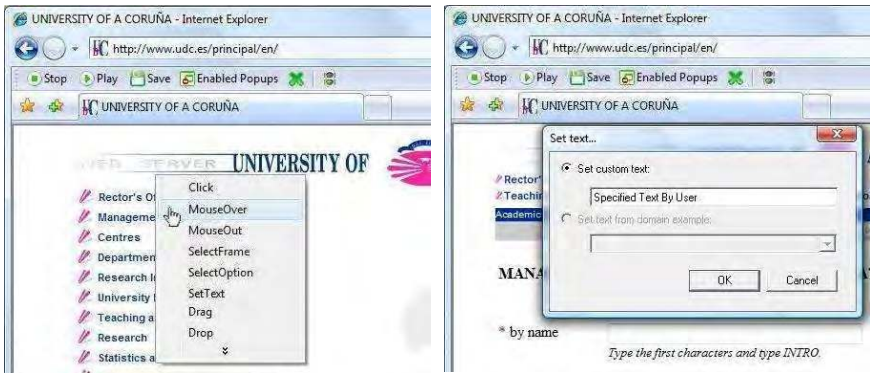

Fig. 2. Recording Method

Our prototype implementation includes actions such as *click*, *mouseover*, *mouseout*, *selectOption* (selecting values on a selection list), *setText* (providing input data into an element), *drag* and *drop*. Since each user action actually generates more than one event, each action has associated the list of events that it causes: for instance,

the *click* action includes, among others, the events *mouseover*, *click* and *mouseout*; the *setText* action includes events such as *keydown* and *keyup* (issued every time a key is pressed) and *change* (issued when an element content changes).

This new method has a problem we need to deal with. By the mere process of explicitly specifying an action, the user may produce changes in the page before we want them to take place. For instance, suppose the user wishes to specify an action on a node that has a listener registered for the *mouseover* event; the listener opens a pop-up menu when the mouse is placed over the element. Since the process of specifying the action involves placing the mouse over the element; the page will change its state (i.e. the pop-up menu will open) before the user can specify the desired action.

We solve this problem by deactivating the reception of user events in the page during the recording process. Once the user has specified an action, we use the browser APIs to generate on the target element the list of events associated to the action; this way, the effects of the specified action take place in the same way as if the user would have performed the action, and the recording process can continue.

Another issue we need to deal with is ensuring that a user does not specify a new action until the effects caused by the previous one have completely finished. Detecting the end of the effects of an action is far from trivial; since it is one of the crucial issues at the execution phase, we will describe how to do it in section 3.3.

3.2 Identifying Elements

During the generation phase, the system records a list of user actions, each one performed on a certain node of the DOM tree of the page. Therefore, we need to generate an expression to uniquely identify the node involved in each action, so the user action can be automatically reproduced at the execution phase.

An important consideration is that the generated expression should be resilient to small changes in the page (such as the apparition in the page of new advertisement banners, new data records in dynamically generated sections or new options in a menu), so it is still valid at the execution stage. In addition, the generated expressions should not be sensible to the use of session identifiers, to ensure that they will still work in any later session. The algorithms we use for this purposes are detailed in [7].

3.3 Execution Phase

The generation phase generates a program capturing the navigation sequence recorded by the user. The execution phase runs the program in the automatic navigation component.

A first consideration is that we opt to use the APIs of commercial web browsers to implement the automatic web navigation components instead of building a simplified custom-browser. The main reason for taking this option is that web 2.0 sites make an intensive use of scripting languages and support a complex event model. Creating a custom browser supporting those technologies in the same way as commercial browsers is extremely vulnerable to small implementation differences that can make a web page to behave differently when accessed with the custom browser than when

accessed with a "real" browser. Our techniques for the execution phase have been implemented in both MSIE and Firefox.

To reproduce an action in the navigation sequence, there are three steps involved:

1. Locating the target node in the DOM tree of the page.
2. Generating the recorded event (or list of events) on the identified node.
3. Wait for the effects of the events to finish. This is needed because the following action can need the effects of the previous ones to be completed (e.g. the action *n+1* can generate an event on a node created in the action *n*).

The implementation of 1) and 2) is quite straightforward using browser APIs and given the output of the recording process. Step 1) uses the XPath expression produced by the process we mentioned in section 3.2, and step 2) uses the events recorded in the process described in section 3.1.

In turn, step 3) is difficult because browser APIs do not provide any way of detecting when the effects on the page of issuing a particular event have finished. These effects can include dynamically creating or removing elements in the DOM tree, maybe also having into account the response to one or several AJAX requests to the server. Previous works have addressed this problem by establishing a timer after the execution of an event before continuing execution. This solution has the usual drawbacks associated to a fixed timeout in a network environment: if the specified timeout is short, then when the response to an asynchronous AJAX request is slower than usual (or even if the machine is very heavily loaded), the sequence may fail. If, in turn, we use a higher timeout valid even in those circumstances, then we are introducing an unnecessary delay when the server is responding normally.

Now we explain the method we propose to detect when the effects caused directly or indirectly by a certain event have finished. The correctness of the method derives from the assumptions stated in section 2, verified by the major commercial browsers.

The method we use to detect when the effects of an event-type *e* generated on a node *n* have finished consists of the following steps (see [7] for more detail):

1. We register a new listener *l* to capture the event *e* in *n*. The code of the listener *l* invokes an asynchronous function specifying the callback function *cf*. What asynchronous function is actually invoked in *l* is mainly irrelevant; for instance, in Javascript, we can simply invoke *setTimeout(cf,0)*. Notice that as consequence of property 1 in section 2, it is guaranteed that *cf* will be executed after all the listeners triggered by the execution of *e* have finished. Therefore, if the listeners had not made any other asynchronous call, then the control arriving to *cf* would indicate that the effects of *e* had finished and the navigation sequence execution could continue. Nevertheless, since the listeners can actually execute other asynchronous calls, this is not enough.
2. To be notified of every asynchronous call executed by the listeners triggered by *e*, we redefine those asynchronous functions providing our own implementation of them (for instance, in Javascript we need to redefine *setTimeout*, *setInterval* and the functions used to execute AJAX requests such as *XMLHTTPRequest*). The template of our implementation of each function is shown in [7]. The function maintains a counter that is increased every time the function is invoked (the counter is maintained as a global variable initialized to zero for every emitted event). After increasing the counter, the function calls the former standard implementation of the asynchronous function provided by the browser but

substituting the received callback function by a modified one. This new callback function invokes the original callback function and then decreases the counter. This way, the counter always takes the value of the number of currently active calls.

3. When the callback function *cf* from step 1 is executed, it polls the counters associated to the asynchronous functions. When they are all 0, we know the asynchronous calls have finished and execution can proceed.

4 Evaluation

To evaluate the validity of our approach, we tested the implementation of our techniques with a wide range of AJAX-based web applications. We performed two kinds of experiments:

1. We selected 75 real websites making extensive use of scripting code and AJAX technology. We used the prototype to record and reproduce one navigation sequence on each site. The navigation sequences automated the main purpose of the site. For instance, in electronic shops we automated the process of searching products; in webmail sites we automated the process required to access e-mails.

Table 1. Experimental Results

Website	Played	Website	Played	Website	Played
www.a9.com/java	✓	www.fidelityasap.com	✓	www.optize.es	✓
www.abebooks.com	✓	www.fnac.es	✓	www.paginasamarillas.es	✓
www.accorhotels.com	✓	www.gmail.com	✓	www.penguin.co.uk	✓
www.addall.com	✓	www.gongdiscos.com	✓	people.yahoo.com	✓
www.voyages-sncf.com	✓	www.hotelopia.es	✓	code.jalenack.com/periodic	✓
www.alitalia.com/ES_ES/	✓	www.hotelsearch.com	✓	www.pixmania.com	✓
www.allbooks4less.com	✓	www.iberia.com	✓	www.planethome.de	✓
www.amadeus.net	✓	www.iit.edu	✓	www.priceline.com	✓
www.amazon.com	✓	www.imdb.com/search	✓	www.renault.es	✓
store.apple.com	✓	www.infojobs.net	✓	www.renfe.es	✓
www.atrapalo.com	✓	www.jet4you.com	✓	www.reuters.com	✓
autos.aol.com	✓	www.laborman.es	✓	www.rumbo.es	✓
www.balumba.es	✓	www.landrover.com	✓	www.shop-com.co.uk	✓
www.barnesandnoble.com	✓	www.es.lastminute.com	✓	www.sparkassen-immo.de	✓
www.bookdepository.co.uk	✓	www.marsans.es	✓	www.sterling.dk	✓
www.booking.com	✓	www.meridiana.it	✓	www.ticketmaster.com	✓
www.carbroker.com.au	✓	www.msnbc.msn.com	✓	tudulist.com	✓
www.casadellibro.com	✓	www.muchoviaje.com	✓	www.tuifly.com/es	✓
www.cervantesvirtual.com	✓	www.musicstore.com	✓	es.venere.com	✓
www.cia.gov	✓	www.myair.com	✓	www.viajar.com	✓
controlp.com	✓	www.mymusic.com	✓	www.vuelosbaratos.es	✓
www.digitalcamerareview.com	✓	www.es.octopustravel.com	✓	www.webpagesthatsuck.com	✓
www.ebay.es	✓	www.ofertondelibros.com	✓	news.search.yahoo.com/news/advanced	✓
www.edreams.es	✓	www.okipi.com	✓	news.yahoo.com	×
www.elcorteingles.es	✓	vols.opodo.fr	✓	mail.yahoo.com	✓

2. Some of the main APIs for generating AJAX-based applications such as Yahoo! User Interface Library (YUI) [12] and Google Web Toolkit (GWT) [3] include a set of example websites. At the time of testing, GWT included 5 web applications and YUI included 300 examples. We recorded and executed 12 navigation sequences in the web applications from GWT ensuring that every interface element from the applications was used at least once. In the case of YUI, we recorded 40

sequences in selected examples (choosing the more complex examples). This second group of tests is useful because many real websites use those toolkits.

The results of the evaluation were encouraging (see Table 1). In the first set of experiments (real websites), 74 of 75 sequences were recorded and executed fine.

In the case of *news.yahoo.com*, the XPath expression generated by the algorithm mentioned in section 3.2 to identify an element was incorrect. This issue and how to solve it is discussed in detail in [7].

The second group of tests was completely successful in GWT applications, while in the YUI case only one sequence could not be recorded. The problem was that the *blur* event was not being generated with the *setText* action. Once this was corrected, the sequence could be recorded.

5 Related Work

WebVCR [1] and WebMacros [9] were pioneer systems for web navigation sequences automation using the "recorder metaphor". Both systems were only able to record a reduced set of events (clicks and filling in form fields) on a reduced set of elements (anchors and form-related elements). In the execution phase they relied on HTTP clients that lacked the ability to execute scripting code or AJAX requests

Selenium [11] is a suite of tools to automate web applications testing. Selenium uses the recorder metaphor through a toolbar installed in Firefox. Due to the inherent limitations to that approach mentioned in section 3.1, it is only able to record a reduced set of events. Another drawback is that Selenium does not detect properly the end of the effects caused by a user action in the recording process, relying in timers for this task, with the associated drawbacks we have already mentioned.

Sahi [10] is another open-source tool for automated testing of web applications. Sahi includes a navigation recording system and it allows the sequences to be executed in commercial browsers. To use Sahi, the user configures its navigator to use a proxy. Every time the browser requests a new page, the proxy retrieves it, adds listeners for monitoring user actions, and returns the modified page. Using a proxy makes the recording system independent of the web browser used. Nevertheless, using a proxy does not allow using approaches where the user explicitly indicates the actions to record; therefore, as discussed previously, it inherently forces to choose between either monitoring only a reduced set of events or suffering from "event flooding". Sahi only supports recording events such as click and change. Other events such as *mouseover* can be used at the execution phase if the user manually codes the navigation scripts. Another drawback is that they do not detect the end of the effects caused by a user action, using timeouts instead.

HtmlUnit [4] is another open-source tool for web applications unit testing. HtmlUnit does not provide a recording tool; instead, the user needs to manually create the navigation sequences using Java coding. In addition HtmlUnit uses its own custom browser instead of relying on conventional browsers. Although their browser has support for many Javascript and AJAX functionalities, it is vulnerable to small implementation differences that can make a web page to behave differently when accessed with the custom browser.

In the commercial software arena, QEngine [8] is a load and functional testing tool for web applications. QEngine also uses the recorder metaphor through a toolbar installed in MSIE (also used as execution component). In addition of the most typical events supported by the previously mentioned systems, QEngine also supports a form of explicitly specifying *mouseover* events on certain elements, consisting in placing the mouse over the target element for more than a certain timeout (avoiding this way the "flooding" problem). Nevertheless, they do not capture other events such as *mouseout* or *mousemove*. In addition, as previous systems, QEngine does not detect the end of the effects of an action. iOpus [5] is another web automation tool that uses the recorder metaphor. Their drawbacks with respect to our proposal are almost identical to those mentioned for QEngine.

In general, all the tools oriented to test automation also have one additional drawback: to identify elements in sequences, they use simple systems based on the text, attributes and relative position of the elements. While this is enough for application-testing purposes where the changes in the tested applications are controlled, it is not enough to deal with autonomous web sources

Kapow [6] is yet another web automation tool oriented to the creation of mashups and web integration applications. Kapow uses its own custom browser. Therefore, in our evaluation it showed to be vulnerable to the formerly mentioned drawback: small implementation differences can make a web page to behave differently. For instance, from the set of 12 sequences from Google Web Toolkit we used in our tests, the Kapow browser could only successfully reproduce 1 of them.

References

1. Anupam, V., Freire, J., Kumar, B., and Lieuwen, D. Automating web navigation with the WebVCR. In proceedings of WWW 2000, 503-517
2. Document Object Model (DOM) Level 3 Events Specification. http://www.w3.org/TR/DOM-Level-3-Events/
3. Google Web Toolkit. http://code.google.com/webtoolkit/
4. HtmlUnit. http://htmlunit.sourceforge.net/
5. iOpus. http://www.iopus.com
6. Kapow. http://www.openkapow.com
7. Montoto, P., Pan, A., Bellas, F., López, J. Automating Navigation Sequences in AJAX Websites. Proceedings of ICWE 2009, 166-180.
8. QEngine. http://www.adventnet.com/products/qengine/index.html
9. Safonov, A., Konstan, J., and Carlis, J. Beyond Hard-to-Reach Pages: Interactive, Parametric Web Macros. Proc. of the 7th Conference on Human Factors & the Web. 2001.
10. Sahi. http://sahi.co.in/w/
11. Selenium. http://seleniumhq.org/
12. Yahoo! User Interface Library (YUI). http://developer.yahoo.com/yui

SKOS en la integración de conocimiento en los sistemas de información jurídica

M. Mercedes Martínez-González[1], Beatriz Pérez-León[1], and M. Luisa Alvite-Díez[2]

[1] Departamento de Informática, Universidad de Valladolid
Edificio T.I.T., Campus 'Miguel Delibes' s/n, 47011 Valladolid
{mercedes,bperezle}@infor.uva.es
[2] Área de Biblioteconomía y Documentación, Universidad de León
Facultad de Filosofía, Campus de Vegazana, s/n, 24071 León
luisa.alvite@unileon.es

Resumen Los tesauros son herramientas conceptuales que representan un área de conocimiento. Los estándares RDF y SKOS, vinculados a la Web Semántica, permiten su representación con lenguajes y vocabularios estándares, lo cual facilita la integración de conceptos. A pesar de la importante ventaja que supone disponer de estos estándares, no es suficiente para garantizar totalmente la integración de las herramientas conceptuales utilizadas. Demostramos esta afirmación en el caso de los sistemas de información jurídica, en los cuales analizamos el uso del tesauro Eurovoc y su representación con SKOS: propuestas, existencia de herramientas estándares para tesauros, y dificultades para la integración. A partir de este análisis se plantean reflexiones sobre el estado de la cuestión en el momento actual.

Palabras clave: SKOS, tesauros, Eurovoc, sistemas de información jurídica, Web Semántica

1. Estándares para la integración de conocimiento

Una de las bases de la integración en los sistemas de información en el nivel de conocimiento ha sido y es la utilización de ontologías y otras herramientas conceptuales similares, como vocabularios controlados, tesauros o topic maps.

La incorporación de los sistemas de organización del conocimiento (*Knowledge Organization Systems, KOS*) como tesauros, vocabularios controlados o esquemas de clasificación a los nuevos entornos tecnológicos ha experimentado un gran empuje gracias a la Web Semántica. Las ontologías son las herramientas conceptuales más potentes de las que disponemos para expresar semántica al más alto nivel. Proliferan en todo tipo de sistemas de información como herramientas que soportan las búsquedas conceptuales y otra serie de funcionalidades. No obstante, no está tan claro que siempre cumplan la función de integración para la que podrían servir, ya que en muchos casos se diseñan ad-hoc para un propósito específico y nunca más son reutilizadas. Esto es, su función se limita al sistema local en cuyo contexto se crearon.

La utilización compartida de un conjunto de conceptos con semántica bien definida (conceptos que se representan mediante un conjunto de términos representantes) sería la situación ideal –reutilización de las mismas herramientas conceptuales–. Pero, como hemos dicho, no es la situación habitual, ya que en muchos casos cada sistema utiliza sus propios conceptos. La integración semántica es la encargada de paliar este problema.

La Web Semántica no sería tal si no se cumple el objetivo de que los datos sean legibles y comprensibles por los agentes software [2], para lo cual son necesarios estándares que permitan la representación de información en formatos interoperables. Con este fin surgen los distintos estándares de representación, comenzando con XML, y siguiendo con los que se apoyan en él. RDF permite representar metadatos. RDF Schema añade la posibilidad de representar el vocabulario utilizado en grafos RDF y relacionarlo mediante estructuras clasificatorias sencillas como taxonomías. OWL aporta el vocabulario estándar para representar ontologías. SKOS es un "modelo de datos diseñado para compartir sistemas de organización de conocimiento en la Web"[17]. SKOS se puede implementar sobre RDF, utilizarse aisladamente, o combinarlo con OWL [17].

En la versión básica de SKOS, los recursos conceptuales, esto es, los conceptos, se identifican mediante URI, cada concepto tiene al menos un término que lo representa, los conceptos se relacionan entre si, creando jerarquías de conceptos, y se agregan bajo estructuras denominadas esquemas de conceptos (*concept schemes*). En su versión avanzada se introduce la posibilidad de agrupar conceptos que están en distintos esquemas de conceptos en *colecciones* (*collections*), que pueden estar o no ordenadas.

También en el marco de la Web Semántica, se proporcionan los lenguajes de consulta estándar que deben permitir recuperar la información modelada utilizando estos estándares de representación. XQuery [4] permite consultar datos XML, y SPARQL [15] es un lenguaje de consulta para RDF que se estabilizó como recomendación del W3C a principios de 2008.

Parece claro en este contexto que la integración en el nivel de conocimiento debe venir de la mano de la utilización combinada de estos instrumentos (herramientas conceptuales y estándares de representación), e incluso que su uso debería ser suficiente para garantizarla. Sin embargo, esta última afirmación debería ser más bien una pregunta. Una pregunta cuya respuesta vamos a analizar en un dominio de aplicación con el que venimos trabajando desde hace tiempo [9]: los sistemas de información jurídica.

2. Representación de conocimiento en los sistemas de información jurídica mediante tesauros y XML

En la manipulación de información jurídica el uso de XML como el estándar más adecuado para representar la información se ha generalizado de tal modo que es imposible hablar de sistemas de información jurídica sin pensar en la utilización de XML [1,3]. Nuestro trabajo viene siguiendo esta filosofía desde hace tiempo [10]. Diseñamos esquemas de representación de los textos jurídicos

y sus metadatos y utilizamos los estándares asociados a XML para representarlos y acceder a ellos desde diversas aplicaciones [9].

Una interesante utilidad para los usuarios de estos sistemas es la posibilidad de anotar los textos, o los elementos de información representativos (elementos de estructura en nuestro caso), con comentarios sobre su campo de aplicación, tema o temas con los que está relacionado y otras observaciones. En el caso de herramientas docentes, como algunas de nuestras aplicaciones [13], esto puede servir para que el profesor indique a los alumnos con qué tema o temas de la materia debería relacionar el documento que está analizando. También está, por supuesto, la posibilidad más tradicional de clasificar los documentos en función de materia o materias, de modo que después se permitan búsquedas de documentos relacionados con una determinada materia. Esta extensión a través de anotaciones es la que motiva nuestra búsqueda de una herramienta donde estén representados los conceptos que utilizarán nuestros usuarios, así como nuestra indagación en los estándares vistos en la sección 1 para representar y manipular convenientemente este conocimiento.

La primera cuestión que nos planteamos fue: ¿existe una herramienta de representación de conocimiento que podamos reutilizar o necesitaremos crear nuestra propia herramienta conceptual? En nuestro caso decidimos utilizar el tesauro Eurovoc [12], mantenido por la Oficina de Publicaciones de las Comunidades Europeas[3]. Las ventajas de este tesauro son varias. En primer lugar, el hecho de que sea el tesauro oficial que la Oficina de Publicaciones utiliza en sus sistemas de información hace de él el más robusto y estándar de los tesauros que se usan en el campo de la información jurídica. Los recursos disponibles para su mantenimiento, entre los cuales es esencial la actualización terminológica que conlleva la inclusión y eliminación de nuevos términos en versiones sucesivas, garantizan que Eurovoc es un tesauro 'vivo', que no queda obsoleto con el paso del tiempo. Su utilización por parte de un buen número de instituciones oficiales en los distintos países de la Unión Europea, como el Senado en España [5], induce su utilización en cuantos sistemas buscan interoperabilidad con cualquiera de ellos. Por último, este tesauro se puede usar libremente, en el marco de un convenio con la Oficina de Publicaciones, en el cual se aceptan las normas que subrayan el debido reconocimiento al origen del tesauro, así como las instrucciones que se deben seguir en el caso de que se proponga la extensión del tesauro con nuevos términos (la eliminación de términos está en principio restringida, de tal modo que sólo se decide eliminar términos cuando se publica una nueva versión).

Eurovoc es un tesauro multilingüe cuyo origen se remota a finales de los años setenta, momento en el que el Parlamento Europeo y la Oficina de Publicaciones Oficiales de las Comunidades Europeas deciden trabajar en la creación de un lenguaje documental común. Eurovoc está estructurado en 21 campos temáticos (dominios) y 127 microtesauros (subdominios). Contiene 6045 descriptores, 7756 no descriptores, 6669 relaciones jerárquicas recíprocas y 891 notas, datos cuantitativos que muestran el volumen y valor conceptual de esta herramien-

[3] En adelante *Oficina de Publicaciones*

ta terminológica. Eurovoc respeta las normas ISO 2788-1986 e ISO 5564-1985. Desde enero de 2009 está disponible la versión 4.3.

Tras firmar el correspondiente convenio entre la Universidad de Valladolid y la Oficina de Publicaciones, recibimos una copia del tesauro en un CD-ROM. El tesauro se distribuye en un conjunto de ficheros XML, con sus correspondientes DTD, sencillos en su estructura interna, aunque complejos en el número y organización de los ficheros. No se utiliza SKOS ni ningún otro estándar de representación de los vistos en la sección 1, razón por la cual nos vamos a referir en adelante a este formato como el 'formato propietario XML-Eurovoc de la Oficina de Publicaciones'.

Una vez disponíamos de una copia del tesauro, los pasos siguientes son buscar una API para tesauros que nos facilite el desarrollo de aplicaciones, y decidir si vamos a utilizar Eurovoc en el formato en que lo hemos recibido o preferimos una adaptación a SKOS, que parece especialmente indicado para representar tesauros en el marco de la Web Semántica [16].

3. Manipulación y representación del tesauro Eurovoc

Existen varias herramientas que permiten manipular tesauros [7,8]. No obstante, hay varios inconvenientes para el uso que pretendemos. Buscamos una herramienta que nos permita abstraernos de las particularidades del almacenamiento del tesauro y cómo está representado. Sin embargo, las herramientas encontradas son aplicaciones de usuario, que permiten crear o manipular tesauros, pero no aportan una API que se pueda utilizar desde otras aplicaciones, que es lo que buscamos. Por otro lado, no es posible importar directamente el tesauro Eurovoc tal cual lo hemos recibido. Esto nos sitúa ante la necesidad de ocuparnos del almacenamiento del tesauro y su gestión desde nuestras aplicaciones. La solución por la que optamos es desarrollar una API genérica y utilizarla desde nuestras aplicaciones.

Almacenar el tesauro supone escoger un formato de representación, para lo cual consideramos varias alternativas. La primera es almacenar el tesauro en el formato XML en que lo hemos recibido. Esta opción se descartó, ya que al ser un formato propietario, su utilización cierra la posibilidad de importar directamente cualquier otro tesauro que no esté representado con este mismo formato, lo cual incluye tesauros representados con estándares como SKOS. Así pues, para facilitar la interoperabilidad, optamos por utilizar una representación estándar para los tesauros, conforme con la propuesta SKOS.

La representación de Eurovoc con SKOS ha sido abordada en varias ocasiones, desde distintos puntos de vista, representando los dominios y microtesauros como esquemas conceptuales (*concept scheme*) y enlazándolos mediante propiedades OWL creadas ad-hoc [11] o representando los dominios como colecciones de esquemas conceptuales, que serían los microtesauros [6]. De ambas propuestas, que se ajustan a los *working draft* anteriores al año 2009 [14], hemos estudiado su posible reutilización.

A pesar de que todas ellas se ajustan a SKOS, son diferentes. Una de las razones de esta diversidad en las soluciones es el hecho de que SKOS no es aún una Recomendación estable. De hecho, existen varias diferencias entre la versión de SKOS de 2008 y la última Recomendación propuesta, de junio de 2009 [17]. Una de ellas está relacionada precisamente con la utilización de los conceptos *Collection* y *ConceptScheme* de SKOS, que son utilizados para representar los *microtesauros* de Eurovoc. La *Proposed Recommendation* de junio de 2009 introduce una variación respecto a los *working draft* previos, de modo que ya no se permite que una colección tenga como elementos a esquemas conceptuales. Esto implica que algunas de las propuestas analizadas quedarán obsoletas una vez se consolide la nueva versión como Recomendación estable.

Por otro lado, el análisis de estas propuestas también nos lleva a plantearnos las siguientes cuestiones: ¿Son compatibles entre ellas estas propuestas? ¿Serán no obstante compatibles entre ellas las nuevas propuestas que se ajusten a la nueva recomendación? Este es un aspecto importante si tenemos en cuenta que los tesauros pueden evolucionar con el tiempo, o si extendemos la experiencia a otros tesauros que no estén tan férreamente controlados en su evolución por algún organismo como es el caso de Eurovoc, en cuyo caso pueden sufrir modificaciones aún con más facilidad. La incompatibilidad entre la representación de una versión y la posterior nos situaría una vez más ante un problema de interoperabilidad.

Finalmente, si pensamos en una API genérica para tesauros, la cuestión que surge es si la elección de una u otra representación condiciona la generalidad de una API y sus implementaciones. La heterogeneidad sintáctica puede solventarse con los lenguajes de consulta apropiados, como SPARQL. Pero la elección entre *Collection* y *ConceptScheme*, dos estructuras diferentes que SKOS ofrece para representar dominios y subdominios de los tesauros, introduce heterogeneidad en un nivel superior, que no pueden resolver los lenguajes de consulta.

4. Conclusiones

La integración de las herramientas de representación de conocimiento utilizadas en distintos sistemas de información es un reto directamente relacionado con la Web Semántica. En este trabajo se ha presentado un caso de aplicación, en el que se pretende utilizar un tesauro 'estándar', y se han analizado los posibles problemas de integración que pueden surgir, tanto en el nivel conceptual como en el de representación de los tesauros.

En nuestra opinión una API pública de uso general para tesauros que utilice los estándares de representación del W3C simplificaría el desarrollo de sistemas que utilizan estas herramientas conceptuales. No obstante, la variedad de propuestas de representación encontradas para el tesauro Eurovoc, y los problemas comentados en las secciones anteriores, demuestra que la integración de conocimiento, tanto a nivel conceptual como formal, encuentra más obstáculos de lo que en principio podría parecer, incluso cuando se utilizan estándares para la representación de semántica.

Referencias

1. Alvite Díez, M. L. Las bases de datos jurídicas y el uso del lenguaje XML en España. *SCIRE. Representación y Organización del Conocimiento 15*, 1 (Enero-Junio 2009), 33–57. ISSN 1135-3716.
2. Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. *Scientific American* (May 2001).
3. Biagioli, C., Francesconi, E., and Sartor, G., Eds. *Proceedings of the V Legislative XML Workshop, Florencia, Italia, 14-16 June 2006* (2007), European Press Academic Publishing.
4. Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., and Siméon, J. XQuery 1.0: An XML Query Language. W3C Recommendation 23 January 2007, W3C, the World Wide Web Consortium, http://www.w3.org/TR/2007/REC-xquery-20070123/, Jan. 2007.
5. Cueto Aparicio, M. Eurovoc thesaurus use at the Senate of Spain. In *Information in Africa project workshop* (27-30 June 2006).
6. Faro, S., Francesconi, E., Marinai, E., and Sandrucci, V. EUROVOC Studies LOT2 D2.3 -Report on execution and results of the interoperability tests. Tech. Rep. 10118, Publications Office of the EC, Institute of Legal Information Theory and Techniques ITTIG, Jan. 2008.
7. Ferreyra, D. TemaTres: software libre para gestión de tesauros. URI: http://www.r020.com.ar/tematres/index.html.
8. Lacasta, J., Nogueras, J., López-Pellicer, F. J., Muro-Medrano, P., and Zarazaga-Soria, F. ThManager: An Open Source Tool for creating and visualizing SKOS. *Information Technology and Libraries (ITAL) 26*, 3 (2007), 39–51.
9. Martínez González, M. M., Vicente Blanco, D.-J., de la Fuente Redondo, P., Adiego Rodríguez, J., Pisabarro Marrón, A. M., and Sánchez Felipe, J. M. Estructura, semántica, extracción de información y XML legislativo: experiencias en la Universidad de Valladolid. *SCIRE. Representación y Organización del Conocimiento 15*, 1 (Enero-Junio 2009), 173–186. ISSN 1135-3716.
10. Martínez, M. M., de la Fuente, P., and Derniame, J.-C. XML as a means to support information extraction from legal documents. *International Journal of Computer Systems Science and Engineering 18*, 5 (Sept. 2003), 263–277.
11. Paredes, L. P., Maria, J., Rodriguez, A., and Azcona, E. R. Promoting Government Controlled Vocabularios for the Semantic Web: the EUROVOC Thesaurus and the CPV Product Classification System. In *The Fifth European Semantic Web Conference, 1-5 june 2008, Tenerife, Spain* (2008), vol. 50212 of *Lecture Notes in Computer Science*, Springer, pp. 111–122.
12. Publications Office. *Eurovoc thesaurus*. Accesible en http://europa.eu/eurovoc (última consulta: 10/07/2009).
13. Vicente, D.-J., Martínez, M., Sánchez, J. M., and Adiego, J. Experiences on teaching international private law with the support of an e-learning tool. In *The LEFIS virtual campus design* (Albarracín (España), May 2007). Avaible online at http://www.lefis.org/meetings/workshops/2007/albarracin_2007/contenido/albarracin2007_damaso.ppt.
14. W3C, the World Wide Web Consortium. *SKOS Simple Knowledge Organization System Reference. W3C Working Draft 29 August 2008*, Aug. 2008. http://www.w3.org/TR/2008/WD-skos-reference-20080829/.

15. W3C, the World Wide Web Consortium. *SPARQL Query Language for RDF. W3C Recommendation 15 January 2008*, Jan. 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.
16. W3C, the World Wide Web Consortium. *SKOS Simple Knowledge Organization System Primer*, June 2009. W3C Working Draft. http://www.w3.org/TR/2009/WD-skos-primer-20090615/.
17. W3C, the World Wide Web Consortium. *SKOS Simple Knowledge Organization System Reference. W3C Proposed Recommendation 15 June 2009*, June 2009. http://www.w3.org/TR/2009/PR-skos-reference-20090615/.

Generación de Tesauros basado en Media Wiki

Ana Flores Cuadrado[1], Eduardo Villoslada de la Torre[13], y Alberto Peláez Gutiérrez[2]

[1] Telefónica Investigación y Desarrollo
Parque Tecnológico de Boecillo, Parcela 120, 47151. Boecillo. Valladolid, España
{anafc,evdlt}@tid.es
[2] Telefónica Soluciones
Parque Tecnológico de Boecillo, Parcela 120, 47151. Boecillo. Valladolid, España
{albertojavier.pelaezgutierrez}@telefonica.es
[3] Universidad de Valladolid, Edificio TIT.
Campus "Miguel Delibes" s/n, 47011. Valladolid, España

Resumen. La Wikipedia alberga a través de su estructura, de forma implícita, un conjunto de relaciones conceptuales muy detalladas y evolución dinámica. Con el objetivo de aprovechar esa información para mejorar los procesos de búsqueda en los sistemas de gestión del conocimiento, se ha definido un algoritmo que permita materializar la estructura conceptual de la Wikipedia y sus relaciones para construir un tesauro.

Palabras clave: Tesauros, SKOS, MediaWiki, Wikipedia, screen scrapper

1. Introducción

Los sistemas de gestión del conocimiento de una empresa tienen como objetivo compartir y difundir el conocimiento sobre tecnologías y procedimientos entre sus trabajadores. Sin embargo, el rendimiento que se obtienen de estos sistemas es mínimo. A menudo, estos sistemas son utilizados como simples repositorios de documentos, donde los usuarios difícilmente localizan la información usando buscadores basados en el uso palabras claves [23] o búsquedas por metadatos. Aunque el uso las tecnologías semánticas y en especial el uso de tesauros [27], puede mejorar la precisión de las consultas (recall), existen dificultades para su aplicación en el ámbito de las tecnologías de la información, debido a que no existe ningún vocabulario específico para este ámbito y que aunque los vocabularios más generales como *Wordnet* [21] incluyen términos de este dominio, su actualización no es lo suficiente dinámica para soportar la rápida aparición de nuevos conceptos.

Con la aparición del concepto de Web 2.0 [22], Internet se ha constituido como un vehículo de participación ciudadana que permite la creación y difusión del conocimiento de forma dinámica. Una de las herramientas Web 2.0 más populares es la Wikipedia [30], donde los usuarios de forma voluntaria se encargan de introducir información sobre los temas más novedosos, en distintos idiomas. En la Wikipedia, cada página o artículo representa un concepto, y la tecnología que soporta la Wikipedia permite relacionar de forma implícita los distintos conceptos, a través de la estructura de sus páginas.

El objetivo de nuestro trabajo es materializar la estructura conceptual de la Wikipedia, y de sus relaciones y construir con él un tesauro que pueda utilizarse en los sistemas de gestión del conocimiento para mejorar los procesos de búsqueda, por una parte, utilizando las relaciones definidas en el tesauro para expandir las consultas [15,14] y así mejorar la precisión en las consultas, y por otra utilizar este tesauro como parte de un mapa semántico que a modo de asistente vaya guiando a los usuarios en los procesos de búsqueda, y permitiéndoles navegar a través de los distintas relaciones entre conceptos con el objetivo de refinar las búsquedas [14].

En la Sección 2 se describen los principales aspectos que son la base tecnológica durante el proceso de construcción del tesauro. En la Sección 3 se presenta el proceso de mapeo de la estructura de la Wikipedia en base al análisis previo de ésta. En esta sección también se presenta la arquitectura de la aplicación así como el algoritmo utilizado en el proceso de mapeo. La Sección 4 presenta los trabajos relacionados y por último en la Sección 5 se concluye con las ventajas y limitaciones del trabajo presentado y las líneas trabajo futuras.

2. Base Tecnológica

El objetivo de esta sección es describir los aspectos tecnológicos que son necesarios conocer para comprender el análisis realizado de la Wikipedia y la tecnología que sustenta la representación de tesauros.

2.1. Tecnología Media Wiki

Se entiende por Wiki una herramienta web colaborativa, accesible mediante un navegador, cuyo contenido puede ser visualizado y editado directamente por cualquier usuario con permisos sin necesidad de tener unos conocimientos especializados. La primera Wiki fue creada en 1995 por Ward Cunningham [28]. Posteriormente en 2001 se usó este modelo para crear *Wikipedia* utilizando un software específico para realizar wikis: *MediaWiki* [18]. Dentro de *Wikipedia* se distinguen cinco tipos de elementos primarios [11,4]:

- **Páginas de Artículos**: texto que describe un concepto; es la entidad básica de MediaWiki.
- **Páginas de Re-dirección**: con el fin de asegurarse de que existe un único artículo por concepto, MediaWiki provee re-direcciones de otros artículos hacia el principal. De esta es la forma se maneja diferentes formas de un mismo término (por ejemplo: ortografías múltiples, seudónimos, mayúsculas/minúsculas, abreviaturas, sinónimos) [24].
- **Enlaces a artículos**: dirigen al usuario de un artículo a otro. Las etiquetas de los enlaces no tienen que ser iguales al título del artículo enlazado, pueden usarse sinónimos, términos relacionados y variaciones lingüísticas.
- **Categorías**: MediaWiki también ofrece la posibilidad a los autores de incluir categorías a modo de metadatos para la clasificación de artículos. Estas categorías puede ser utilizadas para encontrar artículos relacionados.
- **Páginas de Desambiguación**: Páginas que contienen un conjunto de enlaces a artículos con los diferentes sentidos que puede tener el término buscado, es decir, describen múltiples conceptos asociados con un mismo término.

2.2. SKOS

Un *tesauro* es un conjunto de conceptos o términos de un vocabulario controlado organizados en una estructura jerárquica, dónde cada término puede tener una o más relaciones *padre-hijo* con otros términos [27]. En los tesauros es posible representar otro tipo de relaciones como las *relaciones de equivalencia* o de sinonimia, y las *relaciones asociativas* o *related* [12].

Simple Knowledge Organization System (SKOS)[7], es un vocabulario desarrollado por el *World Wide Web Consortium (W3C)*, y que actualmente es el modelo ideal para representar cualquier tipo de vocabulario controlado, y especialmente para representar tesauros en *Resource Description Framework (RDF)* [16]. A diferencia de *Web Ontology Language (OWL)* [2,1], que permite expresar estructuras conceptuales complejas donde el tipo de relaciones

que se establecen entre los elementos depende del dominio, *SKOS* permite permite la representación de vocabularios más complejos, incluyendo ontologías [5] *OWL* [25], mediante esquemas sencillos de conceptos. En *SKOS*, los elementos únicamente se pueden relacionar por relaciones jerárquicas, asociativas o de equivalencia, lo que facilita su tratamiento, cuando se quiere utilizar la navegación entre las relaciones de los elementos como ayuda o asistencia a los usuarios en las búsquedas.

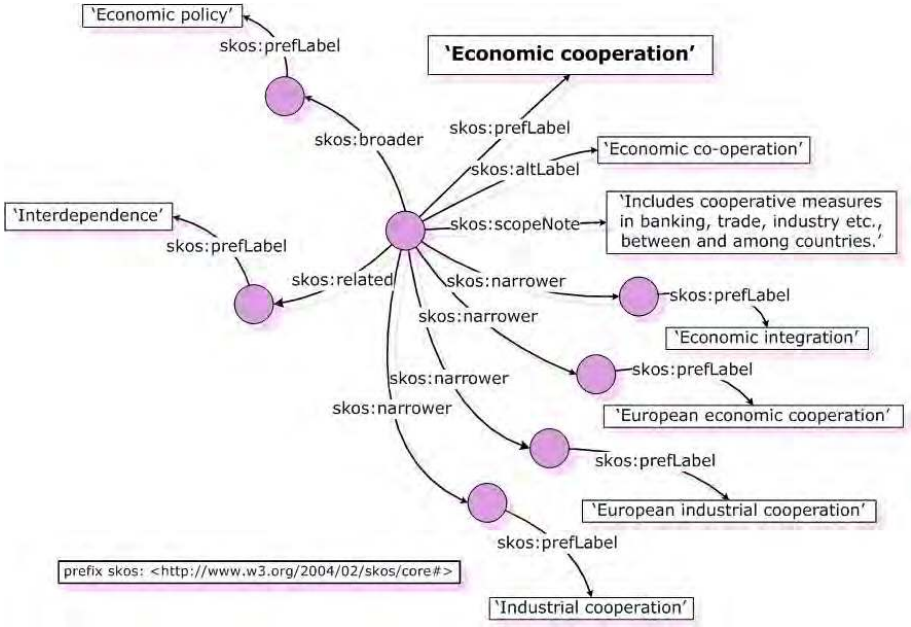

Fig. 1. Representación de un Tesauro usando SKOS

En *SKOS* (figura 1), cada concepto (*skos:Concept*) se identificada por una *Uniform Resource Identifiers (URI)* y representa una colección de términos sinónimos. Uno de los elementos de esta colección es el descriptor del concepto (*skos:prefLabel*) y se utiliza como identificador único de cada concepto. El resto de los sinónimos se representan como términos alternativos (*skos:altLabel*) asociados al descriptor, pudiendo existir un número indefinido de ellos. En el caso de los tesauros multi-idiomas, además se puede establecer relaciones entre un mismo concepto en distintos idiomas, mediante el atributo *lang* de la etiqueta (*skos:prefLabel*). Aunque sólo es posible definir una etiqueta preferida por idioma y tantas etiquetas alternativas como se desee en los diferentes idiomas. Análogamente a los tesauros, las relaciones semánticas que se pueden establecer en *SKOS* son relaciones jerárquicas, donde el término padre (*skos:broader*) suele representar el término más amplio o genérico y el termino hijo representa al más concreto o especifico (*skos:narrower*). Las *relaciones de equivalencia* o entre sinónimos se representa con las etiquetas (*skos:altLabel*) y (*skos:prefLabel*). Y finalmente se pueden establecer *relaciones asociativas* o *related* entre términos del tesauro que no están dentro de la misma jerarquía, pero que tienen cierta relación entre ellos mediante la etiqueta (*skos:related*).

3. Proceso de Construcción del Tesauro

En esta sección se presenta el proceso de mapeo de la estructura de la Wikipedia y de construcción del tesauro, así como la arquitectura de la aplicación utilizada para efectuar el proceso de mapeo.

3.1. Análisis de la información de la Wikipedia

La forma de realizar la extracción de la información de Media Wiki y de realizar la transformación entre sus elementos y los conceptos de *SKOS* (sección 2.2) hay que establecerla basándose en los contenidos de la *MediaWiki* (sección 2.1). Esta transformación no es única, sino que pueden darse distintas interpretaciones y pueden generarse distintos algoritmos en función de los objetivos del tesauro a generar.

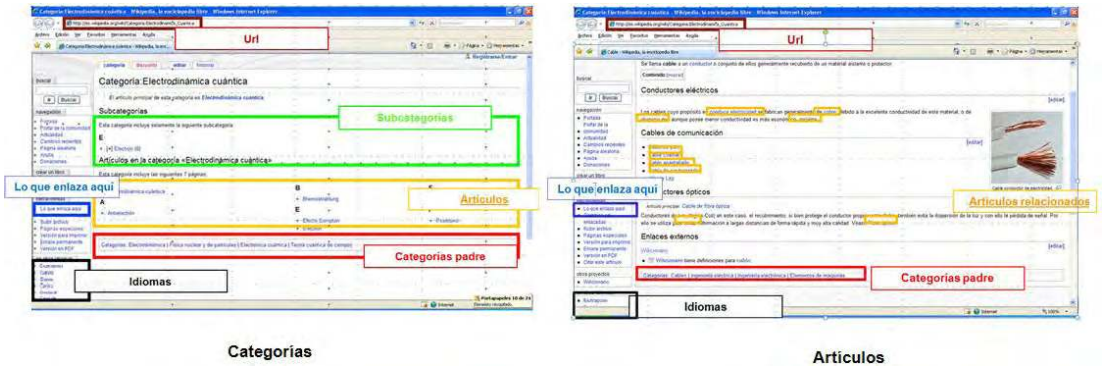

Fig. 2. Esquemas de las páginas de categorías ya artículos.

El algoritmo aquí propuesto trata de ofrecer una visión más general e independiente del dominio del tesauro. Para generar este algoritmo se ha analizado la estructura de las páginas de MediaWiki para determinar su relación con cada elemento del vocabulario *SKOS*. Para ello, en la figura 2 se muestran las distintas partes de las páginas de categorías y de artículos de donde se extrae la información que almacenará el fichero *SKOS*. La conclusion obtenida del análisis es la siguiente:

- **Identificador del concepto** (*skos:Concept*): Se obtiene de la *Uniform Resource Locator (URL)* de la página de la wikipedia.
- El descriptor o **elemento identificador** del concepto (*skos:prefLabel*): Se obtiene directamente de la parte final *URL* de los enlaces.
- El conjunto de **sinónimos o *Alternatives*** (*skos:altLabel*): Serán las redirecciones a la página actual y se obtiene de los enlaces cuyo `id` de la página `mw-whatlinkshere-list` (`Lo que enlaza aquí`).
- La **relaciones jerárquicas** (padre-hijo) o ***Broader/Narrower***: Cada artículo o categoría pertenece a una o varias categorías, que son asignadas como *skos:broader*. Se identifican como aquellos enlaces cuyo `id` es `mw-normal-catlinks`.
  La asignación de *skos:narrower* no se realiza de forma explícita, sino que se diseña el software para que cuando se asigna un término *broader*, el tesauro incluya automáticamente la relación *narrower* correspondiente.
- La relaciones **asociativas** (*skos:related*): La estructura de las páginas de categorías y de artículos es diferente, por lo hay que diferenciar:

  - para los artículos se toman como *related* los enlaces del contenido del artículo a otros artículos, que vendrán identificados con `column-content` en el `id`.
  - para las categorías se toman como *related* aquellos padres que estén relacionados dentro de un mismo concepto; es decir, si un concepto pertenece a varias categorías se asume que hay cierta relación entre ellas.
- **Idiomas**: La página cargada establece un idioma para el término de *preflabel*, pero es posible extraer los términos en otros idiomas a través de los enlaces cuyo `id` es `p-lang`.

3.2. Arquitectura

Para la extracción de la información de la Wikipedia la arquitectura propuesta (figura 3) está formada por un *spider*, un *parser*, un módulo encargado del procesamiento de la información y otro del almacenamiento, en este caso a fichero utilizando **Jena** [8]. El *spider*, es el software que recorre las páginas a partir de sus enlaces, tomando como punto de partida la establecida por el usuario; se ha utilizado **JSpider** [10] y para controlar su comportamiento y filtrar las páginas a parsear se ha desarrollado un *plugin* específico. El *parser*, permite extraer la información de las páginas a partir de una etiqueta *HTML* recuperando y procesando su contenido; se ha utilizado **Jericho HTML** [9].

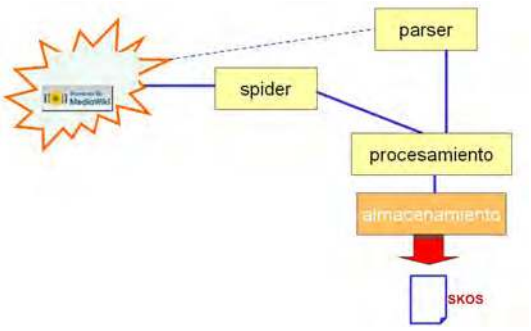

Fig. 3. Arquitectura de generación del fichero *SKOS*.

El algoritmo para generar el fichero *SKOS* (figura 4) se divide en dos fases:

En la primera etapa el *spider* construye la estructura del tesauro: a partir de una página inicial, según se recorre la Wiki se consigue recuperar todos los conceptos (*URI* y *preflabel*), así como sus relaciones (*broader*, *narrower* y *related*). El tamaño de la Wiki, en concreto, en el caso de Wikipedia, hace que el espacio y el tiempo de extracción de los conceptos aumente exponencialmente, por lo que interesa definir estrategias de parada para indicar al *spider* el momento de finalizar su labor de una forma ordenada. Se han planteado las posibles opciones:

- **Parada por número de conceptos**: cuando el *spider* alcance el número de conceptos configurado pasará a la siguiente fase.
- **Parada por niveles**: cuando el *spider* da el número de saltos configurado pasará a la siguiente fase.

La segunda fase recorre el tesauro generado completando la información de cada uno de los conceptos: *alternatives* y *preflabel* en distintos idiomas; así mismo para asegurar una estructura adecuada del tesauro se asigna *broader* a todos aquellos conceptos que no los tengan y se establecen como *Top Concepts* del tesauro aquellos que no tienen *broader*.

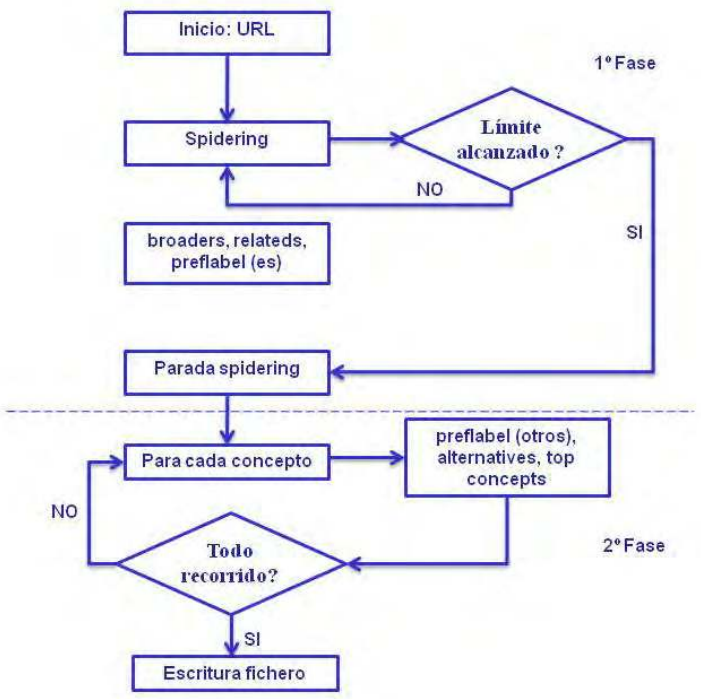

Fig. 4. Algoritmo de generación del fichero SKOS.

3.3. Experimentación

Usando la arquitectura anterior se han realizado diversas pruebas de generación de tesauros, tanto sobre Wikipedia, como desde otras Wikis. En la tabla 1 se recogen los datos de la generación del tesauro a partir de la categoría Microcontroladores de Wikipedia, configurado para recuperar 20 idiomas por concepto.

Descripción	**Número conceptos** (1ª Vuelta/2ª Vuelta)	**Tiempo** (1ª Vuelta/2ª Vuelta)	**Memoria** (1ª Vuelta/2ª Vuelta)
3 Niveles, sin límite de conceptos (Microcontroladores)	7549 / 10767	20 min. / 3h 56min.	29MB / 635MB
1000 Conceptos, sin niveles (Microcontroladores)	1000 / 1754	1 min. 20s. / 35 min.	17MB / 126MB

Tabla 1. Ejecución del algoritmo sobre Wikipedia

Analizando los datos de las ejecuciones, se observa que, independientemente de la estrategia de finalización, existe una diferencia entre los conceptos recuperados en la primera vuelta y en la segunda debido a que en la segunda parte del algoritmo, se crean todos los *broaders* de cada uno de los conceptos obtenidos en la primera vuelta, que aún no existan. En cuanto al tiempo de procesamiento, la segunda parte del procesamiento es bastante más

alta que la primera porque en la segunda parte se entra de nuevo en dos páginas para cada concepto. Para el concepto de partida el resultado obtenido en cualquiera de los casos es el de la figura 5.

```
<rdf:Description rdf:about="http://es.wikipedia.org/wiki/Categor\%C3\%ADa:Microcontroladores">
   <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core\#Concept"/>
     skos:prefLabel xml:lang="fi">Mikrokontrollerit </skos:prefLabel>
     <skos:prefLabel xml:lang="en">Microcontrollers </skos:prefLabel>
     <skos:prefLabel xml:lang="pl">Mikrokontrolery </skos:prefLabel>
     <skos:prefLabel xml:lang="es">Microcontroladores </skos:prefLabel>
     <skos:prefLabel xml:lang="fr">Microcontrôleur </skos:prefLabel>
     <skos:prefLabel xml:lang="da">Mikrocontrollere </skos:prefLabel>
...
     <skos:narrower rdf:resource="http://es.wikipedia.org/wiki/PIC16F84"/>
     <skos:narrower rdf:resource="http://es.wikipedia.org/wiki/AMD_29000"/>
     <skos:narrower rdf:resource="http://es.wikipedia.org/wiki/Logochip"/>
     <skos:narrower rdf:resource="http://es.wikipedia.org/wiki/Freescale_68HC08"/>
     <skos:narrower rdf:resource="http://es.wikipedia.org/wiki/MPLAB"/>
     <skos:narrower rdf:resource="http://es.wikipedia.org/wiki/Serie_A789_de_Atmel"/>
...
     <skos:related rdf:resource="http://es.wikipedia.org/wiki/Categor\%C3\%ADa:Microprocesadores"/>
     <skos:related rdf:resource="http://es.wikipedia.org/wiki/Categor\%C3\%ADa:Electr\%C3\%B3nica"/>
    <skos:related rdf:resource="http://es.wikipedia.org/wiki/Categor\%C3\%ADa:Intel"/>
</rdf:Description>
```

Fig. 5. Resultado de la experimentación

Puesto que el objetivo no concierne únicamente a Wikipedia, sino a la tecnología MediaWiki se han realizado pruebas con otras Wikis que utilizan dicha tecnología, como Madripedia [13], Qwiki [26] o WikiMusicGuide [29]. En estos casos lo que se ha comprobado es hay ciertos parámetros de la Wiki que hay contemplar en el algoritmo de extracción, como el contexto del servidor o las palabras reservadas para categorías y desambiguación.

3.4. Aplicación

La aplicación más inmediata es utilizar el tesauro generado como índice rápido para el manejo de una Wiki, de tal forma que se presente el mapa de conceptos de forma explícita y se pueda navegar por él, hasta llegar a los conceptos deseados, en vez de tener que realizar la búsqueda y navegación de forma explícita a través de la Wiki; una vez identificados los conceptos de interés se podría acceder a ellos directamente a través del enlace. Esta navegación por el mapa de conceptos es aún más útil cuando la Wiki es multiidioma, ya que el tesauro mostrará esa información y es posible ver que dependiendo del idioma las relaciones son más completas, en función de los completos que sean los contenidos en cada idioma.

Al margen de esto, puesto que el tesauro formaliza la relación entre conceptos, puede servir de fuente para realizar un asistente de búsquedas que aproveche la relación entre los conceptos para afinar las búsquedas en otros contextos ya que visualizando los conceptos *related* extraídos, se puede hacer una idea rápida sobre cierto tema y ver que otros conceptos están implicados en lo que se buscando, para así ubicarse mejor en aquello que se pretende buscar.

Un punto de vista complementario es no sólo utilizar el tesauro para afinar los criterios de búsqueda, sino para aplicarlo en la formación y priorización de los resultados obtenidos en la búsqueda, de forma que los primeros resultados serían los que están más ligados

semánticamente al concepto buscado y todos los que le rodean. O en este mismo sentido, no sólo hacer búsquedas por los *preflabel*, sino también por los correspondientes *alternatives*.

4. Trabajos Relacionados

Existen variadas iniciativas con propuestas para estructurar los contenidos Wikipedia, como recoge [3] o [11], pero en estos casos la estructura obtenida no es un tesauro y si bien muestran posibilidades de información a extraer no son comparables a esta propuesta. Existe un trabajo para generar tesauros a partir de Wikipedia en [4], que tiene como objetivo realizar un mapeo de palabras a su concepto correspondiente, a partir de una base previa de palabras (nombre propios y comunes), utilizando Wikipedia como fuente de información de los conceptos, para de esta forma facilitar búsquedas y lograr una mayor precisión. Y es en base a este objetivo como se define el algoritmo de extracción y generación del tesauro. En este caso los títulos de los artículos son utilizados para definir los *preflabel*, las páginas de redirección generan los *altlabel* y se examinan todos los enlaces de los artículos para establecer relaciones entre conceptos (*related*), dentro de los cuales diferencian dos tipos de relaciones: las débiles, en las cuales desde un concepto se referencia a otro, pero no a la inversa, y las fuertes en las cuales la referencia es bidireccional. La cuestión es que *related* expresa una relación simétrica, por lo que la decisión es si incluir las relaciones débiles o solo las fuertes. Aunque hay ideas interesantes, como el tratamiento de *related*, el algoritmo propuesto está muy asociado al objetivo planteado y no puede ser considerado dentro de una visión más general, como la aquí propuesta; por ejemplo, no plantean una estrategia para establecer relaciones *broader* ni *narrower*, y al utilizar únicamente el inglés, tampoco hacen tratamiento de los idiomas, con lo que se pierde conocimiento existente en Wikipedia.

5. Conclusiones y Líneas Futuras

En este artículo se ha presentado un algoritmo que materializa la estructura conceptual de la Wikipedia, y de sus relaciones, en un tesauro que permitirá fácilmente mejorar las búsquedas en los sistemas de gestión del conocimiento aplicando sus propiedades de sinonimia, traducción, generalización y particularización conceptual.

Aunque existe distintas aproximaciones similares[17,20,6,19], el algoritmo aquí propuesto trata de ofrecer una visión más general e independiente del dominio del tesauro, permitiendo generar de forma sencilla tesauros multi-idioma en torno a un concepto dado a partir de la estructura de la Wikipedia. Adicionalmente, al ser un algoritmo basado en Media Wiki, puede ser aplicado a cualquier Wiki realizada con dicha tecnología, permitiendo la creación de un mapa conceptual completo y navegable de la información de las Wikis empresariales.

El trabajo aquí presentado,únicamente se basa en la generación de tesauros, pero no considera la actualización de los mismos, ya que ante cambios sustanciales del mapa de conceptos es posible volver a generar el tesauro; no obstante como líneas futuras se plantea la posibilidad de contemplar cambios incrementales en el mapa de conceptos o tesauro.

Referencias

1. G. Antoniou and F. vanHarmelen. *A Semantic Web Primer*. MIT Press, Cambridge, 2004.
2. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrock, D.L. McGuinness, P.F. Patel-Schneider, L.A. Steinnd, and F.W. Olin. Owl web ontology language reference, February 2004.
3. dbpedia. http://dbpedia.org.
4. A. Gregorowicz and M A. Kramer. Mining a large-scale term-concept network from wikipedia. *Mitre Technical Report*, 6(1).

5. T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.
6. Bachlechner D. Hepp, M. and K. Siorpaes. Harvesting wiki consensus - using wikipedia entries as ontology elements. *First Workshop on Semantic Wikis*, 2006.
7. A. Isaac and E. Summers. Skos simple knowledge organization system primer, February 2008.
8. jena. http://jena.org/.
9. jericho. http://jericho.org/.
10. jspider. http://jspider.org.
11. S. Kruger. Acquiring the semantic relationships of links between wikipedia articles. researh proposal. http://www.coloradostoutenburg.com/cs58920v1.doc.
12. M.J.Lamarca. Tesauros. http://www.hipertexto.info/documentos/tesauros.htm.
13. Madripedia. http://www.madripedia.es/.
14. E. Mäkelä. Survey of semantic search research. In *Proceedings of the Seminar on Knowledge Management on the Semantic Web*, 2005.
15. C. Mangold. A survey and classification of semantic search approaches. *Int. J. Metadata Semant. Ontologies*, 2(1):23–34, 2007.
16. F. Manola and E. Miller. Rdf primer, February 2004.
17. Legg C. Milne D. Medelyan, O. and I.H Witten. Mining meaning from wikipedia. Hamilton, New Zealand: University of Waikato, Department of Computer Science. Computer Science Working Papers, November 2008.
18. MediaWiki. http://es.wikipedia.org/wiki/MediaWiki.
19. D Milne and Nichols D. M. Witten, I.H. Extracting corpus specific knowledge bases from wikipedia. Technical Report Working Paper 03/2007, University of Waikato, 2007.
20. Medelyan O. Milne, D. and I.H. Witten. Mining domain-specific thesauri from wikipedia: A case study. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 442–448, Washington, DC, USA, 2006. IEEE Computer Society.
21. D.I Moldovan and R. Mihalcea. Using wordnet and lexical operators to improve internet searches. *IEEE Internet Computing*, 4(1):34–43, 2000.
22. T. O'Reilly. What is web 2.0—design patterns and business models for the next generation of software, September 2005. http://www.oreillynet.com/lpt/a/6228.
23. T.B. Passin. *Explorer's Guide to the Semantic Web*. Manning Publications Co, 2004.
24. Redirecciones. http://es.wikipedia.org/wiki/Wikipedia:Redirecciones.
25. Simple Knowledge Organization System. http://en.wikipedia.org/wiki/SKOS.
26. Stanford University. Qwiki. http://qwiki.stanford.edu/.
27. Pidcock W. What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model?, January 2003. http://www.metamodel.com/article.php?story=20030115211223271.
28. Wiki. http://es.wikipedia.org/wiki/Wiki.
29. WikiMusicGuide. http://www.wikimusicguide.com.
30. Retrieved Wikipedia: Wiki, September 2008. http://en.wikipedia.org/wiki/Wiki.

Un Editor de Modelos OWL-S: OWL-S Modeller

Ismael Navas-Delgado, Amine Kerzazi and José F. Aldana-Montes,

E.T.S. de Ingeniería Informática, Lenguajes y Ciencias de la Computación, Universidad de Málaga, Boulevard Louis Pasteur 35,
29071 Málaga, España
{ismael, kerzazi, jfam}@lcc.uma.es

Resumen. El esfuerzo colaborativo de varias organizaciones ha dado lugar a la ontología OWL-S, una ontología para la anotación de servicios Web. Esta ontología permite conocer tres datos básicos sobre un servicio: qué proporciona al cliente, cómo se usa y cómo se interacciona con él. Otros grupos de trabajo, en busca de una estandarización en el área de la Web Semántica han desarrollado WSMO, que proporciona un modelo conceptual más completo que OWL-S. El principal problema de OWL-S es la falta de herramientas integradas para el diseño de las anotaciones realizadas sobre los servicios Web, mientras que WSMO dispone de un entorno de diseño y desarrollo completo: WSMO Studio. En este trabajo se aborda el diseño e implementación de una extensión de WSMO Studio para permitir el diseño de anotaciones de servicios mediante OWL-S. De esta forma los desarrolladores de servicios Web Semánticos pueden hacer uso de un mismo entorno para producir servicios anotados con ambas propuestas.

Palabras clave: Herramientas, Servicios Web Semánticos, OWL-S, WSMO.

1 Introducción

El gran valor de la Web reside en que es el mayor repositorio de información disponible en un formato pensado para personas y accesible a través de mecanismos de interacción amigables. Evidentemente, esa ventaja se torna en inconveniente si la información que ofrece un sitio Web es útil en algún proceso de negocio automatizado, pues entonces es necesario que los agentes software del proceso sean capaces de localizar los datos que necesitan usando mecanismos de interacción y formatos que no han sido diseñados para ese objetivo. La implementación de una correcta estrategia de integración de aplicaciones permite incorporar nuevas aplicaciones de manera más eficiente, modificar procesos a bajo costo, automatizar y optimizar procesos que requerían intervención humana y obtener información consolidada. En este sentido, La Web Semántica [1] se define como una extensión de la actual Web la cual pasaría de ser una Web de documentos a una Web de datos, intentando solucionar de este modo los problemas existentes en la Web actual. Frente a la semántica implícita presente, el crecimiento caótico de recursos, y la ausencia de una organización clara de la Web actual, la Web Semántica aboga por clasificar, dotar de estructura y anotar los recursos con semántica explícita procesable

automáticamente. Un problema de la Web Semántica es el de alcanzar un entendimiento entre usuario, desarrollador y aplicaciones, y para conseguirlo se hace uso de las ontologías. La idea fundamental es que la Web Semántica esté formada por una red de nodos tipificados e interconectados mediante clases y relaciones definidas por una ontología compartida por sus distintos autores. Pero no todo serán redes y compartición de recursos, las aplicaciones Web también proporcionan servicios. Los Servicios Web Semánticos proponen describir no sólo información sino funcionalidades y procedimientos para describir Servicios Web: sus entradas y salidas, las condiciones necesarias para que se puedan ejecutar, los efectos que producen, o los pasos a seguir cuando se trata de un servicio compuesto. Dichas descripciones procesables por máquinas permitirían automatizar el descubrimiento, la composición, y la ejecución de servicios, así como la comunicación entre unos y otros.

En este contexto WSMF [2] tiene como objetivo era el de permitir un *E-commerce* basado en Servicios Web de una manera completa, escalable y flexible. Para ello, WSMF provee el modelo conceptual apropiado para desarrollar y describir Servicios Web y sus composiciones. Se basa en dos principios complementarios: fuerte desacoplamiento de los múltiples componentes que realiza una aplicación de *E-commerce* y fuerte servicio de mediación, que permite a cualquier aplicación comunicarse con otros servicios y aplicaciones de una manera escalable.

Por otro lado, como alternativa más cercana a los estándares del W3C surge OWL-S [3], que define una ontología OWL de servicios producto del esfuerzo colaborativo de investigadores de varias organizaciones. Esta ontología sigue en pleno desarrollo, y conectándose con diversos proyectos, como aquellos consistentes en construir ontologías de tiempo y recursos. A menudo, OWL-S es referido como un lenguaje para describir servicios, reflejando el hecho de que nos proporciona un vocabulario estándar que puede usarse junto con los otros aspectos del lenguaje OWL para crear descripciones de servicios.

Pese a estar basado en OWL existen pocas herramientas que den soporte a la anotación de servicios Web mediante OWL-S, y las herramientas existentes resultan muy complejas para esta tarea. Sin embargo WSMF proporciona un entorno integrado de diseño de anotaciones sobre servicios Web de uso más sencillo, WSMO Studio [4].

En este artículo presentamos una extensión de WSMO Studio para el diseño de anotaciones de servicios Web con OWL-S. De esta forma los desarrolladores de servicios Web semánticos pueden hacer uso de un entorno unificado para diseñar anotaciones para estos servicios usando ambas propuestas.

En la Sección 2 se describen los trabajos relacionados con la herramienta presentada en este artículo. La Sección 3 presenta dicha herramienta, *OWL-S Modeller*. Finalmente, la Sección 4 introduce las conclusiones y trabajos futuros.

2 Trabajos relacionados

Existen distintas aplicaciones para la generación de descripciones de Servicios Web en OWL-S. A continuación veremos algunas de objetivos similares a la herramienta presentada en este artículo.

University of Malta OWL-S Editor [5] es una aplicación orientada a para ayudar a los usuarios o programadores inexpertos a crear descripciones de Servicios Web en OWL-S. La creación de descripciones se realiza mediante asistentes que van pidiendo los atributos necesarios en cada momento. Una vez terminado el proceso genera el archivo con la descripción OWL-S. La construcción de descripciones no es dinámica, ya que hay que realizar todos los cambios a la vez para la posterior creación del documento. Tiene la opción de editar descripciones existentes, pero con el mismo sistema de asistentes que dan poco dinamismo a la tarea. Por otro lado, se pueden visualizar los grafos de las descripciones, pero no permite realizar cambios gráficamente.

CMU's OWL-S Development Environment (CODE) [6] es un ambicioso proyectoque pretende aprovechar todos las herramientas para la Web Semántica desarrolladas por CMU, desde el parser, pasando por el generador de WSDL desde código Java, el conversor de WSDL a OWL-S, y terminando por OWL-S Editor. De entre las herramientas integradas en CODE, la más relevante para los propósitos del trabajo presentado en este artículo es CMU OWL-S IDE [7], un entorno de desarrollo de OWL-S en el cual se incluye el plug-in CMU OWL-S Editor. Se trata de un plug-in de eclipse que pretende realizar todas las funciones de tratamiento de documentos OWL-S: creación, edición, composición, etc. También cuenta con un *Process Editor*, que permite crear descripciones de procesos.

Protègè [8] es actualmente la plataforma más usada y conocida para la construcción de ontologías para la Web Semántica, particularmente ontologías OWL. Se trata de una plataforma libre, cuyo propósito es poder modelar bases de conocimiento. Este entorno de desarrollo permite crear dominios de conocimiento mediante una interfaz gráfica y de fácil uso, aunque de un uso más técnico y complejo que la aplicación OWL-S Editor vista anteriormente. Protègè puede ser extendido gracias a la arquitectura de plug-ins que posee y a la API basada en Java para la creación de herramientas y aplicaciones de bases de conocimiento. Esto ha hecho que se genere en torno una comunidad que contribuye activamente ampliándolo mediante la creación de plug-ins, lo que lo convierte en una herramienta bastante potente. Existe un plug-in para Protègè pensado en OWL-S, llamado también OWL-S Editor [9]. Dicho editor es un entorno gráfico para creación y edición de descripciones OWL-S. El plug-in OWL-S Editor es excesivamente técnico y poco intuitivo como para la utilización por parte de usuarios poco experimentados, siendo las generaciones de descripciones más textuales que gráficas y mostrando los árboles de procesos mediante estructuras tipo browser como JTree[1].

OWL-S Matcher (OWLSM) [10] nace de una herramienta similar para descripciones DAML-S, es una implementación Java de un algoritmo matchmaking para emparejar descripciones OWL-S. La aplicación compara dos descripciones (una del que realiza la petición del servicio y otra del que lo proporciona) e identifica las diferentes relaciones entre ambas descripciones. Esta comparación se realiza mediante la evaluación de las entradas y salidas de cada proceso, además de las categorías del servicio.

WSMO Studio [4] es el sucesor de SWWS Studio, y como tal, se trata de un entorno de desarrollo código abierto para el modelado de Servicios Web Semánticos y

[1] http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JTree.html

procesos de negocios semánticos. Este entorno es un conjunto de plug-ins de Eclipse, lo que le proporciona una gran extensibilidad, permitiendo a las comunidades de usuarios poder desarrollar herramientas para esta aplicación del mismo modo que se realizan para Eclipse. WSMO Studio está desarrollado por el grupo Ontotex Lab. Se trata de un "entorno de servicios integrado" (*Integrated Service Environment*). Dicho entorno pretende dar soporte y elaborar la perspectiva abierta por WSMF y WSMO [11], haciendo estas tecnologías transparentes y de fácil uso para el usuario. Esto último se consigue gracias a la interfaz gráfica proporcionada por Eclipse.

La filosofía de WSMO Studio es la de considerar que es importante que la funcionalidad relevante a los dominios de los Servicios Web Semánticos esté presente de una forma que maximice el valor que proporciona.

De entre todas las funcionalidades de WSMO Studio, la que más nos interesa es la de poder manejar documentos BPMO. Esto se consigue gracias al plug-in BPMO Modeller, que se incluye en WSMO Studio. Este plug-in permite la creación y edición de procesos de negocios usando todos los conceptos disponibles de una manera totalmente gráfica y sin necesidad de tener amplios conocimientos acerca de WSMO, WSDL o WSML. BPMO Editor consigue ahorrar de este modo la compleja tarea, propensa a errores, de tener que escribir directamente las descripciones de los procesos de negocio.

3 OWL-S Modeller

OWL-S Modeller es el nombre que se le ha dado a la herramienta objeto de este artículo. Este nombre proviene de la aplicación análoga de WSMO Studio, BPMO Modeller, cuyo objetivo es similar al de este estudio, pero empleando la ontología BPMO en lugar de OWL-S.

El objetivo principal de OWL-S Modeller es el de realizar una herramienta gráfica que permita generar descripciones de Servicios Web Semánticos en OWL-S a partir de un grafo de procesos. Dicha herramienta toma una ontología y permite al usuario poder generar una descripción de un Servicio Web mediante la composición de procesos que usan conceptos de dicha ontología. Una vez se diseña el grafo con forma de árbol, se crea de forma totalmente automática una descripción de un Servicio Web basada en los conceptos obtenidos de la ontología proporcionada y tomando como modelo de procesos el grafo diseñado.

Esta forma de generar descripciones proporciona al usuario una visión general que difícilmente se conseguiría con un interfaz textual, lo que permite que personas sin conocimientos acerca de los lenguajes en los que se describen los Servicios Web Semánticos puedan crear descripciones para compartir sus recursos o aplicaciones. De este modo se consigue acercar OWL-S y la Web Semántica a los usuarios con conocimientos más básicos.

Con OWL-S Modeller se pretende acercar el uso de las tecnologías de la Web Semántica a todos los usuarios, de modo que sea posible la generación de descripciones de Servicios Web Semánticos sin los conocimientos sobre los lenguajes para su implementación. De esta forma el usuario sólo tiene que preocuparse de la semántica de los servicios, sin necesidad de dedicar esfuerzos a la sintaxis o

estructuración del código. Esto se consigue gracias al modelado de descripciones usando árboles de procesos en los que hay que identificar las entradas y las salidas de dichos procesos con conceptos de la ontología deseada.

La aplicación está compuesta por varias vistas, tal y como se puede observar en la Figura 1.

3.1 Navegador del Proyecto

El área del navegador del proyecto funciona exactamente igual que como lo hace en Eclipse y WSMO Studio. Es decir, se muestran los proyectos existentes en el espacio de trabajo clasificados en forma de árbol de carpetas. Dentro de cada proyecto se anidan los documentos pertenecientes a dicho proyecto, tanto los trabajados como los auxiliares (documento layout, por ejemplo). Estos proyectos y documentos pueden ser manipulados del mismo modo que se pueden manipular en el explorador de Windows, permitiendo copiarlos, renombrarlos, moverlos, etc. Desde éste área pulsando el botón derecho aparecerá un menú contextual que ofrecerá múltiples posibilidades, entre ellas las de crear un proyecto o documento OWL-S nuevo.

3.2 Paleta de Procesos

La paleta es un elemento esencial de la creación de diagramas, en ella se encuentran contenidos todos los gráficos dibujables (nodos y conectores). En la Figura 1 se muestra la paleta, de donde se distinguen cuatro tipos de componentes:

- *Palette*. En este grupo se encuentran las acciones de la paleta incluidas por defecto. Es decir, la de seleccionar objetos (*select*) y la de seleccionar objetos en grupo (*marquee*).
- *General*. En este grupo se ha incluido sólo el proceso atómico, que sería posteriormente el elemento que formará los nodos hoja del árbol de procesos. Para hacer uso de los procesos atómicos sería necesario seleccionarlos en la paleta y luego seleccionar el lugar en el diagrama donde se quieran ubicar.
- *Connections*. Se encuentra una única conexión dirigida que irá de padre a hijo. Para usarla se selecciona en la paleta, luego se selecciona el padre para finalmente seleccionar al hijo, lo que automáticamente creará la conexión. El posicionamiento de la conexión se realiza automáticamente y se actualiza cada vez que se decida mover alguno de los nodos que conecta.
- *Control Constructs*. En este grupo se incluyen todas las estructuras de control proporcionadas por OWL-S. Los nodos de este tipo que contengan estructuras condicionales vendrán con una condición seleccionada por defecto, pero resulta recomendable verificar que sea la deseada.

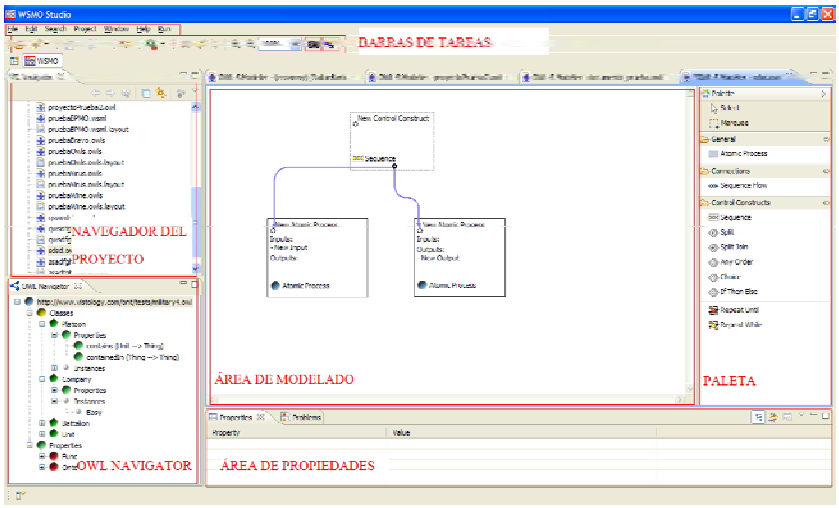

Figura 1. Interfaz de usuario

3.3 Área de Modelado

En el área de modelado (Figura 1) será donde se dibujen todos los modelos de árboles de procesos. Inicialmente será una pantalla en blanco (que será considerada el fondo del diagrama), que posteriormente se irá conteniendo nodos y conexiones procedentes de la paleta. Los objetos que se dibujen tendrán toda libertad de "movimiento" y de "tamaño" en el sentido de que el usuario puede ubicarlos y reubicarlos constantemente a la vez que variar los tamaños de los mismos siempre y cuando no se sobrepongan. El área de modelado cuenta con menús contextuales que dependen del objeto sobre el que se ejecute la pulsación del botón derecho del ratón. Por ejemplo, el menú contextual del fondo del diagrama ofrece la posibilidad de exportar la imagen actual, actualizar el navegador OWL y reubicar todos los elementos.

Como hemos indicado, los nodos se añaden al diagrama haciendo uso de la paleta (seleccionando un objeto y luego haciendo click en el fondo del diagrama donde queramos que se sitúe). Una vez un nodo haya sido dibujado, se pueden realizar distintas operaciones sobre él (algunas dependen del tipo de nodo). Todos los nodos dibujados pueden moverse libremente arrastrándolos y permiten cambiar de tamaño arrastrando sus bordes.

Cada nodo posee un nombre situado en la parte superior (en el nodo de ejemplo corresponde a "*New Control Construct*"). Dicho nombre se puede cambiar bien haciendo doble click sobre el nodo o abriendo el menú contextual del nodo y pulsando "*rename*". Además del nombre, cada nodo dispone de un tipo de nodo y un icono correspondiente que se encuentran en la parte inferior del nodo (en el nodo de ejemplo este tipo corresponde a "*Sequence*"). Por otro lado, los nodos contienen nodos de eventos internos que permiten arcos de entrada y de salida. Se diferencian dos tipos de nodos de evento: nodo final y nodo inicial. El nodo final es el nodo que recibe los arcos procedentes de otros nodos mientras que el nodo inicial es de donde

parten dichos arcos. Los procesos atómicos sólo tienen nodo final, dado que de ellos no saldrá ningún arco. A la hora de realizar las conexiones, concretamente al seleccionar un nodo, el arco apuntará automáticamente al nodo evento que debe apuntar, por lo que no hay que seleccionar el nodo evento.

.

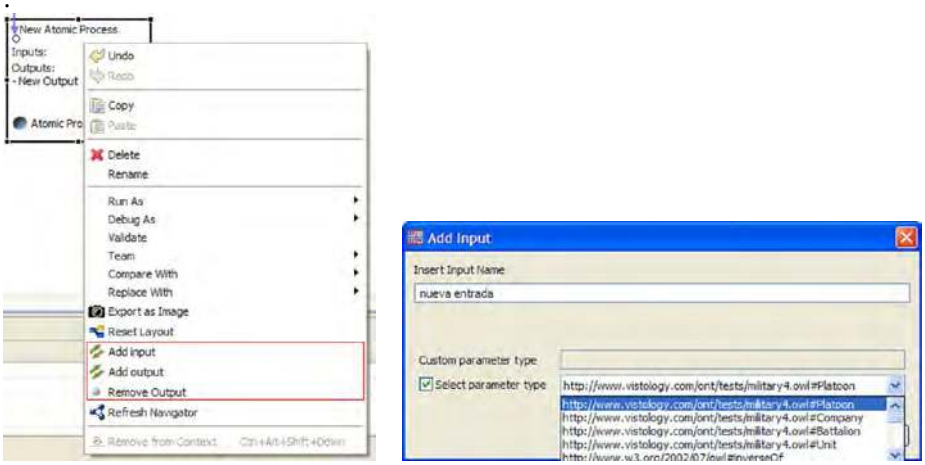

Figura 2. (Izquierda) Menú contextual de un nodo (proceso) atómico. (Derecha) Creación de una entrada a un proceso atómico usando conceptos de la ontología usada para la anotación del servicio Web.

Los procesos atómicos son un tipo especial de nodo, dado que son los que componen el árbol de procesos que finalmente generará la descripción del Servicio Web. Para que esta acción pueda llevarse a cabo se tienen que definir una serie de entradas y salidas de los procesos que serán conceptos de la ontología. En la Figura 2 (Izquierda) se muestra el menú contextual para un proceso atómico con una salida. En dicho menú se ven dos opciones comunes para todos los procesos atómicos: "*Add input*" y "*Add output*". Estas acciones permiten añadir las entradas y las salidas necesarias para poder generar la descripción deseada.

La Figura 2 (Derecha) muestra el caso específico de la adición de una entrada. Se observa el diálogo que aparece donde se puede introducir el nombre de dicha entrada y su tipo. El tipo de la entrada viene dado por una lista desplegable que contiene todos los conceptos y propiedades incluidos en la ontología base de forma que el usuario no debe copiar la URL de los conceptos de otro sitio, aportando comodidad a la utilidad. En el ejemplo se ha decidido nombrar la entrada como "*nueva entrada*" y seleccionar el tipo "*Platoon*". El caso para las salidas es totalmente análogo al de las entradas, por lo que queda explicado.

3.4 OWL Navigator

La vista del navegador se muestra por defecto en la parte inferior izquierda del interfaz de la aplicación. El objetivo de esta parte de la vista de la aplicación es el de permitir desglosar las ontologías base de las descripciones para poder observar jerárquicamente los contenidos de dichas ontologías OWL. Por lo tanto, el contenido

del diagrama dependerá directamente de la ontología base del editor actual. Sin embargo, cuando se cambia de un editor asociado a un documento a otro, no se actualiza automáticamente el contenido del navegador. Para ello se ha aportado la funcionalidad de actualizar el navegador en base al editor activo actual mediante el menú contextual del diagrama (Figura 3), que actualizará inmediatamente el contenido del navegador para mostrar la ontología OWL atribuida al editor activo. El navegador es un explorador de los elementos de la ontología OWL empleada. Estos elementos se clasifican en clases y propiedades. Dentro de cada clase se muestran las propiedades en las que intervienen y las instancias existentes. Por ejemplo, en la Figura 3 se muestra la clase *Platoon* que participa en la propiedad *contains* que tiene como dominio la clase *Unit* y como rango la clase *Thing*. Además se observa en la Figura 3 la clase *Company*, que tiene la instancia *Easy*. Las propiedades existentes se clasifican a su vez en tipos: propiedades funcionales, propiedades de ontología y propiedades de tipos de datos.

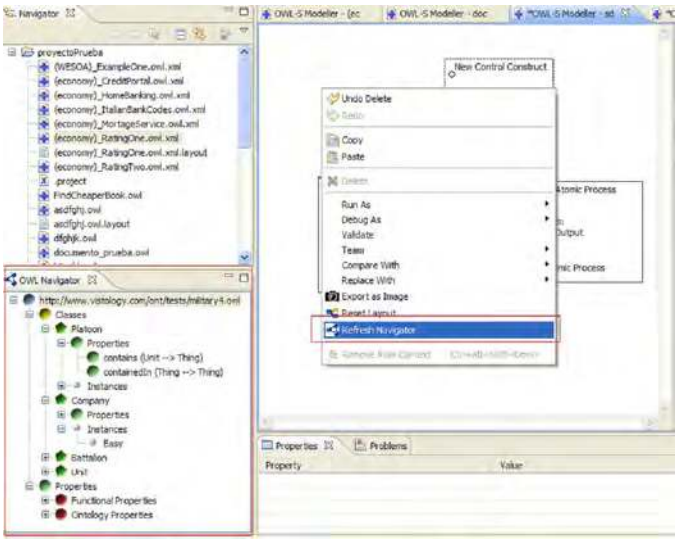

Figura 3. Navegación por la ontología seleccionada

3.5 Opciones del Menú

El proceso de exportación de un diagrama gráfico al modelo de procesos en un archivo OWL-S se lleva a cabo en el momento en el que se desea guardar dicho diagrama. Por lo tanto, para exportar el diagrama actual tan sólo se debe ir a "*File*", "*Save*" o bien pulsar "*CTRL+S*". En este momento se iniciará la validación de modelo y, en caso de cumplir con todas las restricciones, se procederá a la conversión. En caso contrario aparecerá un mensaje indicando los errores encontrados. Si no se cumplen los requisitos, dicho diagrama no se podrá guardar y por tanto no se podrá cargar en un futuro para continuar diseñándolo. En caso de decidir guardarlo aún

existiendo errores, con bastante probabilidad dicho diagrama no se guardará correctamente y no sea posible volver a cargarlo. Por este motivo, se recomienda guardar sólo cuando el modelo esté correctamente construido y validado.

4 Conclusiones

La propuesta principal de integración de aplicaciones es reemplazar la diversidad de interfaces independientes por una aproximación ordenada utilizando un núcleo central, reduciendo costos de mantenimiento. Los diferentes modelos de integración de aplicaciones permiten compartir datos y procesos de negocios a través de diferentes sistemas, en forma independiente del sistema operativo, el lenguaje de programación o el soporte de datos.

Las tecnologías para la Web están avanzando constantemente con nuevas iniciativas. De entre estas iniciativas, la Web Semántica pretende dotar de semántica a todos los datos contenidos en la Web, permitiendo de este modo proporcionar los mecanismos necesarios a las máquinas para que comprendan la información. Además, se pretende convertir la Web en una red de datos en lugar de la red de aplicaciones que controlan los datos que es actualmente. Con ello se conseguiría que existiese una interoperabilidad estandarizada entre aplicaciones Web.

La Web Semántica va poco a poco convirtiéndose en una realidad, prueba de ello son aplicaciones ya implantadas como RSS y FOAF. Pronto será posible acceder a los recursos de la Web a través de sus contenidos en lugar de hacerlo mediante palabras claves, tal y como se realiza actualmente. Para que esta situación sea posible, es necesario el desarrollo de lenguajes de ontologías que permitan modelar todos los conceptos necesarios para la Web Semántica. Este trabajo se ha centrado en OWL y WSMO, que aportan niveles de expresividad diferentes pero, manteniendo el objetivo común de hacer posible la Web Semántica.

La integración es una tarea cada vez mas frecuente en las empresas, y que además de tener un alto coste suele consumir muchos recursos. Para tales empresas la necesidad de integración es, en gran parte, consecuencia de la evolución natural de su negocio.

OWL-S Modeller se presenta como una herramienta diseñada precisamente para realizar esta tarea. A pesar de la existencia de múltiples proyectos relacionados, las principales ventajas de esta herramienta es que aúna de forma integrada tanto el diseño de anotaciones de servicios Web usando WSMO y OWL-S. Esta extensión de WSMO Studio permite por tanto que desarrolladores acostumbrados al mismo puedan realizar anotaciones mediante OWL-S, y para aquellos desarrolladores no acostumbrados a WSMO Studio se ofrece una herramienta basada en Eclipse, facilitando su aprendizaje.

Agradecimientos. Trabajo financiado por el proyecto ICARIA TIN2008-04844 (Ministerio de Ciencia e Innovación).

Referencias

1. Berners-Lee, Tim: A roadmap to the Semantic Web. Informe técnico, septiembre 1998. http://www.w3.org/DesignIssues/Semantic.html.

2. Jos de Bruijn, Rubén Lara, Sinuhé Arroyo, Juan Miguel Gomez, Sung-Kook Han and Dieter Fensel: A Unified Semantic Web Services Architecture based on WSMF and UPML, In International Journal on Web Engineering Technology, special issue on Semantic Web, 2005.

3. David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, Katia Sycara, "Bringing Semantics to Web Services: The OWL-S Approach", Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), July 6-9, 2004, San Diego, California, USA.

4. Dimitrov, M., Simov, A., Momtchev, V., and Konstantinov, M. 2007. WSMO Studio --- A Semantic Web Services Modelling Environment for WSMO. In Proceedings of the 4th European Conference on the Semantic Web: Research and Applications (Innsbruck, Austria, June 03 - 07, 2007). E. Franconi, M. Kifer, and W. May, Eds. Lecture Notes In Computer Science, vol. 4519. Springer-Verlag, Berlin, Heidelberg, 749-758.

5. Scicluna, J., C.Abela y M. Montebello: Visual Modelling of OWL-S Services. En IADIS International Conference WWW/Internet, octubre 2004. Madrid Spain.

6. Srinivasan, Naveen, Massimo Paolucci y Katia Sycara: CODE: A Development Environment for OWL-S Web Services. Informe técnico, 2004. http://iswc2004.

semanticweb.org/demos/10/paper.pdf.

7. Srinivasan, N., Paolucci, M., and Sycara, K. 2006. Semantic Web Service Discovery in the OWL-S IDE. In Proceedings of the 39th Annual Hawaii international Conference on System Sciences - Volume 06 (January 04 - 07, 2006). HICSS. IEEE Computer Society, Washington, DC, 109.2.

8. T. Tudorache, N. F. Noy, S. W. Tu, M. A. Musen. Supporting collaborative ontology development in Protégé. Seventh International Semantic Web Conference, Karlsruhe, Germany, Springer. Published in 2008

9. D. Elenius. The OWL-S Editor – A Domain-Specific Extension to Protégé. 8th International Protégé Conference, July, 2005.

10. Jaeger, Michael C., Gregor Rojec-Goldmann, Christoph Liebetruth, Gero M uhl y Kurt Geihs: Ranked Matching for Service Descriptions Using OWL-S. KiVS, páginas 91-102, 2005.

11. Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel: Web Service Modeling Ontology, Applied Ontology, 1(1): 77 - 106, 2005.

A Model Transformation-based Technique for Flexible XML Data Source Integration

María Pérez, Ismael Sanz, María José Aramburu, and Rafael Berlanga*

Universitat Jaume I de Castelló, Spain
{mcatalan,isanz,aramburu,berlanga}@uji.es

Abstract. Current research in domains such as Bioinformatics depends heavily on the integration and processing of information coming from diverse sources, which are typically highly complex and heterogeneous, and require approximate queries based on a ranking method. Processing these sources requires sophisticated query processing techniques, which are based on indexing and optimization techniques which are highly specific to the chosen ranking method. In this work we present a technique for the selecting appropriate ranking methods for complex sources, which is useful to drive the development of targeted information systems, using model transformations. We describe the general approach, its application context, and showcase it using examples of an $i*$ profile and ATL transformations.

1 Introduction

A number of important applications are based around approximate queries that return ranked results. The classical examples arise in textual Information Retrieval and multimedia databases. Many different approaches have been proposed in both fields, usually build around a given notion of "similarity" which is used for ranking results. Traditionally, systems in these fields have been build and optimized around the chosen notion of similarity, developing ad-hoc query processing techniques and indexing structures; the selection of a different ranking method requires building and testing a new system.

The emergence of new applications fields which also require the use of approximate queries, and the fact that many of these systems require integrating heterogeneous data sources, has caused serious problems in system development. As a case study, we fill focus on the bioinformatics research community, which produces a vast amount of information in the form of large databases of complex data [4], typically available in XML. These databases are highly heterogeneous, contain complex domain-specific data (such as protein sequences or metabolic networks), and are usually very specific to a particular subfield. Typical workflow in the field require approximate queries, and are built in the context of information systems that must store, integrate and process these databases [5].

As a consequence, several techniques and standards developed in the context of Enterprise Information Integration systems have been used to build research-oriented applications, including Data Warehouses, mediator-oriented architectures and peer data management systems [9]. Nevertheless, the development of such systems is mostly performed in an ad-hoc way, with reusable frameworks providing only minimal scaffolding [8]. This makes it very costly to create even relatively simple, small-scale integrated systems.

A major reason for this state of affairs are the distinct requirements of Bioinformatics-oriented systems, when compared with standard enterprise applications. These are caused by the qualitatively different nature of the underlying data that must be processed in the Bioinformatics case: the high complexity of the data and the need to deal with very large

* Supported by grant TIN2008-01825 of the Spanish Ministry of Science and Innovation.

databases of such data requires the use of non-conventional techniques. Usually these techniques provide approximate results (e.g. sequence matching), which poses the problems mentioned above.

In this paper we report our experience developing an approach to integrate approximate data in the bioinformatics domain. Our main goal is to clarify which notions of similarity (in the form of ranking functions or similarity measures; we will use both term interchangeably) are best suited to a given application, in order to focus the development. We will use a disciplined approach based on model transformations that begins with a formal early requirements model based on $i*$, and applies model transformations guided by appropriate semantic domain models to obtain a set of candidate ranking functions, which are the assessed to complete a precise specification of the necessary ranking methods for the target application. Our aim is to show the benefits that arise from applying such an approach instead of ad-hoc techniques.

The rest of this paper is organized as follows. The next section explain the context of our case study. Section outlines our approach. Sections 4 and 5 explain respectively the initial requirements gathering and the subsequent model transformations. Section 6 discusses the lessons learned, and the paper concludes with some directions of further research.

2 Context: Integrating Approximate Data

Some of the most frequent tasks that must be addressed when building Bioinformatics-oriented information systems require producing or processing approximate data, usually in the form of ranked results to a query [13]. Examples include approximate protein sequence matching or full-text bibliographic searches.

When acquired from different independent sources, the integration of such data requires the specification of *ranking functions*, which usually capture important requirements of the problem domain. For instance, when gauging the relevance of a given protein for a particular problem, a biologist may rely solely on sequence searches on a set of databases, or may also want to include the result of a bibliographic search as a factor with a given weight. Both cases require different ranking functions, which may correspond with different use cases or, in a more specialized case, may even be specified at run time within certain constraints.

To solve such cases, multi-similarity systems [3] that support multiple notions of similarity simultaneously have been developed. These systems support the creation of aggregation functions for ranked approximate data, based on the combination of lower-granularity functions that ultimately represent the results of queries to individual data sources. This has the explicit goal of accounting for the diverse user requirements that arise when dealing with heterogeneous data that requires complex processing [12]. Since, in principle, many different combinations of functions are possible, assessment techniques [10] are useful to help choose appropriate out of a pool of candidates.

While multi-similarity systems fit the necessities of bioinformatics systems, they have one major drawback: the quality of the retrieved results depends largely on the selected (combination of) aggregation functions. Unfortunately, these are very difficult to determine, as they are highly dependent on the domain, the particular set of databases used, the desired information and the particular user. The problem of matching the most appropriate function to each specific context and user has not been addressed in depth.

In the next section we introduce a disciplined approach to assist the development of information systems that involve approximate querying using a multi-similarity approach over a set of heterogeneous XML collections, as is typical in the bioinformatics context.

3 Building Multi-Similarity Systems

The main goal of our approach is to allow the user to describe her information requirements without specifying in which XML collection(s) is the data she is looking for and how this data is encoded or represented. In order to achieve this main goal, we propose an approach to assist the design of multi-similarity-oriented ranking functions over a heterogeneous XML document set. The overall approach is shown in Figure 1.

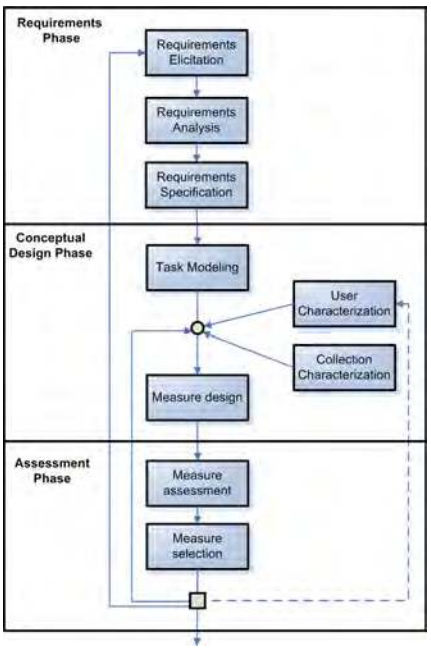

Fig. 1. Overview of the proposed approach.

It consists of three main phases: the *Early Requirements*, the *Conceptual Design* and the *Assessment* phases.

1. **Early Requirements Phase.** The purpose of this phase is to gather the user information requirements and to specify them in a formal way. The user determines which information she needs and, moreover, she gives extra information about her experience and knowledge describing the steps she would make if she had to do the ranking manually. The description of these steps is also specified in the formal definition in order to give extra information in the subsequent phases. All this information is completely independent of the characteristics of the particular collections to be used. The formal definition of the early user requirements is the input of the semi-automatic process of designing the most suitable similarity measures.
2. **Conceptual Design Phase.** The next goal is to conceptually model all the information that is needed to enable the selection and composition of the most appropriate set of measures. This includes a specification of the tasks the system needs to perform to achieve the goals of the user, and the characterization of the user along with the targeted document collection. This specification of the tasks is based on the early user requirements, taking into account the characteristics of the collections. The formal task specification determines a set of ranking functions, which we will term *candidate measures*.

3. **Assessment Phase.** The last phase we will use a semi-automatic method, outlined in [10], that evaluates empirically the quality of candidate measures. According to the results, the user may generate some feedback: she may redefine her requirements definition or she may change some parameters of the similarity measures or even modify the composition of some of them. After some iterations, this phase returns a set with the measures that are the most appropriate for the user requirements, given their particular characteristics and the targeted document collections.
   The similarity measures that are produced by this method capture the user requirements with respect to ranked data, and can be further studied to find out which are the necessary indexing and optimization mechanisms required to integrate them into the final system.

3.1 Model Transformations

[10] presents an approach where all the processes of characterization, design and combination of measures, the evaluation of the candidate measures, and so on, are performed manually by the system designer, and consequently the quality of the results depends largely on the system designer's experience and knowledge.

An important requirement of the approach is that it must provide guidance for the system designer and be a semi-automatic process. All the required information about the user requirements and about her knowledge and experience has to be gathered in the Requirements phase in order not to involve the system designer in the design and selection of the most suitable similarity measures. Once the user requirements are elicitated and formally specified, the feedback of the system designer is only required at the end of the Assessment phase in order to validate the set of similarity measures selected by the system. In this way, the execution of the Conceptual Design phase and the Assessment phase are automatic.

The requirement of being a semi-automatic process is achieved by using a set of models and a set of ATL transformations. Each phase contains a conceptually rich set of models, which are derived from the requirements specification model and which contain information specific to the functionality of each one of the phases. Each model is created by the transformation of model produced by the previous phase. The transformations are implemented in ATL [2] and they consist on a set of matching rules that specify the elements of the target model to be created according to the elements of the source model.

ATL is a model transformation language that allows the user to design transformations that produce a set of target models from a set of source models. It is a hybrid language since it provides declarative and imperative constructs. Its functionality is basically based on matching rules that enable to establish matching relations between elements in the metamodels (or in profiles). Therefore, from a model M_a, which conforms to MM_a, it is possible to create M_b, which conforms to MM_b, executing appropriate rules.

Figure 2 shows a detailed overview of the first two phases and their sub-phases (depicted as rectangles), the models (depicted as rounded rectangles), the related profiles (depicted as ellipses on the right), the domain ontology (depicted as a square on the left) and the ATL transformations (depicted as arrows). The dotted arrows represent the relationships of conformance between the models and the profiles.

4 Information Requirements for Heterogeneous Data Sources

Information requirements in heterogeneous data sources are different from the typical software systems requirements. In this context, the integration solutions have to deal with the lack of well-defined schemas of the sources and therefore, with the user's ignorance in the location of the desired data and its format.

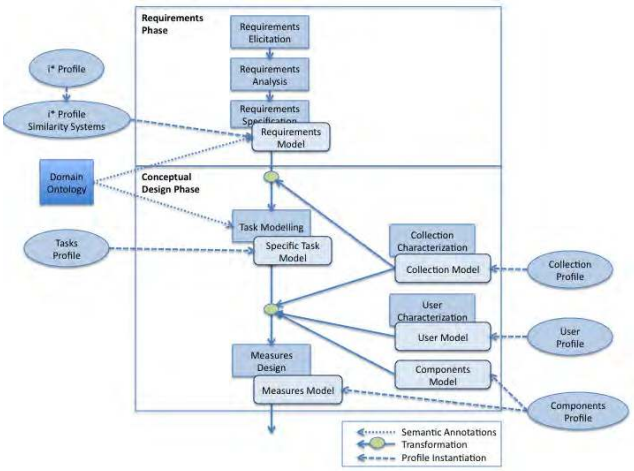

Fig. 2. Detailed overview of the Early Requirement Phase and Conceptual Design Phase.

We aim to solve this problem by gathering the requirements without taking into account the characteristics of the collections or the characteristics of the system. The user should not have to determine in which document or collection is the information she is asking for or how it is exactly represented. In our proposal, early requirements are gathered and formally specified in the Requirements phase. To obtain a formal specification of the users requirements, we follow a requirements engineering model similar to the one proposed in [11].

Early requirements elicitation and analysis. First of all, the system designer interviews the user in order to identify her information needs, why she has those needs, try to detect other hidden requirements the user is not aware of, and so on. Moreover, the system designer also tries to know how the user would fulfill her requirements manually. The system designer has to capture a much information as possible in order to capture in detail the requirements in the formal specification. All the gathered information is analyzed and annotated semantically with a domain ontology, e. g. BioPAX [1]. Usually, the elicitated requirements represent instances of recurring problems in a specific domain. Therefore, elicitated requirements are usually substituted by a more generic requirement that represents the recurring problem and involves a set of specific instances of the generic requirement that possibly provide excessive details.

Requirements specification. Once the requirements are elicitated and analyzed, the next step is to describe them in a formal way. The formal specification of the requirements must fulfill two important conditions: (i) the specification must be done without taking into account the characteristics of the target collections and (ii) the specification must describe *what* the user wants to retrieve and *why* and *how* the user would (manually) carry out the search. The achievement of all of these conditions is crucial, since they are necessary to elaborate automatically the specific models required to the selection of the more appropriate ranking functions.

We have chosen $i*$ language [14] to formalize the requirements. The $i*$ framework is a goal-oriented and agent-oriented language defined by Eric Yu and it was designed to model early-requirements without taking into account the characteristics of the system. The main reasons for choosing $i*$ are: (i) it allows the system designer to specify the requirements independently of the characteristics of the system and therefore, also of the characteristics of the collections, (ii) it offers a formal representation of the goals and their behaviors by using a formal decomposition structure , (iii) it provides sup-

port to define actors and dependencies between actors, (iv) it allows to describe "why" and "what" the user wants and (v) it provides two levels of abstraction: Strategic Dependency (SD) model, that describes the dependency relationships among actors in an organizational context, and the Strategic Rationale (SR) model, that describes actor interests and concerns and how they might be addressed. Due to lack of space we refer the reader to [14] for further details on $i*$.

Since $i*$ gives a high degree of freedom for the construction of the models as remarked in [6], in some cases an adaptation of the $i*$ framework to a specific domain is required. For example, [7] defines a specific UML profile for the design of Data Warehouses based on the $i*$ framework. In the same vein, due to the particularities of similarity-based approximate querying, we adapted the $i*$ framework by relating the elements of the $i*$ models with specific domain concepts from similarity-based systems domain. To do so, we used the profiling mechanism of the *Unified Modeling Language* (UML) to extend it in the following ways: (i) a profile for $i*$, similarly as described in [7], in order to integrate it within our MDA framework; and (ii) a profile which adapts $i*$ to the similarity-based systems domain. Figure 3 shows the $i*$ profile and Figure 4 shows the specific profile for similarity-based systems in the biological domain. In the specific profile, the elements specific of the similarity-based systems are described as specialization of the previously defined $i*$ elements.

Requirements validation. The validation of the requirements is done by the user once the process has been executed and the most appropriate similarity measures have been provided to him. At this moment, the user analyzed if the results are the ones he expected and in case they are not, he might realize inaccuracies in her specified requirements, and could modify them. The execution of the process is iterative and it finishes when the user obtains the expected results.

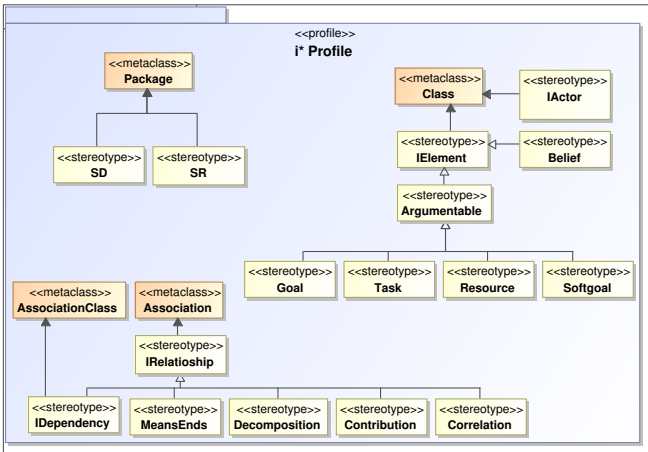

Fig. 3. Overview of the $i*$ profile.

5 Model-driven Data Source Integration

The Requirements model is the input of the semi-automatic process of designing the similarity measures. A set of ATL transformations transforms the collection-independent requirements specification into a collection-dependent set of similarity measures. In each phase,

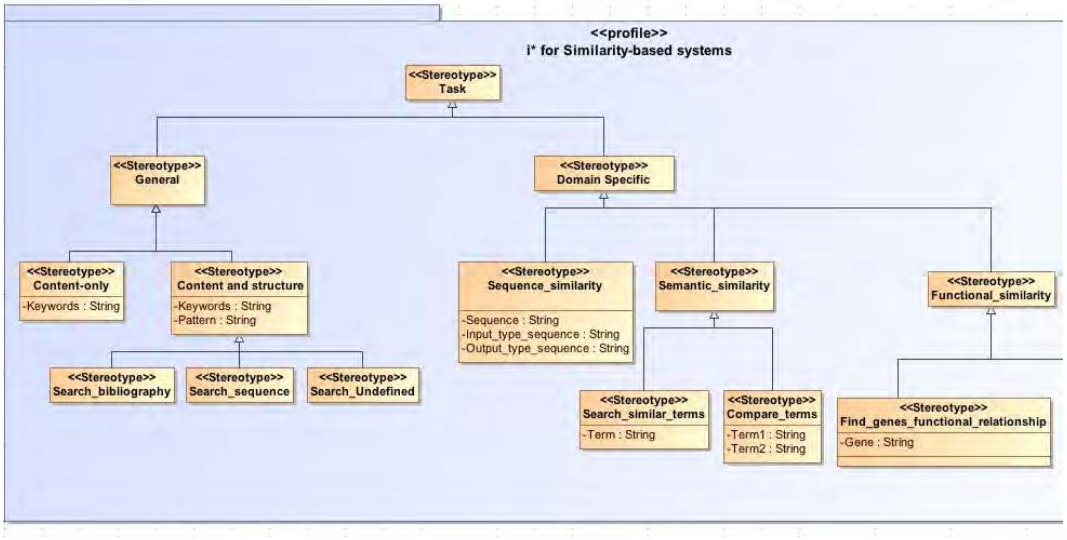

Fig. 4. Overview of the specific i^* profile for similarity-based systems in the bioinformatics domain.

there are a set of models that are derived from the Requirements model. Each model represents a different abstraction level. Here we describe briefly the main purpose of each one of the models:

1. **Requirements model.** It describes the information needs the users have on the target collections. It is independent of the characteristics of the collections and the characteristics of the similarity system. It is formalized using the $i*$ notation.
2. **Specific Task model.** It describes the tasks the similarity system has to perform to fulfill the user requirements. It is created by the transformation of the Requirements Model. In this transformation, the characteristics of the collection are taken into account in order to know which kind of tasks are required, i.e. a structural search or a text search.
3. **Measures model.** It is the result of transforming the Specific Task model. This model contains the specification of the similarity measures that are the candidates to be the most appropriate in that context, with the specific user requirements gathered previously and these targeted collections. The candidate similarity measures must integrate different components that provide the functionality necessary to fulfill the user requirements. Each component provides a different functionality by implementing a different similarity function. The available components are specified in the Measure Components model. The ATL transformation creates a first set of candidate measures, which is going to be refined by the iterative process of the Assessment phase. At the end of the iterative process, when the results are the ones expected by the user, this model will contain the most appropriate similarity measures for those users requirements, that particular user and those targeted collections.

Moreover, there are some additional models that must be created from scratch, that is, they are not derived from any other.

1. **Measure components model.** It contains the specification of all the available similarity measure components. Components are defined based on the *Components Metamodel* described in [12]. The main principle of this model is that of composition: measures are either represented by an elementary *measure component*, which is an implementation

of a similarity function at a given granularity level, or as a composition of other measure components. These components are combined in the Measures model by using basic combinators such as a weighted sum. In addition, every component can be parametrized; for instance, a component that computes similarity at the textual level may allow the user to choose whether common words ("stop words") must be considered. Using this framework, a large number of functions with different requirements can be defined.

2. **User model.** It contains information about the last queries executed by the user. This information may help in the selection or parameterization of the measures.
3. **Collection model.** It contains a summary of the characteristics of the targeted collections. This summary characterized the level and the type of heterogeneity that each collection has. It is used to select and describe the specific tasks in the Specific Task model and to design the similarity measures.

These models make possible the integration of heterogeneous collections, allowing the users to describe their requirements without determining where and how the data is located.

As an example, we show one of the rules that form the ATL transformation defined between the Requirements profile and the Specific Task profile. This rule matches any type of *Content and Structure* (CAS) requirement into two specific tasks: (i) a *Structural search* task and (ii) a *Text search* task with their corresponding parameters.

```
rule CAS{
  from s:UML2!Class (s.hasDeterminedStereotype('CAS'))
  to t : MMb!Class(name<−s.name),
     d : MMb!Class(name<−s.name)
  do {
      thisModule.Global_UmlModel.packagedElement <−
            thisModule.Global_UmlModel.packagedElement −>including(t);
      t.applyStereotype(UML2!Stereotype.allInstances()−>
            select(st | st.name = 'Structural_search'). first ());
      t.setValue(t.getAppliedStereotype('STM::Structural_search'),
                                          'Pattern', s.getTagValue('CAS','Pattern'));

      thisModule.Global_UmlModel.packagedElement <−
            thisModule.Global_UmlModel.packagedElement −>including(d);
      d.applyStereotype(UML2!Stereotype.allInstances()−>
            select(st | st.name = 'Text'). first ());
      d.setValue(d.getAppliedStereotype('STM::Text'), 'Keywords',
                                          s.getTagValue('CAS','Keywords'));
   }
}
```

where *hasDeterminedStereotype(Stereotype)* and*getTagValue(Stereotype, Attribute)* are helper functions. The former checks if the each element of the input model has a determined stereotype and the latter obtains the value of an attribute specific of its stereotype.

A prototype of the tools described in this paper has been implemented in Eclipse. Figure 5 shows a screenshot displaying the structure of some candidate ranking functions obtained, which would go on to be assessed using the techniques outlined in [10] in order to obtain the final functions to be implemented in our example case study.

6 Discussion and Future work

We have outlined a method to help development of systems which are strongly driven by the implementation of ranking functions. Mistakes in the specification can be costly, and

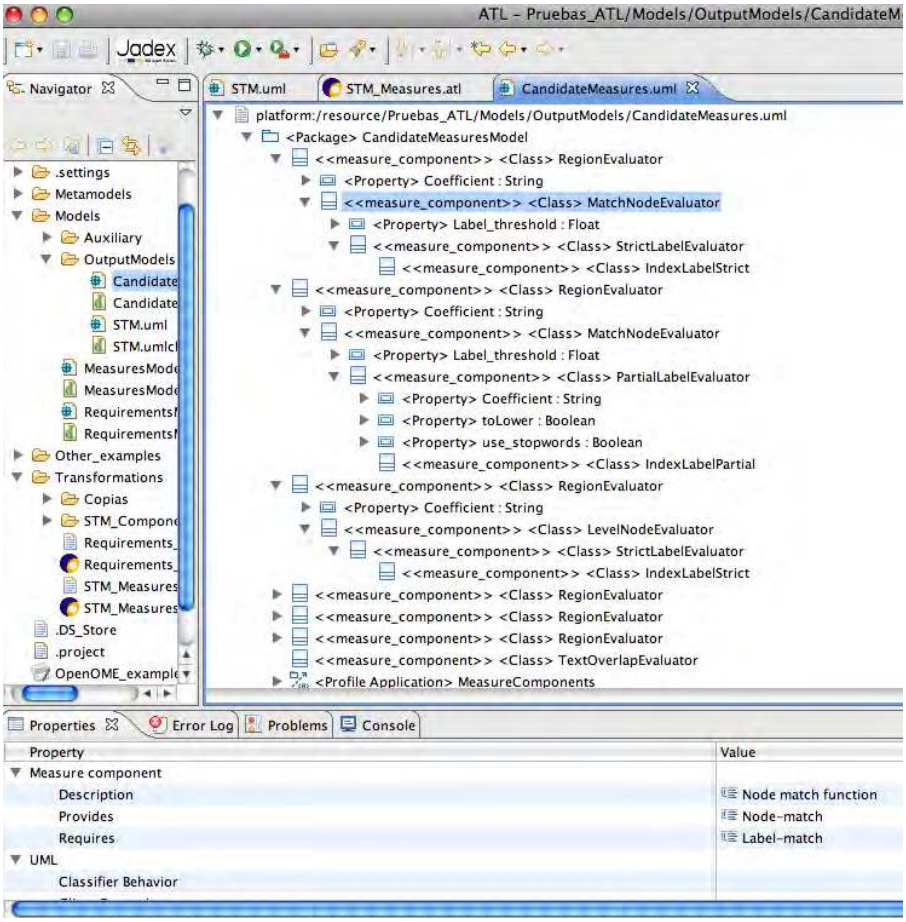

Fig. 5. A screenshot of Eclipse showing the obtained candidate measures after applying the transformations

our main goal is to ensure that the selected functions are as close as possible to the information requirements of the users, in order to avoid the useless implementation of the wrong processing and indexing components.

We consider our initial results encouraging. The models and transformations that we have developed and that we describe in this paper form a good basis for further developments. Our immediate goal is to continue experimenting with realistic examples in order to further refine the models and transformations used in our approach. Eventually, we plan to incorporate the techniques outlined in this work into a more comprehensive methodology which can be used for guiding all aspects of bioinformatics-oriented information systems development, allowing for better maintainability and reusability of such systems and building specific tools to support the developers.

References

1. Biopax. http://www.biopax.org/.
2. ATLAS team: ATLAS Transformation Language (ATL) Home page, 2006.
3. S. Adalı, P. Bonatti, M. L. Sapino, and V. S. Subrahmanian. A multi-similarity algebra. In *SIGMOD '98*, pages 402–413, New York, NY, USA, 1998. ACM Press.
4. Michael Y. Galperin and Guy R. Cochrane. Nucleic Acids Research annual Database Issue and the NAR online Molecular Biology Database Collection in 2009. *Nucleic Acids Res.*, 37:D1–4, Jan. 2009.
5. Carole Goble and Robert Stevens. State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics*, 41(5):687–693, Oct. 2008.
6. G. Grau, C. Cares, X. Franch, and F.J. Navarrete. A comparative analysis of i* agent-oriented modelling techniques. In *Proceedings of the 18th International Conference on Software Engineering and Knowledge Engineering (SEKE 2006)*, pages 657–663, July 2006.
7. Jose-Norberto Mazón, Jesús Pardillo, and Juan Trujillo. A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In *ER Workshops*, pages 255–264, 2007.
8. Marco Mesiti, Ernesto Jiménez-Ruiz, Ismael Sanz, Rafael Berlanga, Paolo Perlasca, Giorgio Valentini, and David Manset. XML-based approaches for the integration of heterogeneous bio-molecular data. *BMC Bioinformatics*, in press, 2009.
9. Marco Mesiti, Ernesto Jiménez-Ruiz, Ismael Sanz, Rafael Berlanga, Giorgio Valentini, Paolo Perlasca, and David Manset. Data integration issues and opportunities in biological XML data management. In Eric Pardede, editor, *Open and Novel Issues in XML Database Applications: Future Directions and Advanced Technologies*. IGI Global, 2009.
10. María Pérez, Ismael Sanz, and Rafael Berlanga. Measure assessment for heterogeneous XML collections. In *JISBD 2008*, 2008.
11. S. Raghavan, G. Zelesnik, and G. Ford. *Lecture Notes on Requirements Elicitation*. Education Materials. CMU/SEI-94-EM-10. Software Engineering Institute. Carnegie Mellon University, Pittsburgh, PA 15213 (USA), 1994.
12. Ismael Sanz, Marco Mesiti, Giovanna Guerrini, and Rafael Berlanga. Fragment-based approximate retrieval in highly heterogeneous XML collections. *Data & Knowledge Engineering*, 64(1):266–293, January 2008.
13. Robert Stevens, Carole Goble, Patricia Baker, and Andy Brass. A classification of tasks in bioinformatics. *BIOINFORMATICS*, 17:180–188, 2001.
14. Eric Yu. *Modelling Strategic Relationships for Process Reenginering*. PhD thesis, University of Toronto, Canada, 1995.

Integración de Aspectos de Calidad de Datos en Sistemas de Información

Ismael Caballero, Mª Ángeles Moraga, Coral Calero

Universidad de Castilla–La Mancha
Grupo Alarcos - Instituto de Tecnologías y Sistemas de la Información
Pº de la Universidad 4, 13071 Ciudad Real (Spain)
{ MariaAngeles.Moraga,Ismael.Caballero, Coral.Calero}@uclm.es

Abstract. Para desarrollar con éxito una tarea es imprescindible que los usuarios de información puedan disponer de datos con los niveles adecuados de calidad. Para tomar una decisión sobre si usar o no los documentos, necesitan tener una evaluación de esos niveles. Para evaluarlos, es necesario disponer de un modelo de calidad de datos. Como quiera que la evaluación depende del contexto, y unos mismos datos deben estar disponibles para varios contextos, entonces toda la infraestructura de evaluación debe estar disponible para ser accedida desde distintas aplicaciones. Esto requiere que los datos estén adecuadamente descritos para que puedan ser entendidos. Este artículo plantea algunos de los retos de investigación que estamos afrontando, como por ejemplo la posibilidad de aplicar los principios de LinkedData para anotaciones de calidad de datos, o como aplicar los resultados que vayamos obteniendo a los datos almacenandos en bases de datos heredadas, de modo que puedan ser utilizados en entornos distintos para los que fueron plantearon, de forma que puedan seguir siendo utilizados.

Palabras Clave: Calidad de Datos, Dimensiones de Calidad de Datos, Web Semántica, Integración**,** Evaluación de Calidad de Datos, Linked Data.

1. Introducción

Los usuarios necesitan datos para desarrollar tareas asociadas a su trabajo. Estos datos están recogidos o agrupados en documentos, o que pueden ser obtenidos consultando bases de datos. Habitualmente estos datos son reutilizados en diferentes tareas por diferentes usuarios desde distintas aplicaciones. Por ejemplo, una historia clínica electrónica de un paciente puede ser usada (leída) por un médico para diagnosticar una enfermedad, o puede ser usada (actualizada) por un analista de un laboratorio para incluir los resultados de un análisis de sangre, o puede ser usada (leída y actualizada) por un enfermero para consultar el tratamiento que se le debe dar a un enfermo durante su hospitalización y apuntar los datos requeridos en el seguimiento. Cada uno de estos tres actores podría estar usando una aplicación diferente, pero todos ellos están trabajando sobre los datos relativos a un paciente, datos que en este

caso están agrupados en un documento conocido como "historia clínica electrónica de un paciente", y que va evolucionando con la vida del paciente.

Para tener éxito en sus respectivas tareas, estos actores necesitan que los datos contenidos en los documentos tengan niveles adecuados de calidad: precisamente en el ejemplo propuesto, cualquier error derivado de una decisión errónea puede ser crítico para el paciente. Se dice que los datos tienen calidad si sirven para el propósito (tarea) para el que se pretenden usar [1]. Esta definición de calidad de datos basada en *fitness for use* pretende ser más general que la percepción habitual de que los datos son de calidad si no tienen defectos. Esta definición conlleva dos implicaciones importantes: por un lado está la percepción multidimensional del problema (es necesario evaluar la calidad de los datos contenidos en un documento usando distintos criterios, a los que como se verá en la sección 2, se les denominan dimensiones de calidad de datos), y por otro lado se incluye el carácter subjetivo de la calidad de datos (cada uno de los actores anteriores, médicos, analistas de laboratorio, enfermeros, … puede tener una percepción diferente del nivel de calidad, incluso para la misma dimensión de calidad de datos). Como puede entenderse, estas dos implicaciones son más difíciles de gestionar que la percepción de que los datos tengan "*cero defectos*".

De estas dos implicaciones, se puede deducir que estimar cada una de las dimensiones de calidad requeridas no es tarea fácil. Una forma de realizar esta estimación es mediante la definición de medidas capaces de cuantificar la calidad de cada dimensión. Para poder definir dichas medidas es preciso entender el contexto del usuario y lo que puede entender por el hecho de que los datos se ajusten a su uso. En algunas ocasiones y para ciertas dimensiones de calidad de datos, puede ser posible establecer medidas objetivas (y repetibles) sobre los propios datos de manera independiente del contexto, mientas que en otras muchas, es necesario disponer de datos adicionales (normalmente denominados "*metadatos*" [2]) que de alguna forman representan el contexto de los usuarios. De hecho, podría decirse que estos datos adicionales permiten completar el significado del dato en el sentido marcado por la dimensión de calidad elegida. Así por ejemplo, para determinar el grado de *actualidad* (véase sección 2) de un valor en la bolsa es necesario adjuntar al dato correspondiente al valor del título una fecha que indique cuándo ha sido proporcionado; o por ejemplo, para determinar si una opinión en un foro merece ser tenida en cuenta (grado de *reputación*), un usuario debería poderse conocer quién dio esa opinión (seguramente el foro tenga establecido un sistema de clasificación de reputación de usuarios basado en la antigüedad de los usuarios, o en el número de opiniones proporcionada que son aceptadas como útiles por otros usuarios, …).

En cualquier caso se hace imprescindible adjuntar y hacer persistentes dichos datos adicionales junto a los datos utilizados por los usuarios, de modo que las distintas aplicaciones puedan disponer de ellos para establecer sus propios métodos de medición. Es importante resaltar que este dato adicional no tiene por qué ser el valor de la medida en sí, o una anotación directa de calidad como proponen Price y Shanks en [3], sino que generalizando, podemos considerar que es un elemento que permitirá calcular la medida de calidad de datos para una determinada dimensión. La bibliografía recoge algunas formas de mantener estos vínculos, tales como la propuesta de Wang y Madnick para el modelo relacional en [2] o la de Caballero et al en [4] para documentos XML.

Nuestra motivación inicial para esta investigación está basada en el hecho de que recientemente, en un proyecto de I+D relacionado con la gestión de historias clínicas electrónicas, nos planteábamos aplicar la propuesta presentada en [2] a una base de datos XML nativa, para optimizar, basándonos en la calidad de datos de los documentos usados, algunos de los procesos que se estaban ejecutando en un sistema de información relacionado con gestión sanitaria, y que ya estaba en producción. La idea era establecer los mecanismos adecuados para medir la calidad de los datos almacenados en dicha base de datos para proporcionar a los usuarios una estimación del nivel de calidad que tenían los datos, de modo que pudieran decidir si usarlos o no en función del nivel de calidad obtenido. Pero pronto nos dimos cuenta que esto implicaba modificar tanto el modelo de procesos como el modelo de datos del SI, pues teníamos que almacenar los datos adicionales y agregar las consultas correspondientes. Tras inspeccionar algunas alternativas, llegamos a la conclusión de que estando el sistema en producción no podíamos hacer las modificaciones correspondientes, al menos las correspondientes al modelo de datos, pues implicaría hacer un mantenimiento adaptativo del resto de la aplicación, extremo que la empresa que ejecutaba el software no se podía permitir. Además surgió otro inconveniente: de hacerlo así, los datos sólo estarían disponibles para una determinada aplicación, y sería muy complicado integrar los resultados en otras aplicaciones, por lo que podía no ser una propuesta suficientemente viable.

Tras inspeccionar los trabajos propuestos por Missier et al en [5] o Stivilia et al en [6] basados en la aplicación de tecnologías semánticas para gestionar aspectos de calidad de datos, llegamos a la conclusión de que la solución pasaba por crear una capa adicional que permitiese gestionar los aspectos relacionados con la medición de la calidad de los datos, y que además permitiera tenerlos disponibles para distintas aplicaciones.

En este artículo presentamos una parte de los avances que hemos ido haciendo al crear tanto la capa, a la que hemos llamado **mapas de agregación,** como los aspectos tecnológicos asociados.

El resto del artículo está estructurado como sigue: en la sección 2 se presenta los conceptos relevantes relacionados con calidad de datos que debemos tener en cuenta para esta gestión. En la sección 3 presentamos los fundamentos de los mapas de agregación, y finalmente, en la sección 4 presentamos algunas conclusiones y líneas de trabajo futuro.

2. Conceptos de Calidad de Datos

Existen numerosas definiciones del concepto de calidad de datos, pero la más aceptada es sin duda aquella basada en la idea de *fitness for use* [7]. Siguiendo esta definición, una de las estrategias más comunes para afrontar el estudio de la calidad de datos (en adelante DQ, de las siglas en inglés Data Quality, por simplificar) consiste en descomponer la percepción global de calidad de datos de un conjunto de datos en una serie de "*subcalidades*" denominadas **dimensiones de DQ** [7], (al estilo de lo que hace la ISO 9126 [8] para el software) y establecer determinados criterios de aceptación para cada una de ellas. Estos criterios de aceptación deben estar definidos

en función de los posibles rangos de los resultados de medición. Así, si tras medir el nivel de calidad según cada una de las dimensiones de DQ del documento, se obtienen resultados fuera de los rangos de aceptación preestablecidos, entonces el documento podría considerarse como "*defectuoso*" y, por tanto, podría considerarse el no usarlo para la tarea, o al menos saber que se puede estar cometiendo un error al usarlo. La situación deseable es que existiese un sistema de medición integrado en el sistema de información que fuera capaz de identificar esos datos o documentos "defectuosos" y recomendar al usuario, al menos, su "no utilización".

Al conjunto de dimensiones de DQ que se utiliza en un contexto determinado para determinar el nivel de DQ de un documento se le denomina **modelo de DQ.** Para cada contexto (entendiéndose como tal el ámbito de una tarea) es necesario identificar aquellas dimensiones que mejor representen los requisitos de DQ de los usuarios. La bibliografía muestra ejemplos de modelos de DQ específicos para ciertos contextos: hospitalarios y de salud [9], o web [10], por nombrar unos cuantos representativos. Así mismo es interesante mencionar que ISO, ha publicado recientemente el estándar ISO/IEC 25012 [11], como parte de la familia SQUARE y que dicho estándar propone un modelo genérico de DQ para Sistemas de Información. En cualquier caso, el modelo que más se viene utilizando es el propuesto por Strong et al. en [1]. Estos autores agrupan las dimensiones DQ en cuatro categorías, que hacen referencia a los puntos de vista desde los cuales es posible evaluar la calidad de los datos (Tabla 1).

Categoría	Descripción
Intrínseca de DQ	Con dimensiones como *precisión*, *objetividad*, *credibilidad*, y *reputación*. Se refiere a la calidad que tienen los datos por sí mismos;
Accesibilidad de DQ	Con dimensiones como *accesibilidad*, *seguridad en el acceso*. Están orientadas a determinar si el alcance de los mecanismos de seguridad es suficiente como para poder acceder a los datos en condiciones de poder ser usados adecuadamente
Contextual DQ	Que engloba dimensiones como *relevancia*, *valor añadido*, *actualidad*, *compleción*, *cantidad apropiada de datos*. Estas dimensiones se refieren al alcance en que los datos pueden ser usados en un contexto determinado.
Representacional de DQ	Que engloba dimensiones como *interpretabilidad*, *facilidad de compresión*, *grado de representación concisa*, *grado de representación consistente*. Estas dimensiones se orientan a determinar si la forma en la que se representan y/o transmiten los datos les hace o no usables.

Tabla 1. Categorías de dimensiones definidas por Strong et al en [1] .

Una definición más completa del significado de estas dimensiones, asumiendo la posibilidad de la dependencia del contexto, puede encontrarse en distintos trabajos en la bibliografía, aunque los más interesantes son, a nuestro juicio, aquellos presentados en [7, 12, 13].

Siguiendo con el proceso de evaluación, el siguiente paso es definir propiamente cómo medir la DQ de los datos para cada una de estas dimensiones. Dado el gran número de propuestas en la bibliografía relacionadas con los aspectos de medición de datos, para una mejor comprensión se recomienda seguir la nomenclatura basada en ISO/IEC 15939 proporcionada por Caballero et al. en [4]. En este trabajo se intenta unificar las diferentes terminologías referidas a la medición de DQ encontradas en la literatura. Además destaca la existencia de tres tipos de medidas: las medidas base (que requieren un método de medición), las derivadas (que requieren una función de cálculo) y los indicadores (que requieren de un modelo de análisis y de un criterio de decisión). En la bibliografía [7, 12], se suele recoger una función de medición basada

en el porcentaje (ratio) de unidades de datos que satisfacen o no un criterio, tal y como está representado en la Figura 1. En base a esta fórmula se suelen definir la mayoría de las funciones de medición de DQ.

$$DQ_{Medida} = 1 - \frac{NúmeroDeUnidadesDeDatosQueNoSatisfacenUnCriterio}{NúmeroTotalDeUnidadesDeDatos}$$

Figura 1. Una Función de Medición de Calidad de Datos**.**

En la fórmula de la Figura 1, se pueden apreciar dos medidas base: *NúmeroDeUnidadesDeDatosQueSatisfacenUnCriterio* y *NúmeroTotalDeUnidades-DeDatos.* El método de medición de ambas consiste en un conteo del número de unidades. Si para la segunda no hay mayor complicación que contar (por ejemplo, con un *select count(*) from NombreTabla* en SQL), para la primera aparece el problema de expresar, usando la tecnología asociada al sistema de información, si se cumple o no el criterio. Un criterio se puede definir como una **regla de negocio** que define cuándo un dato es válido desde el punto de vista del negocio (tanto de la semántica como de la sintaxis). Según Loshin en [14], esta regla de negocio puede recoger parte del *know-how* empresarial (por ejemplo los sueldos de los trabajadores según sus categorías), la legislación relacionada (como por ejemplo, edades legales para trabajar), u otros aspectos [13, 15, 16]. En cualquier caso, el resultado de decidir si una unidad de datos satisface un criterio puede ser por simplicidad "*Verdadero*" o "*Falso*", aunque en algunas otras ocasiones es posible permitir distintas gradaciones del nivel de cumplimiento ("*Mucho*", "*Normal/Suficiente*", "*Nada*"). Usando la formulación más sencilla, y a fin de obtener un valor para una determinada medida, hay que contar cuántas unidades de datos obtienen un resultado de "*Verdadero*" en la prueba de *no satisface un criterio.* En entornos relacionales este conteo puede ser fácilmente resuelto introduciendo en la instrucción de consulta la cláusula *where*, siempre que sea factible representar así la condición del criterio: por ejemplo contar el número de valores nulos para la dimensión de compleción, o que el valor del dato pertenezca a un dominio definido como parte de la base de datos (en el caso de que el sistema gestor de bases de datos permitiera definir dominios), o bien en el caso de que el dominio se pudiera definir como un conjunto de valores almacenados en otra tabla y relacionados mediante integridad referencial: si está adecuadamente implementado, un intento de inserción de valores que violase el dominio generaría el correspondiente error, como es sabido. En estos casos los criterios pueden ser establecidos objetivamente e independientemente de cualquier otra evidencia.

Como se dijo anteriormente, para evaluar algunas dimensiones de calidad de datos se requiere la emisión de un juicio, individual o colectivo, sobre el valor del dato. Pero muchas veces el dato por sí mismo no aporta suficiente información para emitir dicho juicio, sino que es necesario hacerlo sobre un dato adicional (en [5] son llamados "evidencias") que completa el significado del dato en la dirección de la dimensión que se pretende evaluar. Este dato de juicio puede ser objetivo o subjetivo y se puede realizar sobre datos adicionales proporcionados de manera también objetiva o subjetiva. La Tabla 2, recogida en [4], muestra algunos ejemplos de escenarios donde se emiten juicios objetivos o subjetivos teniendo en cuenta datos adicionales proporcionados.

	Juicio Objetivo	Juicio Subjetivo
Valores Objetivos	El estudio de la *actualidad* de una oferta de trabajo publicada en Internet con una fecha de inicio y otra fecha de final de validez. El portal Web debería proporcionar ambas fechas a los usuarios, de modo que puedan hacer un juicio basándose en evidencias objetivas al comparar la actualidad de la oferta con la fecha actual.	El estudio del *Valor Añadido* de un dato: Alguien interesado en una cámara digital con zoom óptico mira las especificaciones técnicas del producto que busca para saber si el fabricante proporciona además información sobre el sistema de alimentación (pilas o baterías). Si lo encuentra el usuario debe decidir si los datos proporcionados hacen mejor o no la información que tiene sobre la cámara.
Valores Subjetivos	En el estudio de la *reputación:* Alguien está interesado en ver una película y busca críticas en internet o en un periódico. Puede tener más ganas de ver la película o perder el interés en ella, si la crítica que encuentra está hecha por alguien que se considera que tiene siempre un buen criterio a la hora de hacer las críticas.	El estudio de la *credibilidad* de una noticia publicada en un periódico y que ha sido proporcionada por una agencia de noticias. Si alguien tiene prejuicios de que las noticias publicadas por ese periódico defienden unas determinadas ideas políticas, en caso de tener ideas distintas podría no creer las noticias publicadas, o en caso contrario creer la mayoría de ellas.

Tabla 2. Ejemplos de tipos de escenarios donde la calidad de los datos se mide en función de datos adicionales proporcionados [4].

De todos modos, se requiere tener tanto el o los datos adicionales necesarios, almacenados y enlazados convenientemente junto con el dato cuya calidad se pretende evaluar y con la correspondiente regla de negocio. Como puede apreciarse, la dificultad está en mantener esta infraestructura, y es incluso más complicado para aplicaciones ya desarrolladas. Se necesita por tanto describir y almacenar adecuadamente la regla de negocio para poder reutilizarla después para otras instancias del proceso de medición, bien dentro de la misma aplicación, bien como parte de otras aplicaciones. En cualquier caso, es patente la necesidad de modificar tanto el modelo de datos como el modelo de negocio de la aplicación. Este paso depende obviamente de la tecnología usada en el desarrollo de la aplicación: sistemas gestores de bases de datos (SGBD) más o menos avanzados cuentan con lenguajes procedimentales embebidos, PLSQL o TransactSQL, para acometer estas tareas, pero no todas las empresas pueden tener o tienen posibilidad de utilizar estos SGBDR. Además nuevos modelos de negocio demandan el uso de otras tecnologías y arquitecturas que permitan el acceso remoto a través de mecanismos bien definidos y asequibles como puede ser http. Precisamente, este va a ser uno de los muros que tenemos que superar en nuestra propuesta.

Otra dificultad añadida es la forma de operar con los resultados obtenidos de los procesos individuales de medición para cada una de las dimensiones: si todos los procesos de medición proporcionan como resultado medidas expresadas en la misma escala numérica (típica y preferiblemente en formato ratio como se describió anteriormente), entonces la evaluación de un dato se hace mediante una media ponderada entre las dimensiones y los resultados de la medición [12, 17], aunque no debe descartarse el uso de etiquetas lingüísticas, que necesitan de operadores específicos para obtener un resultado agregado de todas ellas [18, 19]. Es importante destacar que normalmente, y en aras de la simplicidad, los modelos de calidad de datos dan la misma importancia (peso) a todas las dimensiones de calidad que la

componen. Pero, como Wang y Strong demostraron en [20], no todas estas dimensiones son igualmente importantes como queda demostrado en [17], por lo que puede ser necesario hacer previamente una estimación del peso o importancia que se debe asignar a cada una de las dimensiones.

3. Una Arquitectura Software para gestionar mediciones de DQ

Volviendo al problema que motivó esta línea de investigación y teniendo en cuenta los contenidos presentados en la sección segunda, en este apartado queremos introducir brevemente una primera aproximación a la solución.

Básicamente nuestra propuesta consiste en una arquitectura consistente en dos capas: una que complementa al modelo de datos y otra que complementa al modelo de procesos del Sistema de Información. Por razones de eficiencia, diseño y seguridad es un complemento no invasivo es decir, en la medida de lo posible, no debe modificar ninguna de ellas.

La capa de datos, a la que llamamos **mapas de agregación**, tiene como objetivo recoger los datos adicionales necesarios para la evaluación. Además de la necesidad de que los datos estén suficientemente auto-descritos, debemos tener en cuenta el requisito tecnológico de la interoperabilidad semántica. Por todo ello pensamos en el uso de tecnologías semánticas ya que permiten un descubrimiento más efectivo, la automatización e integración así como la reutilización de los datos entre distintas aplicaciones [21]. Además de satisfacer los requisitos, aportarían valor añadido a nuestra solución, como la posibilidad de disponer de espacios de anotación coherentes con el espacio de datos.

Por su parte, la capa que complementa al modelo de procesos, a la que llamamos **Capa de Gestión de Lógica de DQ,** debería englobar tanto los procesos técnicos relacionados con la adquisición y mantenimiento de estos datos adicionales, las correspondientes reglas de negocio, y su ligadura a los datos almacenados en las bases de datos organizacionales, como aquellos procesos de gestión orientados a la computación y cálculo de los niveles de calidad de datos de los datos basados en la reglas de negocio. Además también existiría la posibilidad de extrapolar medidas en contextos similares usando razonadores, establecer mecanismos de recomendación de documentos, y la posibilidad de optimizar ciertos procesos típicos del sistema de información atendiendo a criterios de calidad de datos, como puede ser la búsqueda y recuperación de información filtrando aquellos documentos (o conjunto de datos) cuyos niveles de calidad no alcancen los mínimos deseables.

A fin de modelar adecuadamente esta arquitectura software, son varias las tareas que tenemos que acometer: (1) identificar claramente qué datos están afectados por los requisitos de calidad de datos, (2) identificar cuáles son las dimensiones de calidad más adecuadas para los datos y cómo identificar los datos adicionales; (3) definir cómo conseguir valores para los datos adicionales y cómo anotarlos, y finalmente (4) identificar cómo calcular las evaluaciones de calidad de acuerdo a la percepción de DQ de diferentes grupos de agentes usando los modelos de calidad de datos proporcionados.

A continuación presentamos los retos de investigación que nos estamos encontrando a la hora de aplicar esta solución.

En la Web Semántica, también llamada Web de Datos, los datos se modelan como un grafo etiquetado dirigido, donde cada nodo se corresponde con un recurso (sujetos y objetos), y cada arco con un predicado. En una primera aproximación proponemos es hacer anotaciones RDF de los datos adicionales correspondientes a las dimensiones de calidad que son de interés para un determinado grupo de usuarios.

A partir del Modelo de Información de Medición de Calidad de Datos presentado en [4], hemos creado una ontología usando OWL que nos permitirá hacer las anotaciones pertinentes. Una vez acotado el espacio de datos, es preciso acotar el espacio de los elementos anotables según el nivel de granularidad. En [22] los autores identifican los siguientes: RDF Database, Documentos de Web Semántica (SWD) subgrafos RDF o Web Semántica. En esta primera aproximación hemos decidido centrarnos en las sentencias (como si fuesen hechos), por lo que nuestro objetivo será escribir sentencias de calidad de datos sobre sentencias, proceso conocido como reificación [23]. Para identificar las dimensiones de calidad que mejor se adaptan a un contexto, proponemos al lector consultar algunos de los modelos de calidad de datos propuestos en la literatura: por ejemplo para entornos Web se puede consultar el propuesto por Moraga et al en [24].

Por ejemplo, consideremos la sentencia "*Buddy owns a business*" y "*business has-WebSite http://www.budy.com"* cogido directamente de [23] y mostrado en la Figura 2. Imaginemos que alguien quiere conocer por ejemplo cómo de fiable, qué grado de reputación tiene o lo actual que las sentencias son. Para ello se necesitarán datos adicionales como los marcados en la mencionada figura.

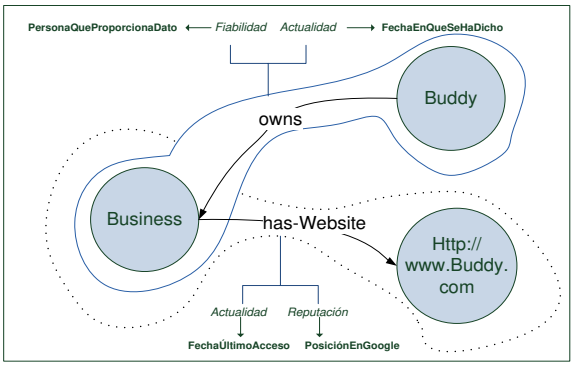

Figura 2. Dimensiones de DQ para diferentes Sentencias, extraído de [24]

Desde un punto de vista práctico/tecnológico aparecen cuatro aspectos claves en la medición de DQ sobre los que estamos trabajando en la actualidad: (1) cómo podemos acceder a los datos almacenados en las bases de datos organizacionales y cómo representarlos respetando tanto el modelo de datos como la legalidad (2) quién debe proporcionar los datos adicionales y cuántos deberían recogerse (3) cómo y dónde deben almacenarse estos valores y (4) cómo se puede conseguir un valor representativo para todos los valores del mismo dato adicional para todos los usuarios a fin de evaluar la calidad.

Según estamos constatando en nuestras investigaciones los principios de DataSpaces [25] y de Linked Data [26], definido como un conjunto de buenas prácticas para la publicación y conexión de datos estructurados en la web, permitirían plantear una respuesta a la pregunta (1). Para el caso de Sistemas de Información con Bases de datos heredadas, disponemos de un sistema de encapsulamiento (*wrapping*) definido en [27], que convierte mediante reingeniería una base de datos heredada en servicios Web, y que al estar basado en transformaciones MDA, y definiendo las adaptaciones oportunas, se podrían expresar de acuerdo a la ontología de medición de DQ creada a tal efecto. Para responder a las preguntas (2) a (4) se están explorando los beneficios de la redes sociales: en [19], los autores introducen formas de almacenar las diferentes opiniones y de operar con ellas para obtener un nivel de calidad.

4. Conclusiones y trabajo futuro

En este artículo hemos descrito brevemente los avances que vamos realizando en nuestra línea de investigación, en la que hemos observado la necesidad de tener en cuenta los aspectos de calidad de datos en la operativa de los sistemas de información para optimizar su rendimiento. Una parte fundamental de este trabajo consiste en poder generar, almacenar y usar datos relativos a la calidad de datos de modo transparente a las aplicaciones, facilitando así la integración y reutilización de dichos datos a través de distintas aplicaciones.

Esta labor, que está siendo desarrollado actualmente, tiene abiertas dos líneas de trabajo: una dedicada a la identificación y aplicación de modelos de calidad de datos a distintos contextos, sobre todo relacionados con la Web (como por ejemplo aplicaciones Web 2.0 y *Mashups*) y por otro el establecimiento y prueba de los mapas de agregación (usando los modelos de calidad obtenidos en la línea paralela), así como la prueba de los procedimientos correspondientes para la adquisición de datos adicionales, y el cálculo de los niveles de calidad.

Agradecimientos. Esta investigación forma parte de los proyectos DQNet (TIN2008-04951-E) y ESFINGE (TIN2006-15175-C05-05) financiada por el MEC, e IQMF-Tool financiada por la UCLM.

Referencias

1. Strong, D.M., Lee, Y.W., Wang, R.Y.: Data Quality in Context. Communications of the ACM **40** (1997) 103-110
2. Wang, R.Y., Madnick, S.: Data Quality Requirements: Analysis and Modelling. Ninth International Conference on Data Engineering (ICDE'93). IEEE Computer Society, Vienna, Austria (1993) 670-677
3. Price, R., Shanks, G.: The Effect of Data Quality Tag Values and Usable Data Quality Tags on Decision-Making. MCIS'09. LNCS, Brisbane, Australia (2009)
4. Caballero, I., Verbo, E.M., Calero, C., Piattini, M.: A Data Quality Measurement Information Model based on ISO/IEC 15939. 12th ICIQ, MIT, Cambridge, MA (2007)

5. Missier, P., Embury, S., Greenwood, M., Preece, A., Jin, B.: Quality views: capturing and exploiting the user perspective on data quality. Proceedings of the 32nd international conference on Very large data bases-Volume 32 (2006) 977-988
6. Stivilia, B.: A Model for Information Quality Change. In: Robbert, M.A., O'Hare, R., Markus, M.L., Klein, B. (eds.): 12th International Conference on Information Quality, MIT, Cambridge, USA (2007) 39-49
7. Lee, Y.W., Pipino, L.L., Funk, J.D., Wang, R.Y.: Journey to Data Quality. Massachussets Institute of Technology, Cambridge, MA, USA (2006)
8. ISO/IEC: ISO/IEC 9126. Software Engineering-Product Quality. Parts 1 to 4. International Organization for Standardization/International Electrotechnical Commission. (2001)
9. Al-Hakim, L.: Procedure for Mapping Information Flow: A case of Surgery Management Process. In: Al-Hakim, L. (ed.): Information Quality Management: Theory and Applications. Idea Group Publishing, Hershey, PA, USA (2007) 168-188
10.Caro, A., Calero, C., Caballero, I., Piattini, M.: A proposal for a set of attributes relevant for Web Portal Data Quality. Software Quality Journal (2008)
11.ISO/IEC-JTC1/SC7: CD 25012.2 Software engineering: Software Quality Requirements and Evaluation (SQuaRE) A“ Data Quality Model - N3574- 2006-07-10. International Organization for Standarization (2008)
12.Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques. Springer-Verlag Berlin Heidelberg, Berlin (2006)
13.English, L.: Improving Data Warehouse and Business Information Quality: Methods for reducing costs and increasing Profits. Willey & Sons, New York, NY, USA (1999)
14.Loshin, D.: Enterprises Knowledgement Management: The Data Quality Approach. Morgan Kauffman, San Francisco, CA, USA (2001)
15.Loshin, D.: Data Quality and Business Rules. In: Piattini, M., Calero, C., Genero, M. (eds.): Information and Database Quality. Kluwer Academic Publishers (2001)
16.Wang, R.Y.: A Product Perspective on Total Data Quality Management. Communications of the ACM **41** (1998) 58-65
17.Helfert, M., Foley, O., Ge, M., Cappiello, C.: Limitations of Weighted Sum Measures for Information Quality. AMCIS, San Francisco, CA, USA (2009)
18.Herrera-Viedma, E., Pasi, G., Lopez-Herrera, A.: Evaluating the Information Quality of Web Sites: A Quality Methodology Based on Fuzzy Computing with Words. Journal of American Society for Information Science and Technology **54** (2006) 538-549
19.Caballero, I., Verbo, E.M., Serrano, M.Á., Calero, C., Piattini, M.: Tailoring Data Quality Models using Social Network Preferences. 2nd International Workshop on Managing Data Quality in Collaborative Information Systems Springer, Brisbane, Australia (2009) 1-15
20.Wang, R., Strong, D.: Beyond accuracy: What data quality means to data consumers. Journal of Management Information Systems; Armonk; Spring 1996 **12** (1996) 5-33
21.Guha, R., McCool, R., Miller, E.: Semantic search. Proceedings of the 12th international conference on World Wide Web. ACM Press, Budapest, Hungary (2003) 700-709
22.Ding, L., Finin, T., Joshi, A., Peng, Y., Pan, R., Reddivari, P.: Search on the Semantic Web. IEEE Computer **38** (2005) 62-69
23.Daconta, M., Obsrt, L., Smith, K.: The Semantic Web: A guide to the future of XML. Web Services and Knowledge Management. Willey Inc, Indianápolis, Indiana (2003)
24.Moraga, C., Moraga, M., Calero, C., Caro, A.: SQuaRE-Aligned Data Quality Model for Web Portals. QSIC'09. IEEE, Jeju (2009)
25.Halevy, A., Frankling, M., Maier, D.: Principles of DataSpaces Systems. PODS (2006)
26.Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. Special Issue on Linked Data, International Journal on Semantic Web and Information System (2009) To appear
27.Pérez-Castillo, R., García-Rodríguez , I., Caballero, I.: PRECISO: a reengineering process and a tool for database modernisation through web services. SAC'09 (2009) 2126-2133

Context-aware and Home Care: Improving the quality of life for patients living at home

Juan A. Fraile[1], Javier Bajo[1] and Juan M. Corchado[2]

[1]Pontifical University of Salamanca, c/ Compañía 5, 37002 Salamanca, Spain
[2]Departamento de Informática y Automática, University of Salamanca,
Plaza de la Merced s/n, 37008 Salamanca, Spain
{jafraileni, jbajope}@upsa.es, corchado@usal.es

Abstract. This paper presents a multiagent system that facilitates the performance of daily tasks for people within a context-aware environment. The paper analyzes the important aspects of context-aware computing and presents a prototype that can be applied to monitor dependent individuals in their homes. The system includes computation elements that are integrated within a domestic environment with the goal of capturing context-related information and managing the events carried out by the patient. The services are support by the processing and reasoning out of the data received by the agents in order to offer proactive solutions to the user. The results obtained with this prototype are presented in this paper.

Keywords: Context-Aware Computing, Home Care

1 Introduction

The preferred characteristics when designing software applications include autonomy, security, flexibility and adaptability. In order to achieve this objective, it is necessary to have mechanisms, methods and tools that can develop systems capable of adapting to changes within the environment. The search for flexible software applications that can continually improve their ability to adapt to the demands of the users and their surrounding leads us to context-aware systems that store and analyze all of the relevant information that surrounds and forms a part of the user environment.

Context-aware systems provide mechanisms for developing applications that understand their context and are capable of adapting to possible changes. A context-aware application uses the context of its surroundings to modify its performance and better satisfy the needs of the user within that environment. The information is usually obtained by sensors. The current trend for displaying information to the system users, given the large number of small and portable devices, is the distribution of resources through a heterogeneous system of information networks. Web applications and services have been shown to be quite efficient [15] in proc-

essing information within this type of distributed system. Web applications are run in distributed environments and each part that makes up the program can be located in a different machine. Some of the web technologies that have had an important role over the last few years are multiagent systems and SOA (Service Oriented Architecture) architectures, which focus on the distribution of system service functionalities. This model provides a flexible distribution of resources and facilitates the inclusion of new functionalities within changing environments. In this respect, the multiagent systems have also already demonstrated their aptitude in dynamic changing environments [3] [9]. The advanced state of development for multiagent systems is making it necessary to develop new solutions for context-aware systems. It involves advanced systems that can be implemented within different contexts to improve the quality of life of its users. There have been recent studies on the use of multiagent systems [3] as monitoring systems in the medical care [2] patients who are sick or suffer from Alzheimer's [9]. These systems provide continual support in the daily lives of these individuals [8], predict potentially dangerous situations, and manage physical and cognitive support to the dependent person [4].

This paper presents the Home Care Context-Aware Computing (HCCAC) multi agent system that supervises and monitors dependent persons in their homes, providing the user with a certain degree of self-sufficiency. The proposed system focuses on incorporating mechanisms that facilitate the integration of web applications. The HCCAC system provides wireless communication between its elements, and integrates intelligent agents with sensors and autonomous components that obtain context-aware information and are proactive in their interaction with the users. HCCAC facilitates the automation of context devices and the ability to respond to the elements from a remote location. One of the most important characteristics of HCCAC is the use of intelligent agents as one of the principal components in its system services-oriented approach. The system proposes a new and simple method of a distributed construction where the system functionalities are modeled as services that are activated by agents, which, essentially, control and coordinate the services.

The remainder of the paper is structured as follows: section 2 presents the problems of context-aware computing and introduces the need for developing new systems that can improve the living conditions of patients in their homes. Section 3 describes the proposed system and the interaction between agents and devices, focusing specifically on context-aware capabilities and the value-add provided by HCCAC. Finally, section 5 presents the results and conclusions obtained after evaluating the prototype in a Home Care scenario.

2 The context-aware computing.

The idea behind context-aware systems began when Want et al. [19] presented the Active Badge Location System, which is considered to be the first context-aware application. It is a system for locating individuals in their office, where each per-

son wears a badge that uses network sensors to transmit information signals about their location to a centralized service system. In the mid-1990s, several location-aware tour guides emerged [1] [7] offering information about the location of the user. The most commonly used context-aware feature is by far user location. Over the last few years the use of other information attributes for context-aware systems has grown. It is difficult to describe the term "context-aware" and many researchers try to find their own description and the relationship among the features that are included in context-aware systems. The first written reference to the term "context-aware" appeared in Schilit and Theimer [18]. There are authors that describe context-aware as the location or identification of persons or objects [17] [12] [5]. These descriptions are frequently used during the initial research of these systems. One of the more precise definitions was provided by Dey and Abowd [10]. These authors refer to context-aware as information that can be used to determine the situation of entities (e.g., people, places or objects) that are considered relevant for the interaction between a user and an application.

There are several location-aware infrastructures capable of gathering positional data [11] [6] [13] [16]. These systems include GPS satellites, mobile telephone towers, proximity detectors magnetic card readers, barcode readers, etc. These sensors can provide information about location or proximity, differing mainly in the precision detail. Some need a clear line of vision, other signals can penetrate walls, etc. The previously mentioned systems only use one context attribute: positional information for the object or person. The use of different context attributes such as noise, light and location allows for the combination of much higher levels of context-aware objects. These elements are necessary for building systems that are more useful, adaptive and easy to use. An example of a this type of context-aware infrastructure is the system presented by Muñoz et al. [14], which improves communication by adding context-awareness to the management of information within a hospital environment. All of the users (doctors, nurses, etc.) are given a mobile device for writing messages that are sent when a determined number of circumstances are met. The contextual attributes that are included in this system are location, time, roles and the state of the user or entity that is being analyzed.

The previously mentioned case studies demonstrate that the attributes used by the majority of context-aware systems are the identification and location of people, objects or entities. Few systems can use information from different context attributes and connect different types of data to interact with users or patients. We would like to take the next step and, in addition to using different context attributes in the system that we propose, make it possible for the different types of data gathered by the system to be stored, processed and reasoned out in order to improve the quality of life for dependent persons living at home.

We would like to propose the context model-based Home Care Context-Aware Computing (HCCAC) multiagent system that offers context-aware services for patients in a dependent environment and includes a set of independent services that can gather and interpret context data. The fundamental characteristics of the system are to process and reason out data furnished by the context to improve patient care. The system allows for easy development of context-aware services and applications in a variety of contexts.

3. HCCAC multiagent system.

HCCAC can be defined by the need to control various devices and gather user information in a non-intrusive and automatic way within Context-Aware environments [5]. Additionally, HCCAC brings greater speed to the gathering, processing and storage of data, variety in the communication channels among system players, and security in the management of access and administration of information. Each of these characteristics allows users to feel safe within the context knowing that they are being monitored and that any type of unexpected incident can be prevented. Additionally, the variety of communication channels also facilitates the administration of the system and the interaction between the components within the environment. The HCCAC system also facilitates the integration and management of agents and devices. Communication between platform agents follows the standard FIPA ACL [8] (Agent Communication Language) specification. Communication protocol between agents and services is based on the SOAP [9] standard. The services that the agents invoke can be of two types: services that capture information from the context data that are obtained by the autonomous components; and services that function within devices installed in the context. All of this allows HCCAC to be a system that is easy to implement in complex environments and is not platform specific.

HCCAC makes it possible to automatically obtain information on users, their actions and environment in a distributed manner. HCCAC primarily focuses on monitoring a person in their home and sending notifications on the state of the individual or possible incidents. Additionally, it combines the management of personal information with a model of daily activities that are developed by using the data provided by the sensors through the household network. The interpretation and reasoning of the base knowledge and the daily activity models developed by the system provide an added value to the system. HCCAC makes is possible to easily use and share context-aware applications within changing physical spaces. As seen in Figure 1, the system is composed of the following agents:

- Provider agents capture and summarize the data obtained by both internal and external heterogeneous context sources so that the Interpreter and Database agents can process and reuse these data. The system is dynamic and is capable of incorporating an information Provider agent at any given moment by adding the corresponding sensor or capturing the necessary information from a server or external provider.
- Database agents store the context data obtained by the Provider agents. The organization of this information is similar for different environments. This agent is in charge of managing and exploiting the stored data. Additionally it provides the necessary information to the interpreter agent and records and updates the action plans that are determined by the interpreter agent.
- Interpreter agent provides logical reasoning services to process the contextual information. The interpreter agent reasons out the actions that must be taken by creating a plan that determines the optimal course of action for reaching an objective. This agent provides the ability to reach high level

and complex objectives and avoids errors that could result in inefficacies. It also allows for greater flexibility when dealing with new objectives.

- Resource Identification Agent (LcA) makes it possible for the provider agent and the interpreter agent to work directly with the applications and the users in order to avoid dangerous situations or particular incidents with the user. This agent is in charge of maintaining a record of the provider agents that are active in the system, in addition to allowing or denying the inclusion of new provider agents.

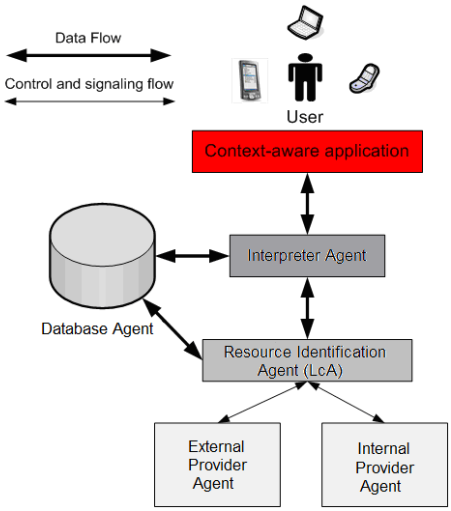

Fig. 1. Overview of the multi-agent system HCCAC.

Context-aware applications in HCCAC also check the information available from the context providers and are in constant listening mode to deal with possible events that the context providers transmit. The applications use different levels of context information and adapt their behavior according to the active context. They check the functionalities registered in the system and have a location for all of the context providers made available within the environment. One way of developing context-aware applications is to specify actions that respond to changes in the context under specific conditions and rules.

The agents described function independently from the platform on which they are installed. The external provider agents obtain context information through external resources such as, for example, a server that provides meteorological information about the weather in a specific place, or a location server that provides information on the location of a person who is not at home. The internal provider agents gather information directly from the sensors installed within the environment, such as RFID based location sensors installed in the home of a patient, or light sensors. The functions of the interpreter agent include both processing information provided by the database agent, and reasoning out the information that has been processed.

The interpreter agent offers context data of high level interpretation to both the context-aware applications and the devices in which it acts, from low-level context data. Context-aware applications use different levels of context information and can adapt their performance according to the context within which they are executed. After consulting the data registered by the LcA, these applications can locate the web services handled by the context providers. Context provider agents facilitate the access of the applications and web services that provide information to the system. For example, web applications provide meteorological data to the HCCAC system. In order to obtain context data, context-aware applications can ask the Provider agent or wait for an event from the Provider agent. LcA allows users and agents to locate different context applications. LcA controls large areas where there are context providers that can be located in internal or external networks. The LcA searches and adapts to the changes that are introduced within the context when adding or deleting physical sensors or reconfiguring the actual devices. It also unfolds a mechanism that allows context providers to notify about their functionalities within the LcA.

4. Using context-aware computing to apply patient control.

Our case study developed prototype for improving the quality of life for a patient living at home. The system gathers information from the sensors that capture data and interact with the context. The primary information gathered by the sensors that have been installed is the location-aware for the user in the environment. The system also processes information relative to the temperature in the different rooms of the patient's home, and the lighting conditions in the areas where the user moves about. All of the access doors in the house have an automatic open and close mechanism. The house was installed with (i) passive infrared motion detectors on the ceiling and (ii) automatic door opening mechanisms. The motion detectors and door opening mechanisms, interact with the microchip Java Card & RFID [11] users to offer services in run time. Each dependent user is identified by a (iii) Sokymat ID bracelet Band Unique Q5 that has an antenna and a RFID-Java-Crypto-Card chip with a 32K Module and Crypto-CoProzessor (1024 bit RSA) compatible with the SUNs JavaCard 2.1.1 [20]. También se instalan (iv) sensores de luminosidad, que se utilizan para gestionar el nivel de luminosidad de una estancia haciendo que se mantenga entre unos valores pre-asignados modificables, (v) una pantalla TFT de superficie, que permite visualizar y gestionar los elementos principales del sistema. There were also light sensors (iv) installed to manage the level of lighting in the home, maintaining the level within a pre-determined set of values, (v) a TFT surface screen that makes it possible to see and manage the principal elements of the system. The sensors or actuators are placed in strategic positions within the home. All of these devices are controlled by agents. The sensors network is responsible for generating alarms after comparing the user's current state with the parameters of the user's daily routine, which has been stored in the system. The system can generate alarms if it recognizes a significant change

within the parameters of the user's stored daily routine, such as if the user gets up prior to a specific hour on a non-work day, if the use spends more time than normal standing at a door without entering, or if the user remains motionless in the hallway for an extended period of time.

The main idea, which can be considered the core of the case study, is the system's ability to hide the available technological resources from the patient, ensuring that they remain concealed from the patient's daily life activities. In this way, users have only to be concerned with informing the system of their preferred living conditions and the system itself takes care of managing the resources and acting accordingly. It is necessary to apply or develop an appropriate analysis and design methodology, and to develop an organizational structure. The first step, however, is to analyze the extent of the system's performance as it is installed in the patient's home.

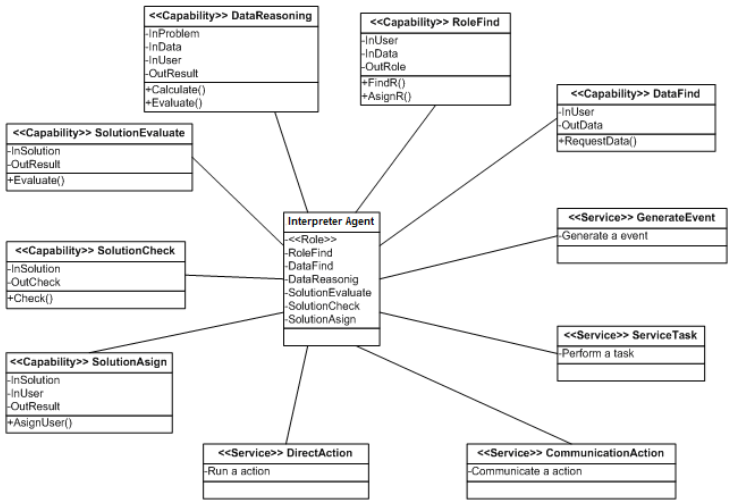

Fig. 2. Diagram of the different types of Interpreter Agents

The agent on which the HCCAC system is based is the Interpreter Agent. The diagram in Figure 2 shows the different types of Interpreter Agent, which involve a variety of roles or capacities: (i) RoleFind, assigns a role or finds a valid role for a user based on the identification of the user and the data assigned to him or her; (ii) DataFind, obtains information associated with a user; (iii) DataReasoning, detects any problem and the information associated with a user, evaluate the information and determines a solution to the problem; (iv) SolutionEvaluate, evaluates and transmites the result of a solution; (v) SolutionCheck, checks the solution and gives an assessment; and (vi) SolutionAssign, assings a solution to a user. The Interpreter agent can also initiate a series of services: (i) DirectAction, executions an action for a user; (ii) CommunicationAction, communicates an action that must be taken by a user; (iii) ServiceTask, initiates a task in the system; and (iv) GenerateEvent, generates an event after calculating or evaluating the data.

The interaction between agents and the sensors installed in the home for the prototype presented in this paper can be described by using an example, as shown in Figure 3, where the internal Provider Agent (Situ Ag) executes web services to send signals to the sensors, and communicates with the Interpreter agent to inform it of the user's situation. Situ Ag also receives signals from the Interpreter Agent in order to, for example, allow or prohibit user access to controlled areas. The LcA agent registers each of the services offered by the Situ Ag. The LcA agent continually updates the list of services provided by the Situ Ag so that it can be transferred through web services to the active applications in the context. Furthermore, the Database Agent manages the context information based on the data provided by the Interpreter Agent.

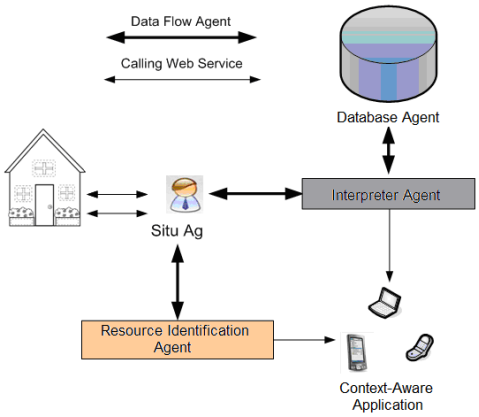

Fig. 3. Ejemplo de interacción entre agentes, recursos y aplicaciones Context-Aware.

It is also important to consider the transformation of the information that is produced within the system. First of all the system gathers data from the components of the context, which is then stored in the information system. Information on the context is taken from these data and subsequently reasoned out. To do so, the agents use case based reasoning (CBR) in a computational environment with intelligent processing capability. The CBR systems use the experience from similar previous situations in order to solve a new problem. This makes it possible to personalize the context of any accounting for user data and preferences that are pertinent to similar previous cases. The information Provider agents work together with the Interpreter agent to carry out this task.

5 Results and Conclusions.

HCCAC was used to develop a prototype used in the home of a dependent person. It incorporates JavaCard technology to identify and control access, with an added

value of RFID technology. The integration of these technologies makes the system capable of automatically sensing stimuli in the environment in execution time. As such, it is possible to customize the system performance, adjusting it to the characteristics and needs of the context for any given situation. Different studies related to context-aware systems, such as [11] [6] [13] [16], focus exclusively on gathering positional data on the user. The authors of these papers gather the positional data on the users through GSP signals, mobile telephone towers, proximity detectors, cameras and magnetic card readers. Many of these signals work with a very wide positioning range, which makes it difficult to determine the exact position of the user. In contrast, the system presented in this paper determines the exact position of the user with a high level of accuracy. To do so, the system uses JavaCard and RFID microchip located on the users and in the sensors that detect these microchips in their context. Others studies, such as [14], in addition to locating the users in their context, try to improve the communication between patients and medical personnel in a hospital center by capturing context attributes such as weather, the state of the patient or role of the user. In addition to capturing information from various context attributes such as location, temperature and lighting, HCCAC also incorporates the Interpreter agent reasoning process to provide services proactively to the user within a Home Care environment. HCCAC incorporates new information Provider agents in execution time. In this respect, HCCAC proposes a model that goes one step further in context-aware system design and provides characteristics that make it easily adaptable to a home care environment.

Although there still remains much work to be done, the system prototype that we have developed improves home security for dependent persons by using supervision and alert devices. It also provides additional services that react automatically in emergency situations. As a result, HCCAC creates a context-aware system that facilitates the development of intelligent distributed systems and renders services to dependent persons in their home by automating certain supervision tasks and improving quality of life for these individuals. The use of a multiagent system, web services, RFID technology, JavaCard and mobile devices provides a high level of interaction between care-givers and patients. Additionally, the correct use of mobile devices facilitates social interactions and knowledge transfer. Our future work will focus on obtaining a model to define the context, improving the proposed prototype when tested with different types of patients.

Acknowledgements. This work was supported JCYL SA071A08 project.

References

1. Abowd, G. D., Atkeson, C. G., Hong, J., Long, S., Kooper, R., and Pinkerton, M. (1997). Cyberguide: A mobile context-aware tour guide. Wirless Networks, 3(5).
2. Angulo, C., & Tellez, R. (2004). Distributed Intelligence for smart home appliances. Tendencias de la minería de datos en España. Red Española de Minería de Datos. Barcelona, España.

3. Ardissono, L., Petrone, G. and Segnan, M. 2004. A conversational approach to the interaction with Web Services. Computational Intelligence, Blackwell Publishing. Vol. 20. pp. 693-709.
4. Bahadori, S., Cesta, A., Grisetti, G., Iocchi, L., Leone1, R., Nardi, D., Oddi, A., Pecora, F. & Rasconi, R. (2003). RoboCare: Pervasive Intelligence for the Domestic Care of the Elderly. In AI*IA Magazine Special Issue, January, 2003.
5. Brown, P. J. (1996). The stick-e document: A framework for creating context-aware applications. In Proceedings of the Electronic Publishing, Palo Alto, pages 259–272.
6. Burrell, J. and Gay, G. (2002). E-graffiti: evaluating real-world use of a context-aware system. Interacting with Computers – Special Issue on Universal Usability, 14(4):301–312.
7. Cheverst, K., Davies, N., Mitchell, K., Friday, A., and Efstratiou, C. (2000). Developing a context-aware electronic tourist guide: some issues and experiences. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems, pages 17–24, New York, NY, USA. ACM Press.
8. Corchado, J.M. , Bajo, J. , de Paz, Y. & Tapia, D. (2008). Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. Decision Support Systems. ISSN 0167-9236.Vol 34 (2) pp. 382-396.
9. Corchado, J.M., Bajo, J. & Abraham, A. (2008). GERAmI: Improving the delivery of health care. IEEE Intelligent Systems. Special Issue on Ambient Intelligence - Mar/Apr 08.
10. Dey, A. K. (1998). Context-aware computing: The CyberDesk project. In Proceedings of the AAAI, Spring Symposium on Intelligent Environments, Menlo Park, CA.
11. Espinoza, F., Persson, P., Sandin, A., Nyström, H., Cacciatore, E., and Bylund, M. (2001). GeoNotes: Social and navigational aspects of location-based information systems. In Proceedings of the 3rd International Conference on Ubiquitous Computing, Atlanta, Georgia, USA, pages 2–17.
12. Hull, R., Neaves, P., and Bedford-Roberts, J. (1997). Towards situated computing. In Proceedings of the International Symposium on Wearable Computers.
13. Kerer, C., Dustdar, S., Jazayeri, M., Gomes, D., Szego, A., and Caja, J. A. B. (2004). Presence-aware infrastructure using web services and RFID technologies. In Proceedings of the 2nd European Workshop on Object Orientation and Web Services, Oslo, Norway.
14. Muñoz, M. A., Gonzalez, V. M., Rodriguez, M., and Favela, J. (2003). Supporting context-aware collaboration in a hospital: an ethnographic informed design. In Proceedings of Workshop on Artificial Intelligence, Information Access, and Mobile Computing 9th International Workshop on Groupware, CRIWG 2003, pp. 330–334.
15. Oren, E., Haller, A., Mesnage, C., Hauswirth, M., Heitmann, B., Decker, S.: A Flexible Integration Framework for Semantic Web 2.0 Applications. IEEE Software, vol. 24, no. 5, pp. 64-71, (2007)
16. Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. (2000). The cricket location-support system. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pages 32–43. ACM Press.
17. Ryan, N., Pascoe, J., and Morse, D. (1997). Enhanced reality fieldwork: The context-aware archaeological assistent. Computer Applications in Archaeology.
18. Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. IEEE Network, 8(5):22–32.
20. Want, R., Hopper, A., Falcao, V., and Gibbons, J. (1992). The Active Badge Location System. ACM Transactions on Information Systems, 10(1):91–102.
21. ZhiqunChen (Sun Microsystems). Java Card Technology for Smart Cards. Addison Wesley Longman. ISBN 0201703297.

Developing Advanced Services for SMEs using Service-Centric Tools: Experiences and Challenges

Francisco J. Garijo[1], Juan Pavón[2], Carlos Rodríguez[2], and Damiano Spina[2]

[1] Telefónica I+D,
Emilio Vargas 6, 28043 Madrid, Spain
fgarijo@tid.es
[2] Facultad de Informática, Universidad Complutense de Madrid,
Profesor José García Santesmases s/n, 28040 Madrid, Spain
jpavon@fdi.ucm.es, carlosrodriguez@computer.org,
damianospina@gmail.com

Abstract. This paper presents the experiences and the results achieved for delivering sophisticated Business Support Systems to Small and Medium Enterprise (SMEs), using advanced service centric solutions developed in the SECSE project. The SECSE methodology provides concepts methods, processes, and techniques for developing service centric applications, and the supporting toolkit include tools for supporting the principal activities such as publication, discovery, execution, monitoring, and testing. The paper illustrates the use of the methodology for building a pilot, which delivers to SMEs integrated solutions including computing and communication infrastructure, and complex functionality such as CRM, Work-Flow processing, and logistics. Metrics and evaluation results for assessing both the technical benefits and the business benefits of the SECSE outcome are also described.

1 Introduction

Materialization of the "service paradigm" which includes concepts like Service Oriented Development, Service Oriented Architecture, Service Oriented Applications and others, call for the need of a joint effort between industry and academia to provide technology, methodology and supporting tools for building service systems, and making them accessible to users. The SECSE project [1] has faced this challenge by gathering IT companies, developers and research labs to find out new solutions for exploiting the full potential of software services and for providing a sound engineering framework for the creation of services and service-centric systems. The project has delivered two principal outcomes: A methodology providing a conceptual model, methods, processes, and techniques for developing service centric applications, and supporting tools including capabilities to describe, compose, discover, monitor and deliver services. Research results have been validated by Industrial partners exercising both the techniques and the

supporting tools for developing application pilots in Automotive and Telecommunication domains. Additional development experiences have been done integrating new services from each industrial partner, in order to set up a common experimental setting with which to assess technical and business benefits. Research effort lead by academic partners has been extensively described in different papers [9,10,11,12,8,5,14,7,16]. This paper focus on the industrial experiences and specifically in the experiences and the results achieved developing advances services for SMEs. The organization of the paper is the following. After this introduction, section 2 gives a brief outline of the SECSE methodology and the supporting toolkit. Section 3 presents the evaluation approach describing the Logistic4SME pilot, the evaluation process and metrics. The evaluation results are in section 4, and finally the Conclusions and open issues are summarized in section 5

2 The SECSE methodology and toolkit

The Project has produced two principal outcomes: A methodology and supporting tools for developing service centric systems.

The SECSE methodology provides a conceptual model including terminology, concepts and relationships for service engineering, methods, processes, and techniques for developing service centric applications. The functional areas are depicted in figure 1.

- *Service-centric System Engineering* comprises the principal processes for building service-centric systems such as analysis, design, development, deployment and operation. SECSE provide specific subprocesses and supporting tools for design-time and runtime.
- *Service Engineering* addresses development practices aiming to deliver atomic services.
- *Service Acquisition and Provisioning* focus on process and tools for "service marketplace" actors such as service consumers, service providers and service brokers, to publish, negotiate, monitor, and deliver services.

The SeCSE suite aims to support service stakeholders to apply the methodology for building and managing service-centric systems. The toolkit covers the entire life cycle including capabilities to describe, compose, discover, monitor and deliver services.

1. *Service Discovery* tools provide specialized support for finding services and approximate orchestration starting from business level UML use cases and captured requirements. The SECSE Tools are Early Service Discovery Suite ESD, Architecture Service Discovering (ASD) and Runtime Service Discovery (RSD).
2. *Service Specification* tools enhance standard service specification and exposition formalisms by means of facets [7] incorporating semantic descriptions. The SECSE Tool is Facet Management Tool.

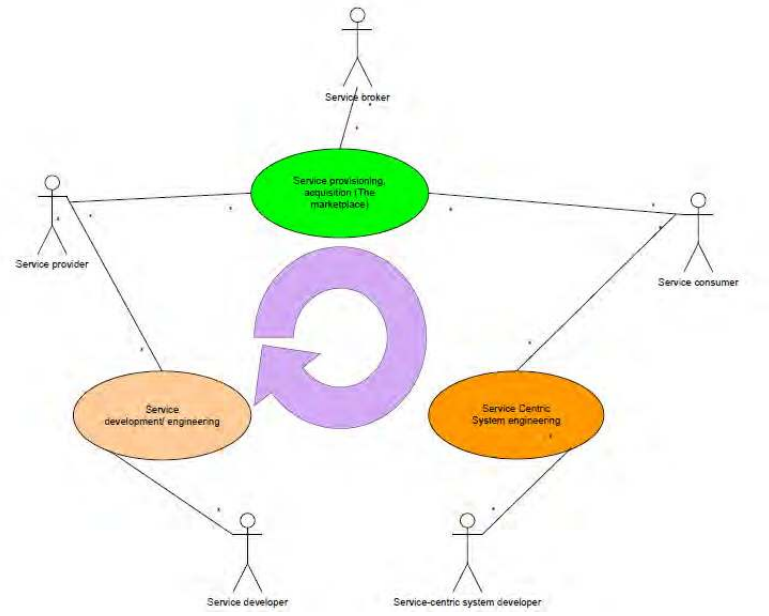

Fig. 1. SECSE Methodology (from [1])

3. *Service Publication* tools allow service registration and exposition. The SECSE Tool is SeCSE Registry.
4. *Service Testing* tools test services to build a confidence in the service's functional and non functional features. The SECSE Tools are Testing Facet Generator, Regression Testing Tool and Quality of Service Tool.
5. *Service Monitoring* aim to provide service-centric facilities for checking compliance with functional and non-functional requirements. The monitoring tools work both at process level (loose monitoring) and at single operation level (tight monitoring). The SECSE Tools are Tight Monitoring and Loose Monitoring.
6. *Service Composition* tools enhance standard BPEL[13] tools incorporating execution and monitoring rules. The SECSE Tool is Composition Designer.
7. *Service Deployment and Execution* tools execute enriched BPEL services by offering negotiation, binding and re-planning mechanisms as well as run-time service discovery. The SECSE Tools are Deployer and Runtime Execution engine.

3 A SECSE case study: Advanced Logistic Service for SMEs

The SECSE project has developed several case studies for assesment of the SECSE methodology and tools [2,3]. One of them is the a Business Support Sys-

tem for SMEs. The business goal of this system is to make available for SMEs through Web service technologies, sophisticated and expensive applications similar to those available in big companies such as CRM, EnterpriseWorkforce Management, accounting and billing.

A concrete case has been considered for a SME specialized in home repair, this includes every mishap bound to happen in your home, from a dripping tap to a blown pipe, to a malfunctioning heater, etc... This functionality might be adapted to other domains related to repair and maintenance.

The Logistics4SME service is delivered to SMEs as an integrated, flexible, scalable, and customizable package; for home repairing.

The system supports the following SME Actors: Secretary who manages phone calls from customers, creates incidence records, assigns priorities to tasks, and assigns tasks to technicians; Technician who gets his/her daily work plan, performs the tasks according to the schedule, accepts/rejects new tasks, sends task reports back to the company.

The system has been built by an iterative process of creating, selecting, adapting, validating, and assembling in-house components and commercial components. In-house components were initially based on the BOGAR [6] library which has proven their effectiveness for the development of several kinds of interactive services in Telefnica I+D. Using BOGARs components leads to a logical architecture structured into two layers: The control layer populated by control components, and the Resource layer made up of Application resources which offer interfaces to the controllers, an/or send information to them.

The pilot is made up of the following Controller components implemented with BPEL:

- **Access Controller** which manages user access and authentication. It uses the following resources:
  - *Access Visualization*: Implements access visualization through web interfaces
  - *Persistency Resource*: Provides persistency operations related to store, retrieve, and validate user access and authentication
- **Secretary Assistant** is responsible of the following activities:
  - *Customer management*: functionality related to the management of SME clients. It allows adding and removing customers from the database, modifying their personal data or searching among customers with varied criteria.
  - *Failure incidences*: functionality and support to create incidence records assign priorities to tasks and assign tasks to technicians.
  - *Technicians work flow*: functionality and support for Technician's workflow management. It helps the secretary to add new technicians to the database, manage their timetables or search which ones are working at a certain moment.

  The resources needed to achieve its activities are the following:
  - *Secretary Visualization*: Implements the secretary User Interface, providing high level operations for visualizing and managing application information, and user input.

  - *Enterprise Work Force Management*: Provides operations for finding the most suitable technician to achieve repair tasks taken into account the type of incident, technician, skills, user location, and technician work-plan. It also elaborates, manage and provide upon request technicians work-plan.
  - *Persistency Resource*: Provides persistency operations related to store, retrieve, and validate customers, incidences, and technicians.
- **Tecnician Assistant** is responsible of managing the daily work plan of each technician, informing about the failure incidences he has accepted (customer information, failure description, priorities). He/she can accept or reject proposed tasks, and send task reports back to the SME, after completing the reparation.
  The resources needed to achieve its activities are the *Enterprise Work Force Management Service* which provides operations to retrieve, and update technicians work-plan; the *Billing Service* which provides operations for elaborating customers billing; the *Spare Parts Service* which provide operations for searching and locating spare parts providers.

4 The evaluation process

The evaluation process has been based on the development of three versions of the pilot. It was achieved by tight cooperation among service providers, service integrators for service assembly, and service developers.

Logistics4SME_V1 was developed using open source development tools such as Eclipse and Active BPEL.The pilot implemented an initial set of functions for the secretary and the technician. Access and integration of key components such as the EWMS platform, and support for user mobility, were simulated.

Logistics4SME_V2 incorporates technicians mobility (PDA), integration of the EWMS platform and new Web Sercies for third parties such as User Location and Spare Parts Management.

Logistics4SME_V3 was a re-engineered version of Logistics4SME_V2 using the SECSE methodology and the SECSE toolkit.

The two scenarios have been used to define a collection of testing cases for validating the requirements on the SECSE tools.

The first scenario aims to assess application of the SECSE methodology for the development of a service component which will be integrated with other single and composite service for delivering the final service to the SME. The methodology Areas covered by this scenario are: Service Engineering, Service Acquisition and Provisioning. The tools used in this scenario are: Specification tools, Service Publication and Delivery tools, and Testing tools.

The second scenario assumes that the composite Web Service has been developed using conventional BPEL based tools. SECSE tools are used for enhancing the composite Web Service using the binding and negotiation mechanism and the monitoring rules. This scenario covers methodology activities in Service-centric Service Engineering and Service Acquisition and Provisioning. SECSE tools used

are: Service-centric Service Engineering tools, Service Discovery tools, and Testing tools.

The development of the first prototype led to the definition of an initial set of 345 requirements, and the metrics to measure the satisfaction of the requirements. Four types of metrics have been defined.

- M1: addresses the degree of the implementation. Values range from 1: not implemented, to 5 fully implemented.
- M2: measures the accuracy level – percentage of acceptable errors, with error levels defined/measured as aside. 1 is 0%, 2 is 25%, 3 is 50%, 4 is 75% and 5 is 100%.
- M3: estimates the quality of documentation : values range from 1 awful, 2 is bad, 3 medium, 4 good and 5 excellent.
- M4: estimates the quality of the user interfaces. Values range from 1 awful, 2 is bad, 3 medium, 4 good and 5 excellent.

The initial list of requirements has been refined during the second phase of the pilots obtaining a final set of 190 requirements which have been evaluated using the testing cases and the evaluation metrics.

The evaluation approach consisted on associating to each testing case the requirements covered, and the evaluation metrics for each requirement. Then executing the testing cases using the computing and communication infrastructure of the third prototype, and estimating the metric value according the result of the testing case.

5 Evaluation results

Overall evaluation has been done gathering the testing results by each partner, grouping metric scores by service engineering activity, and then calculating average values for each group. More detailed description of this process can be found in [4]. The results are summarized in the table 1 [3].

Global scores shows that values for tools supporting the principal activities of the methodology ranges from a minimum of 3,6 to a maximum 4,2. This may be interpreted as a score closed to good, according the metrics used. It should be noted that this score come from to the evaluation of functional aspects, consequently the results confirm that the acceptability of the tools delivered.

Score differences among partners are due to their pilots and testing cases, which cover specific requirements. In general, score under 3 are due to the partial coverage of required functionality in the tools, and usability under the acceptable levels. The lack of value in TID's service discovery row was due to the fact that the Early Service Discovery tool was not considered in the testing case. In addition, the functionality of Architecture Service Discovering Tools (ASD) and Runtime Service Discovering tool (RSD) couldn't be tested because they were

[3] ATOS (Atos Origin), CRF (Centro Ricerche Fiat), KD (KD Software), TID (Telefnica Investigacin y Desarrollo)

Table 1. Overall evaluation scores

Activity	ATOS	CRF	KD	TID
Service Specification & Testing	4,5	2,9	4,1	3,0
Service Discovery	4,8	3,1	3,7	
Service-centric System Engineering & Testing	3,8	4,1	2,0	3,6
Service Publication & Delivery	4,2	4,0	5,0	3,8
Integration	3,7	4,1	4,6	3,6
General requirements	4,4	2,7	4,1	4,0

not integrated in the SECSE run-time platform. Experiences on tool usage and suggestions for improvement identified by TID where the following:

Service Specification and Testing tools. The Facet Management Tool eases the process of describing the most significant characteristics of a web service. It provides an initial collection of service concepts and properties which are represented as facets. These facets covers functional aspects such as interfaces, signature and semantics, constraints and exceptions, quality of service aspects, and business aspects. Facet description is based on XML. This facilitates flexible formal definition and syntactic validation of facets, and also to attach semantic annotations which would allow the semantic validation of local properties and the overall service properties.

Improvements to meet industrialization requirements are: consistency checking and semantic validation of facets, and compatibility with existing standard, e.g. UDDI, in order to guarantee universal access and use of deployed services. Enhancement of user assistance for data definition, diagnostic and explanations when errors occurs, and pro-active support for error correction would also be necessary.

The two tools provided by the Regression Testing Toolkit: the Testing Facet Generator and the Regression Testing Tool, are suitable for industrial use. The Regression Testing Tool executes the testing cases associated to the service testing facet, which could be generated by the Testing Facet Generator tool. In practice, there is a strong need to maintain the consistency among the service code, the testing cases uploaded in the registry and the testing results obtained using the Regression Testing Tool. This could be achieved through the integration of both tools. To facilitate the service validation process there is a need for a single interface integrating the functionality for generating the testing cases, uploading them into the service testing facet, and then executing the testing cases using the service code.

Service Discovery tools. Three different tools for Service Discovery have been successfully tested independently. The Early Service Discovering tool ESD provides a wide range of facilities for expressing search criteria by combining

functional and non functional requirements. Industrial use of the ESD tool is strongly dependent on the availability of a large amount of registered services with clear and consistent specifications based on widely accepted standards.

The Architecture-time Service Discovery Tool (ASD) offers very interesting features but definition of search queries during the design phase requires the use of UML profiles created with proprietary UML tools.

Runtime Service Discovery Tool (RSD) offers dynamic and contextual queries during service execution. Industrial use of the RSD will be strongly dependent on fulfilling the highest quality standards for robustness, reliability and efficiency.

Service Publication and Delivery. Publication facilities are delivered as part of the SECSE registry. User functionality incorporate: web based interfaces for registration and management of Web Service descriptions using facets. Specific functions for uploading, browsing and deleting Web Service are available. Right now it is mandatory to use the publication facilities provided by the SECSE registry in order to use the overall SECSE environment. However it would be advantageous that publication tools would be decoupled from the registry. Overall the experience of using the SeCSE registry User Interface is satisfactory. Suggested improvement are: stability and robustness of the User Interface functionality still necessary, version control, and additional operation and User Interfaces to get the Service Access Points (SAP).

Federation capabilities work well, and are one of the most appealing features of the toolkit.

Service-centric Systems Engineering tools. This includes tools supporting service composition, service execution and service monitoring. The Composition Designer -CD- provides facilities to enhance standard BPEL tools incorporating innovative features such as dynamic binding and execution and monitoring rules. Its textual interface for defining the workflow logic and the binding rules might be improved to ease understandability and usability. In its present state this can be an obstacle for industrial use, due to the existence BPEL design tool which offer more sophisticated graphical interface

The Runtime Execution engine executes enriched BPEL compositions defined with the CD. This includes implementation of negotiation, binding and re-planning mechanisms as well as runtime service discovery.

Monitoring capabilities are fundamental in order to get quality control metrics, and if necessary undertake corrective actions. The monitoring tools have been tested in isolation providing satisfactory results; however integrated testing using the SECSE execution environment couldnt be done. Successful integration of the CD, the run time engine, and the monitoring tools to fulfill industrial requirements regarding robustness, stability, reliability and efficiency still an open issue. Although its resolution would require more effort than planned, the business advantages in terms of cost-benefit would be largely positive.

6 Conclusions and future challenges

The SeCSE project is one the leading EU funded IPs which has assumed the challenge to find out innovative technologies, and supporting tools for building service centric systems. Technical and business evaluation of both the SECSE methodology and the supporting toolkit has been done exercising them developing application pilots by industrial partners involved in the project.

The evaluation conclusions which were drown from the experiences of developing advanced logistics service for SMES are detailed below.

- Service developers found the SECSE methodology easy to understand. The conceptual model, and description of functional areas, processes, tasks, and work-products, allows the development team to have a common view of the development process, then facilitating sharing of knowledge and improving team communication. The methodology provides advice and guidance for accomplishing tasks using the SeCSE Tools. It supports rapid and incremental development including specific aid for the integration of legacy systems and heterogeneous components.
- The SECSE toolkit gives integrated functionality and support for achieving key service engineering activities such as service specification, publication, delivery, composition design, service execution, and monitoring, and testing. Tool integration is key to optimizing the development process, lowering costs and speeding up service provisioning. Thus using SECSE tools increase business efficiency reducing service development and maintenance costs, decreasing the time to market of new services, and facilitating extensibility, and envolvability.
- Evaluation conclusions based on the testing cases have also demonstrated business viability and possible industrialization of tools supporting service specification publication and delivery. For the rest of the tools, evaluation of the prototypes has shown the potential of key disruptive features such as the specification and execution of service monitoring properties, negotiation, dynamic binding and re-planning mechanisms, as well as run-time service discovery. Although this functionality has been implemented according to industrial needs, its assimilation into service engineering practices would require the tools to satisfy the industry standards of usability, user interface, documentation, stability and efficiency. Tool installation and interoperability issues should also be considered, as they were a serious obstacle for running the tools and the testing cases.

Overall industrial evaluation has confirmed the feasibility and the potential of the methodology and the supporting tools. Future challenges for persistency, acceptance and evolution of the SECSE work are: influencing standards bodies to recognize service specifications enhancement; demonstrating robustness, scalability, interoperability and usability of architecture and computing models designed and implemented in the project. Delivering the project implementations under open source would facilitate overcoming this issues which require an

active participation in standard forums, and spending additional development effort to upgrade the prototypes into stable tools.

References

1. A5.D4.2 SeCSE Methodology V1.3 .(http://secse.eng.it).
2. A6.D11 - CRF Automotive pilot development architecture (v2) (http://secse.eng.it).
3. A6.D12 - T-A Telecommunication pilot development architecture (V2) (http://secse.eng.it).
4. A6.D16 Testing and evaluation of web services for pilots.
5. F.J. Nieto, L. Bastida, M. Escalante, A. Gortazar.: "Development of Dynamic Composed Services based on the Context". Proceedings of the 2nf International Conference Interoperability for Enterprise Software and Applications (I-ESA 2006), Bordeaux, France, March 2006.
6. Garijo, F J., Bravo S., Gonzalez, J. Bobadilla E. BOGAR_LN: "An Agent Based Component Framework for Developing Multi-modal Services using Natural Language". L.N.A. I, Vol. 3040. pp 207-. Springer-Verlag. 2004.
7. J. Walkerdine, J. Hutchinson, P. Sawyer, G—. Dobson, V. Onditi : "A Faceted Approach to Service Specification", Proc. Second International Conference on Internet and Web Applications and Services (ICIW07), Mauritius, May 2007. IEEE Computer Society Press.
8. Kozlenkov A., Spanoudakis G., Zisman A., Fasoulas V., Sanchez Cid F.: "Architecture-driven Service Discovery for Service Centric Systems" Journal of Web Services Research, 4(2), pp. 81-112, 2007.
9. L. Baresi and S. Guinea. "Towards Dynamic Monitoring of WS-BPEL Processes", Proceedings of the 3rd International Conference of Service-oriented Computing (ICSOC'05). Amsterdam, The Nederlands, 2005. Lecture Notes in Computer Science, Vol. 3826, pp. 269-282.
10. M. Colombo, E. Di Nitto, M. Mauri: SCENE: A Service Composition Execution Environment Supporting Dynamic Changes Disciplined Through Rules. ICSOC 2006: 191-202.
11. M. Di Penta, R. Esposito, M.L. Villani, Roberto Codato, M. Colombo, and E. Di Nitto. WS-Binder: "A Framework to enable Dynamic Binding of Composite Web Services". In Proc. International Workshop on Service Oriented Software Engineering (IW-SOSE'06), pages 77-80, Shanghai, China, May. 2006. ACM.
12. M. Di Penta, G. Canfora, M. Bruno, G. Esposito, V. Mazza – "Search Based Testing of Service Level Agreements" to appear in proceedings of the Genetic and Computation Conference (GECCO 2007), July 2007, London, UK, ACM Press.
13. OASIS Web Services Business Process Execution Language Version 2.0 Committee Draft, 25 January 2007.
14. Spanoudakis G., Mahbub K., Zisman A.: "A Platform for Context Aware Runtime Web Service Discovery", IEEE 2007 International Conference on Web Services (ICWS 2007), Salt Lake City, Utah, USA.
15. UDDI v3.0, OASIS Standard, February 2005, http://www.uddi.org
16. Zisman A. Mahbub K., Spanoudakis G.: "A Service Discovery Framework based on Linear Composition", IEEE 2007 International Service Computing Conference, Salt Lake City, Utah, USA.