Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

UNA APROXIMACIÓN PARA LA SIMPLIFICACIÓN DEL DESARROLLO DE APLICACIONES WEB SEMÁNTICAS Y EL USO DE DATOS SEMÁNTICOS

TESIS DOCTORAL
MAYO 2009

MARIANO RICO ALMODÓVAR
DIRECTORES: DAVID CAMACHO FERNÁNDEZ
ÓSCAR CORCHO GARCÍA

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Autónoma de Madrid, el día ..... de ..... de 2009.

Presidente: D. .....

Vocal: D. .....

Vocal: D. .....

Vocal: D. .....

Secretario: D. .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2009 en .....

Calificación: .....

EL PRESIDENTE EL SECRETARIO

LOS VOCALES

A mi más bella historia de amor: Ángela Lucía y José Soyuz

Agradecimientos

Cómo no, a mi padre y a mi madre, que hicieron todo lo posible porque "el mayor" estudiase una carrera, pese a que tenía la costumbre de "empezar la casa por el tejado".

Agradecer también a mi "núcleo duro" de compañeros, y sin embargo amigos, los buenos ratos pasados y vuestro apoyo en los malos momentos. Ya sabéis quiénes sois, no hace falta que os nombre. Os agradezco la dedicación, el tiempo, y las sonrisas, en momentos en que todo esto escaseaba.

Mención especial para Alejandro, que me convenció para que dejase la empresa privada y me quitase la espinita de la docencia y la investigación. Pese a que se han trastocado mucho los planes iniciales, sigo estando contento por el paso que me ayudó a dar. También fue responsable de ese salto al vacío Xavier, con el que me siento en deuda por su apoyo, honestidad y cercanía.

De mi etapa en el grupo NETS, quiero agradecer a Pablo la confianza que depositó en mí, y que me hablase por primera vez de "algo llamado" Web Semántica. A los miembros del grupo debo agradecerles su apoyo y los buenos momentos pasados juntos.

A José Antonio quiero agradecerle lo que me enseñó de HCI y su paciencia conmigo.

En cuanto a mis co-directores, agradecerles lo mucho que he aprendido de ellos en lo técnico, pero también (si no más) en lo humano. Ha sido un placer trabajar con vosotros, y espero que esta etapa que termina ahora sea el principio de una larga y próspera relación profesional y personal.

A la UAM quiero agradecerle sus bolsas de viaje, su ERASMUS Staff Mobility Program, su Escuela Infantil, su Programa de Formación, y su Servicio de Deporte.

Y para terminar, a los miembros del B-408 (Luis, Javi, Víctor, Fabri, Ana *et al.*), agradecerles las risas, las miradas cómplices, y los intercambios casuales de conocimientos. Pese a la alta tasa de personas y máquinas por metro cuadrado, el nivel de ruido siempre fue increíblemente bajo. Ha sido un placer y un honor compartir espacio vital con vosotros.

Este documento ha sido creado con LaTeX 2_ε usando el estilo de la universidad de Standford, por lo que me siento en deuda con Donald E. Knuth (creador de TeX), Leslie Lamport (creador de LaTeX) y Joseph Pallas (creador del estilo suthesis-2e.sty). Gracias a todos ellos, y tantas otras personas que donan altruistamente su trabajo.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción y motivación

1.1. La Web Semántica

Aunque existieron esfuerzos en los años 90 encaminados a la anotación basada en ontologías de recursos Web como (KA)[2] [Staab *et al.*, 2000] o SHOE [Heflin y Hendler, 2001], el término *Web Semántica* se hizo popular a partir de la publicación en 2001 de un artículo de Berners-Lee *et al.* en la revista de divulgación científica Scientific American [Berners-Lee *et al.*, 2001]. En este artículo, el creador de la Web describe cómo ésta puede proporcionar contenidos dirigidos no sólo a humanos, sino también a agentes software capaces de *entender* el significado de contenidos específicamente diseñados para ellos así como su manipulación.

Los dos pilares básicos para creación y la publicación de estos contenidos son las ontologías y las anotaciones. La primera versión de las especificaciones de lenguajes de representación de ontologías y anotaciones en este entorno data de 1997 [1]. Desde entonces, algunas de ellas se han convertido en recomendaciones del World Wide Web Consortium (W3C). Entre ellas cabe destacar RDF [Klyne y Carroll, 2004], RDFS [Brickley y Guha, 2004], OWL [Bechhofer *et al.*, 2004], y SPARQL [Prud'hommeaux y Seaborne, 2008]. RDF, RDFS y OWL son recomendaciones W3C desde 2004, y SPARQL lo es desde 2008.

Entre los beneficios citados en [Berners-Lee *et al.*, 2001] que puede aportar la Web Semántica destacan:

- Integración de información. Es la base fundacional de la Web Semántica. Supongamos dos empresas con información almacenada en bases de datos heterogéneas. Actualmente no se puede crear fácilmente un nueva base de datos que integre de forma automática los esquemas y la información de dichas bases de datos. Las tecnologías de la Web Semántica facilitan esta tarea, permitiendo unir los esquemas (ontologías) y los datos, y añadir información adicional acerca de equivalencias entre términos (a nivel de clase, propiedad e instancia).

- Recuperación de información. Por ejemplo, si se busca en Google "libros en los que se mencionan libros de García Márquez" no se obtiene el resultado adecuado. Añadir información semántica al texto de la consulta y a los documentos indexados produce unos resultados más precisos que la indexación de texto tradicional [Castells *et al.*, 2007; Guha *et al.*, 2003; Kiryakov *et al.*, 2004].

[1] Véase `http://www.w3.org/TR/WD-rdf-syntax-971002/`

- Delegación de tareas. Los agentes semánticos personalizados serán nuestros "mayordomos" en la Web, a los que se encargarán tareas que requieran la interacción con otros agentes, con otras personas, o con empresas.

En el contexto de la Web Semántica, también se trabaja en la descripción semántica de los servicios web con el objetivo de facilitar las tareas de descubrimiento y composición de los mismos para la construcción de aplicaciones complejas. El W3C tiene como candidatos a recomendación a WSMO [Lausen *et al.*, 2005b] y OWL-S [Martin *et al.*, 2005]. Sin embargo, esto queda fuera del ámbito de este trabajo.

1.2. Terminología utilizada

Dada la ambigüedad de algunos de los términos utilizados en la amplia bibliografía existente relativa a la Web Semántica, a continuación se describe la interpretación de los mismos que será utilizada en este trabajo.

- Un *agente* software se puede definir como un programa capaz de gestionar información electrónica. Un agente web es aquel capaz de manejar datos de la Web. Aunque los denominados agentes inteligentes o autónomos son más generales [Wooldridge y Jennings, 1995], algunas de sus características fundamentales, como aprendizaje o proactividad, no son consideradas, por lo que la acepción de agente que se utiliza en este documento se puede considerar una versión simplificada de los mismos.

- Información *semántica* se refiere a información que puede ser interpretada con mínima ambigüedad por los programas. Para prevenir dichas ambigüedades los estándares de la Web Semántica se sustentan en sólidos formalismos lógicos [Muñoz *et al.*, 2007]. Por ejemplo, OWL permite a los razonadores garantizar cierto grado de computabilidad en las operaciones de inferencia siempre que se den un conjunto de condiciones bien precisas. Una *aplicación semántica* será aquella que publique información semántica. La publicación consistirá en situar la información semántica en una URL a disposición de los agentes semánticos, o en algún repositorio que permita ser consultado.

- Una *ontología* es una especificación formal y explícita de una conceptualización

compartida [Studer *et al.*, 1998]. A modo de ejemplo, una de las ontologías más usadas, FOAF (Friend Of a Friend)[2] define, entre otras, las clases *Person* y *Organization*; las propiedades *name*, *surname* y *email*; y la relación *knows* (aplicable a individuos de la clase *Person*). Aunque, formalmente, una ontología agrupa tanto a definiciones (clases, propiedades y relaciones) como a individuos (instancias de clases), se usará el término "ontología" para referirnos sólo a las definiciones, reservando el término "datos semánticos" (o simplemente "datos") para los individuos. Ontologías y datos se identifican con un espacio de nombres (namespace) [3], y aunque no es obligatorio que correspondan a una URL, es práctica común que exista esta superposición.

- Las *tecnologías de la Web Semántica* se refieren a los estándares de la Web Semántica citados anteriormente (RDF, RDFS, OWL, y SPARQL) así como a las herramientas que permiten crear y manipular ontologías y datos semánticos.

- Un *usuario final* es una persona con escasos conocimientos técnicos o entrenamiento previo en el uso de una aplicación software dada. Conviene distinguir esta definición de otras como [Nardi, 1993; Gilb y Finzi, 1988] que consideran al usuario final como el elemento final de la cadena de desarrollo de software, esto es, el usuario al que se dirige una aplicación que posea un interfaz adaptado a sus necesidades, que ha sido entrenado para usar la aplicación desarrollada, y que suele tener conocimientos previos del dominio de aplicación de la herramienta informática.

- Un *diseñador web* es una persona especializada en diseño gráfico para aplicaciones web. Este tipo de profesionales se caracteriza por disponer además de conocimientos técnicos en tecnologías web cliente como HTML, CSS, Javascript, etc.

- Un *desarrollador web* es una persona con conocimientos técnicos en tecnologías web, especialmente en la parte servidora (e.g. PHP, JSP, .NET, Ruby, etc.), y típicamente también de la parte web cliente.

[2] Véase `http://www.foaf-project.org`

[3] En realidad se trata de una referencia a una URI (Uniform Resource Identifier). Véase `http://www.w3.org.TR/rdf-concepts/#dfn-URI-reference`

1.3. Acercando la Web Semántica a desarrolladores y usuarios

El éxito de la Web Semántica está relacionado con dos factores. El primero es la existencia de información semántica en la Web, y el segundo es la utilización que finalmente se haga de la misma.

La información semántica comprende ontologías y datos semánticos que se refieren a dichas ontologías. En 2005 se estimaba que había 11.000 ontologías disponibles en Internet [Warren y Baker, 2005] [Ding *et al.*, 2004], y en 2007 alrededor de 23.000 [d'Aquin *et al.*, 2007]. A su vez, los datos semánticos han experimentado un crecimiento exponencial durante los últimos diez años [Finin y Ding, 2006]. En octubre de 2007 había más de 2.000 millones de tripletas RDF relacionadas entre sí por unos 3 millones de enlaces RDF [4].

Pese a su rápida expansión, aún queda lejos de la velocidad de expansión que tuvo la Web. Por ejemplo, el primer servidor Web se puso en funcionamiento en agosto de 1991, y dos años después se creó la primera especificación de HTML (IETF, junio de 1993) y apareció el primer navegador (Mosaic, en abril de 1993). Si trasladamos los 12 años que han pasado desde la aparición de la primera especificación de RDF, debemos comparar la situación actual de la Web Semántica con la de la Web en 2005. Entonces la Web tenía 11.500 millones de páginas web [Gulli y Signorini, 2005], y lo que es más importante, era mucho más popular que la Web Semántica a fecha de hoy.

Uno de los motivos que hacen que esta información de alta calidad permanezca oculta para la mayor parte de los usuarios finales es que hay muy pocas herramientas que permitan a dichos usuarios tratar con datos semánticos. Como señala García-Castro *et al.* [García-Castro *et al.*, 2008]: *El desarrollo de aplicaciones web semánticas es una tarea difícil para gente que no pertenece a la comunidad científica*.

En esta tesis se considera que el desarrollo de estas aplicaciones supone un gran reto para los desarrolladores de aplicaciones debido a la convergencia de dos factores principales: (1) la creciente dificultad para crear aplicaciones web, y (2) la complejidad de las tecnologías de la Web Semántica. A continuación se detallan cada uno de estos factores.

[4] Véase `http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`

1.3.1. Complejidad creciente en la creación de aplicaciones Web

A diferencia de la creación de aplicaciones locales, crear software que explote la Web requiere dominar no sólo un lenguaje para la parte servidora (Java, .NET, Ruby o PHP, entre otros) y una tecnología web (ASP, JSP, JSF, RubyOnRails), sino conocimientos adicionales para la parte cliente (HTML, Javascript, CSS, DHTML, o AJAX). El lenguaje de programación y la tecnología web se puede elegir en función de las preferencias del desarrollador, en la mayor parte de los casos, especialmente cuando se van a realizar prototipos. No sucede lo mismo con las tecnologías web cliente, que no han ofrecido alternativas sino un mayor número de lenguajes de complejidad creciente. Esto ha convertido a los diseñadores web en programadores, como apunta Rochen *et al.* [Rochen *et al.*, 2006], lo que ha resultado en una reducción del número de personas cualificadas para crear sitios web modernos (por ejemplo, los llamados Web 2.0).

La parte cliente es con diferencia la menos robusta, ya que se dispone de una gran variedad de navegadores, cada uno de los cuales implementa de distinta manera algunos detalles de las especificaciones de las tecnologías cliente. Esto ha provocado que las aplicaciones web se diseñen, en el mejor de los casos, únicamente para los navegadores "más utilizados", y que la "compatibilidad" entre aplicaciones web y navegadores siga siendo un problema abierto [Koechley, 2006]. Aunque algunos entornos de desarrollo proporcionan a los desarrolladores facilidades gráficas dirigidas a minimizar el esfuerzo de desarrollo y la máxima compatibilidad, la mayor parte de ellos son productos comerciales con un precio elevado (e.g. BEA WebLogic). Notables excepciones son Ruby on Rails [5], Strips [6] o GWT [7] (Google Web Toolkit), ya que proporcionan a los desarrolladores poderosos entornos gratuitos con los que crear aplicaciones web.

1.3.2. Complejidad de las tecnologías de la Web Semántica

Por otra parte, las tecnologías de la Web Semántica se han centrado en proporcionar a los estándares tecnológicos citados una descripción lógica precisa que los haga utilizables por agentes. En particular, la creación de ontologías usando los lenguajes RDFS o OWL requiere conocimientos muy especializados [Muñoz *et al.*, 2007; Horridge *et al.*, 2007] sobre representación de conocimiento (en el caso de OWL en lógica descriptiva). Por ejemplo,

[5] Véase `http://www.rubyonrails.org/`
[6] Véase `http://www.stripesframework.org/display/stripes/Home`
[7] Véase `http://code.google.com/webtoolkit`

una ontología OWL se puede ver como un conjunto de axiomas, con complejas reglas de inferencia que pueden llevar a la aparición de relaciones inesperadas o a inconsistencias. Sin necesidad de inferencia, conceptos tales como *dominio*, *rango*, *tipo*, *propiedad funcional*, *propiedad funcional inversa*, o especialización de propiedades, no son sencillos de entender, y requieren ser claramente diferenciados de conceptos homónimos, como es el caso de los conceptos de *clase*, *propiedad* (*atributo*), e *instancia* de la Programación Orienta a Objetos.

Por ejemplo, en [Oren *et al.*, 2007b] se muestran las diferencias más destacadas entre RDFS y lenguajes de programación orientados a objetos (POO) como Java o C#. Aunque en la siguiente lista se hace referencia al lenguaje Java, se debe entender "la mayor parte de los lenguajes de POO".

- Pertenencia a clase: en Java, un objeto es miembro de una única clase, fijada en la instanciación. En RDFS, un recurso puede pertenecer a múltiples clases, su pertenencia a una clase no es fija, sino que se define mediante su **rdf:type** y las propiedades que pertenecen al recurso.

- Herencia de clases: en Java, las clases pueden heredar como máximo de una clase. En RDFS, las clases pueden heredar de varias clases (incluso ciclos de herencia).

- Atributos vs. propiedades: en Java, los atributos son locales a la clase y sólo pueden ser usados por las instancias de la clase. En RDF, las propiedades son entidades independientes que pueden ser usadas por cualquier recurso de cualquier clase y que pueden tener valores de tipos diferentes.

- Herencia estructural: en Java, los objetos heredan sus atributos de la clase padre. En RDFS, como las propiedades no pertenecen a las clases, las propiedades no se heredan. Por el contrario, el dominio de las propiedades sí se propaga pero, como indican la pertenencia de clase de los recursos que las usan, los dominios se propagan en la dirección contraria a la esperada a lo largo de la jerarquía de clases.

- Composición de las instancias: en Java, la estructura de las instancias sigue exactamente la definición de la clase. En RDFS, la definición de una clase no es completa, y no limita la estructura de sus instancias, de forma que, cualquier recurso RDF puede usar cualquier propiedad.

- Flexibilidad: en Java no se permite que la definición de una clase evolucione en el tiempo. RDF está diseñado para que ontologías y datos evolucionen en el tiempo.

Por tanto, el proceso de creación de ontologías, en teoría llevado a cabo por expertos del dominio, usualmente requiere especialistas en lenguajes de ontologías a fin de traducir, de manera precisa, el conocimiento de los expertos del dominio a una ontología concreta. A pesar de los esfuerzos realizados para crear herramientas gráficas (por ejemplo, Protege[8] o Swoop[9]) que asistan a los expertos del dominio en el proceso de creación de ontologías, esta tarea no puede ser llevada a cabo todavía de una manera intuitiva.

1.4. Interacción con la Web Semántica

La interacción entre personas y la Web Semántica se encuentra limitada por las dificultades técnicas para crear aplicaciones, descritas en las secciones anteriores. Por ejemplo, una de las primeras aproximaciones exitosas para la navegación por la Web Semántica, una aplicación web (un "navegador de la Web Semántica") llamada Tabulator [Berners-Lee *et al.*, 2006] adolece de los problemas de "compatibilidad" citados anteriormente ya que sólo puede ser usada con el navegador Firefox.

Aunque se reduzcan los problemas descritos y se logre facilitar la creación de aplicaciones web semánticas que utilicen datos semánticos, queda abierto el problema de cómo los usuarios y desarrolladores deben interaccionar con esa información de manera eficiente. La dificultad para tratar con ontologías y datos semánticos, y en general con los nuevos conceptos introducidos por la Web Semántica, desde la desaparición del concepto de página web a la mezcla de fuentes heterogéneas de datos, ha planteado nuevos retos a los expertos en Interacción Persona-Ordenador [Schraefel *et al.*, 2008]:

- ¿Cómo puede buscar un usuario un conjunto de información relacionada?
- ¿Cuál es la mejor forma de presentar dicha información para su exploración y su uso?
- ¿Cuál es la manera más consistente de presentar fuentes de datos heterogéneas?
- ¿Cómo debe una interfaz de usuario revelar las posibles conexiones entre un dato y otros muchos?

[8] Véase `http://protege.stanford.edu/`
[9] Véase `http://www.mindswap.org/2004/SWOOP/`

- Siendo capaces de identificar la fuente de todo dato, ¿cómo podemos hacer útil y usable su procedencia dentro de la interfaz?
- Si las relaciones y la inferencia se usan sin que el usuario lo sepa, ¿cómo se puede hacer para que las acciones y las suposiciones se vuelvan transparentes sin incrementar la carga cognitiva de los usuarios?.

En esta línea se pueden encontrar trabajos donde se han aplicado técnicas de Interacción Persona-Ordenador (IPO) para permitir a los usuarios mejorar la navegación por la Web aprovechando la información semántica; creando, por ejemplo, los llamados "navegadores web semánticos" (Semantic Web-browsers) [10], así como nuevos conceptos de interacción entre humanos y la Web Semántica como la "navegación facetada" [Lima y Schwabe, 2003a].

Un primer paso en esta dirección es facilitar la presentación de datos semánticos, así como la creación de interfaces de usuario capaces de generar datos semánticos a partir de la información introducida por el usuario. La importancia de proporcionar una interfaz de usuario queda patente en estas citas:

> "*After more than 10 years of work into various aspects of the Semantic Web... I am now fully convinced (read: no longer in denial) that most of the remaining challenges to realize the Semantic Web vision have nothing to do with the underlying technologies... Instead, it all comes down to user interfaces and usability*".
>
> Ora Lassila, "Semantic Web Soul Searching", 19 de marzo de 2007 [11].

> "*Creating interfaces for interacting with this data is difficult because there is no well-defined web "page" or homogeneous structure to the data. Creating data is at least as important as presenting it, so facilitating efficient and effective data entry is an important aspect of user interaction*".
>
> CHI 2008 [12].

[10] No confundir con los "navegadores de la Web Semántica" (Semantic-Web browsers), que no aprovechan la información semántica, limitándose a mostrar los datos de la Web Semántica siguiendo una navegación tradicional.

[11] Véase `http://www.lassila.org/blog/archive/2007/03/semantic_web_so_1.html`

[12] ACM Conference on Human Factors in Computing Systems. Semantic Web User Interaction. Véase `http://swui.semanticweb.org/SWUI2008CHI/`

Las dificultades se incrementan aún más cuando esperamos que estas herramientas se adapten al dispositivo usado por el usuario (PC, PDA, móvil, TV, con todas sus variantes de marcas y modelos), las limitaciones interactivas del usuario (daltonismo, reducida agudeza visual), o sus preferencias estéticas. Aunque esto puede parecer un requisito superfluo, los agentes semánticos deben ser capaces de resolver este problema ya que, en la visión de la Web Semántica, los agentes semánticos especializados en interaccionar con humanos [Rico *et al.*, 2009] son los encargados de crear un interfaz de usuario adecuado, descargando de esta tarea a los productores de la información semántica.

1.5. Conclusión

Aunque la Web Semántica dispone de un conjunto razonable de estándares tecnológicos, ontologías, y datos semánticos, aún no se ha popularizado. En este capítulo se ha mostrado la necesidad de acercar dos perfiles de usuario con conocimientos muy diferentes. De una parte se encuentran los desarrolladores de aplicaciones web y de otra se encuentran los expertos en tecnologías de la Web Semántica. En este trabajo se constata que para un desarrollador de aplicaciones web es muy difícil usar datos semánticos. Asimismo, para un experto en tecnologías de la Web Semántica es muy difícil crear aplicaciones web atractivas, robustas, y que cumplan unos requisitos básicos de calidad. El objetivo de esta tesis es proporcionar a ambas partes mecanismos que permitan cubrir la brecha que separa a estos perfiles.

1.6. Estructura y contenidos de la tesis

El siguiente capítulo muestra el estado del arte en las áreas en las que esta tesis realiza contribuciones, esencialmente el desarrollo de aplicaciones web semánticas por parte de dos tipos de desarrolladores: (1) los desarrolladores con conocimientos de las tecnologías de la Web Semántica pero sin grandes conocimientos de las tecnologías de desarrollo de aplicaciones Web, y (2) los desarrolladores con experiencia en aplicaciones Web pero sin conocimientos de las tecnologías de la Web Semántica. También se muestran las herramientas disponibles para desarrolladores y usuarios finales.

El capítulo 3 muestra las hipótesis de trabajo y los objetivos de la tesis. En particular se detallan las limitaciones actuales en cuanto al desarrollo de aplicaciones web semánticas,

las condiciones bajo las que se pueden aplicar las propuestas de la tesis y las soluciones aportadas en esta tesis.

Las contribuciones principales de la tesis se detallan en los capítulos 4 y 5. El capítulo 4 describe una infraestructura orientada a desarrolladores con conocimientos de las tecnologías de la Web Semántica que permite crear de forma sencilla aplicaciones web semánticas sin requerir conocimientos de las tecnologías web. El capítulo 5 describe un conjunto de aplicaciones dirigidas a diseñadores web que permiten crear elementos gráficos para manejar datos semánticos que no requieren conocimientos de tecnologías de la Web Semántica. Estos elementos gráficos pueden ser usados fácilmente por desarrolladores web para incorporar datos semánticos a sus aplicaciones, e incluso por usuarios finales.

El capítulo 6 describe los experimentos realizados para probar que las propuestas de esta tesis cumplen los objetivos marcados. En concreto, se detallan las evaluaciones experimentales llevadas a cabo con desarrolladores de aplicaciones, expertos en usabilidad, y diseñadores web con diferentes niveles de conocimiento de las tecnologías web cliente.

Para finalizar, el capítulo 7 muestra las conclusiones de la tesis doctoral y algunas de las posibles líneas de trabajo futuro.

Capítulo 2

Estado del arte

En este capítulo se muestra el estado del arte en aquellas áreas directamente relacionadas con el trabajo realizado en esta tesis doctoral. En concreto, la creación de aplicaciones web semánticas por parte de desarrolladores con distintas competencias, y el uso de estas aplicaciones por parte de los usuarios finales. Cada uno de estos perfiles requiere unos niveles mínimos de competencias que serán analizados. Los desarrolladores requieren plataformas sobre las que construir de manera sencilla aplicaciones semánticas, y los usuarios finales requieren aplicaciones con unos valores razonables de usabilidad y de calidad de la interfaz de usuario.

2.1. Aplicaciones Web Semánticas

En esta sección se intenta acotar el significado del término "Aplicación Web Semántica", con distintos criterios para caracterizar a este tipo de aplicaciones. También se muestra una clasificación no exhaustiva de los tipos de aplicaciones web semánticas existentes desde la perspectiva de los tipos de usuarios a los que van dirigidas, atendiendo a su nivel de conocimientos: expertos de la Web Semántica, usuarios especializados, y usuarios finales.

2.1.1. Definición

A falta de una definición comúnmente aceptada, en este trabajo se considera que una aplicación web semántica es **toda aplicación web que utiliza datos semánticos**. Este uso de datos puede ser externo (lectura y/o escritura de datos semánticos) y/o interno. Un ejemplo de uso externo de sólo lectura puede ser un navegador de la web semántica, en el que la navegación consiste en saltar por las relaciones entre los datos semánticos, mostrándolos al usuario. Un ejemplo de uso externo con lectura y escritura puede ser un portal web semántico, que permite la visualización y edición de información basada en ontologías. Y un ejemplo de uso interno puede ser una aplicación que use razonadores para inferir nuevos datos semánticos o comprobar la consistencia de la información semántica.

A continuación se muestran dos criterios que se pueden encontrar en el estado del arte acerca de los requisitos que deben cumplir una aplicación web para que sea considerada semántica.

Criterio del Semantic Web Challenge

El Semantic Web Challenge [1] es un concurso anual que busca mostrar las ventajas de las tecnologías de la Web Semántica. Los participantes deben presentar aplicaciones que cumplan unos requisitos básicos y, a ser posible, un conjunto de recomendaciones deseables. Estos requisitos se muestran en la tabla 2.1, y se pueden considerar característicos de lo que debe ser una aplicación web semántica.

Criterio de las Aplicaciones Web Semánticas de nueva generación

Motta y Sabou [Motta y Sabou, 2006] diferencian entre aplicaciones web semánticas que siguen fielmente el paradigma de la Web Semántica de aquellas que son equivalentes a sistemas tradicionales basados en el conocimiento. Estas últimas son sistemas semánticos "cerrados" en el sentido de que típicamente usan sólo una ontología para agregar datos en un dominio específico. Las primeras son sistemas semánticos "abiertos" en el sentido de que típicamente son heterogéneas en cuanto a la caracterización ontológica y la procedencia de los datos semánticos que manejan. La tabla 2.2 muestra sus características más relevantes.

A la vista de estos criterios queda claro que conviene detallar la definición inicial de **utilizar datos semánticos**. Para cumplir los criterios básicos del Semantic Web Challenge ese uso debe generar información difícilmente consegible mediante otras tecnologías (criterio B1), así como proporcionar información útil (criterio B3) a los expertos del dominio. El criterio B2 obliga a que se usen diversas fuentes de datos, y el criterio B4 es un requisito técnico que compete a los razonadores utilizados. Para cumplir los criterios de Motta y Sabou 2006, los criterios M1 y M3 obligan a usar múltiples fuentes de datos semánticos (equivalente al criterio B2), el criterio M2 obliga a usar internamente múltiples ontologías, el criterio M4 obliga a arquitecturas que permitan aplicaciones escalables (equivalente a D3), el criterio M5 debe permitir aprovechar recursos no semánticos, el criterio M6 indica cierto estilo de aplicaciones (equivalente a D1), el criterio M7 hace que la aplicación no sólo interaccione con usuarios humanos, sino también con agentes software.

[1] Véase `http://challenge.semanticweb.org/`

Tabla 2.1: Requisitos del Semantic Web Challenge

Requisitos **básicos**	
B1	El significado de los datos debe jugar un papel central. El significado debe ser representado usando descripciones formales. Los datos deben ser manipulados o procesados de forma que se genere información útil que sea más difícil de lograr (o imposible) con otras tecnologías.
B2	Con respecto a las fuentes de datos usadas, deben ser de distintos propietarios (es decir, sin control de la evolución de los datos), deben ser heterogéneas (sintácticamente, estructuralmente, y semánticamente),y deben ser datos reales (no se aceptan pequeños conjuntos de datos de prueba).
B3	La aplicación debe proporcionar a los expertos del dominio algún valor añadido.
B4	Las aplicaciones deben asumir un mundo abierto (es decir, la información no es nunca completa)
Requisitos **deseables**	
D1	La aplicación web proporciona una interfaz de usuario atractiva y funcional.
D2	Evaluaciones rigurosas que demuestren los beneficios de la tecnologías de la Web Semántica, o validen los resultados obtenidos.
D3	La aplicación debe ser escalable en términos de la cantidad de datos usados y de los componentes distribuidos utilizados.
D4	Novedosa en la aplicación de las tecnologías semánticas a un dominio o tarea que no había sido considerado antes.
D5	La funcionalidad difiere, o va más allá, de la simple recuperación de información.
D6	La aplicación tiene un potencial comercial claro y/o una clara base en el usuario.
D7	La información de contexto se usa para ordenaciones o evaluaciones
D8	Se usan documentos multimedia de alguna forma.
D9	Se usan datos dinámicos (por ejemplo, workflows), quizás en combinación de información estática.
D10	El resultado debe ser tan preciso como sea posible (por ejemplo, una ordenación de resultados conforme al contexto).
D11	Soporte para múltiples lenguajes y accesibilidad para un rango de dispositivos.

Tabla 2.2: Criterio Motta y Sabou 2006

ID	AWS de nueva generación
M1	Generación de datos semánticos (evitar autoconsumo)
M2	Manejo de múltiples ontologías
M3	Abierto a nuevas fuentes de datos
M4	Capaz de manejar grandes cantidades de datos heterogéneos
M5	Abierto a recursos web tradicionales (no semánticos)
M6	Conforme al paradigma Web 2.0.
M7	Abierto a servicios

2.1.2. Tipos de aplicaciones web semánticas

En este trabajo se distinguirán las aplicaciones por el **perfil de usuario** al que van dirigidas, distinguiendo entre expertos de la Web Semántica, usuarios especializados, y usuarios finales. El perfil de usuario queda determinado por el tipo de conocimientos. Los expertos de la Web Semántica conocen la lógica de los lenguajes de descripción de ontologías, y son capaces de crear ontologías. Los usuarios especializados suelen ser expertos de algún dominio del conocimiento, pero no disponen de conocimientos avanzados de la Web Semántica. Los usuarios finales no disponen de conocimientos especializados. La tabla 2.3 muestra los conocimientos de cada uno de los perfiles de usuario que se pueden identificar atendiendo a sus conocimientos de Web Semántica y Web tradicional, así como los conocimientos de cada perfil de desarrollador capaz de proporcionar la funcionalidad necesaria para las mismas áreas de conocimiento.

Se puede identificar un nuevo tipo de perfil de usuario del que no se ha hablado hasta el momento en este trabajo. Se trata del perfil b, correspondiente a un usuario especializado en tecnologías web tradicionales de la parte servidora, con conocimientos acerca del funcionamiento de la web, capaz de crear sus propias páginas web, y que utiliza servicios tipo Google Gadget (descritos en la sección 2.3.2) para cortar y pegar código HTML generado por las aplicaciones tipo Google para incorporar funcionalidad a sus páginas web.

También aparece un perfil de desarrollador sin especialización (perfil D), que en la literatura se denomina End-User Developer (EUD), pero no será considerado en este trabajo. Se puede encontrar información sobre este tipo de perfil en el apéndice D.1.

La figura 2.1 muestra una clasificación de las aplicaciones web semánticas por tipo de usuario: experto en tecnologías de la web semántica (parte derecha de la figura), usuarios especializados (parte central) y usuarios finales (parte izquierda). Las aplicaciones para

Tabla 2.3: Conocimientos de los distintos perfiles de usuarios y desarrolladores.

Especialidad	Usuario	Desarrollador
Web Semántica	**Experto en Web Semántica (perfil a)** Conocimientos de lógica descriptiva, diseño de ontologías con herramientas (e.g. Protege, Swoop)	**Experto en tecnologías de la Web Semántica (perfil A)** Creación de aplicaciones (típicamente no web): lenguaje de programación (típicamente java), librerías especializadas para interfaz de usuario (eg. Java Swing, Eclipse SWT) y gestión de información semántica (e.g. Jena, Sesame)
Web (servidor)	**Usuario Google Gadget (perfil b)** Conceptos de web, invocación de servicios (eg. corta & pega de código HTML en página web)	**Experto en tecnologías Web Servidor (perfil B)** Lenguaje de programación (e.g. Java, PHP, Phyton, .NET, Ruby), arquitectura MVC (e.g. JSP, JSF, RubyOnRails)
Web (cliente)	**Diseñador Web (perfil c)** Herramientas de autor (e.g. HTML, CSS, Javascript, Flash)	**Experto en Tecnologías Web Cliente (perfil C)** Javascript, uso de servicios (e.g. Google Maps API), uso de librerías javascript (e.g. Dojo), XML, DOM, AJAX.
Ninguna	**Usuario Final (perfil d)** Ninguno de los anteriores	**Desarrollo por usuario final (EUD) (perfil D)** Ninguno de los anteriores.

usuarios especializados pueden ser subdivididas atendiendo al criterio de observar si el **usuario** manipula de forma **explícita** los elementos de las ontologías. Si este es el caso, la aplicación es (típicamente) un *wiki semántico*, y el usuario debe seleccionar el término adecuado de la(s) ontología(s) durante el proceso de anotación de la información. Si la manipulación es **implícita** y, por tanto, el usuario no percibe la existencia de ontologías ni datos semánticos se trata (típicamente) de un *portal semántico*. Estos tipos de aplicaciones se muestran en detalle en las secciones 2.3.1 y 2.3.3 respectivamente. Se puede observar que los expertos en Web Semántica disponen de muchas herramientas especializadas, mientras que los usuarios finales únicamente disponen de algunos navegadores semánticos. Los Semantic Web Pipes (descritos en detalle en las secciones 2.2.1 y 2.3.1) se encuentran entre los usuarios expertos y los especializados ya que permiten que usuarios con conocimientos de la Web Semántica creen aplicaciones de manera sencilla.

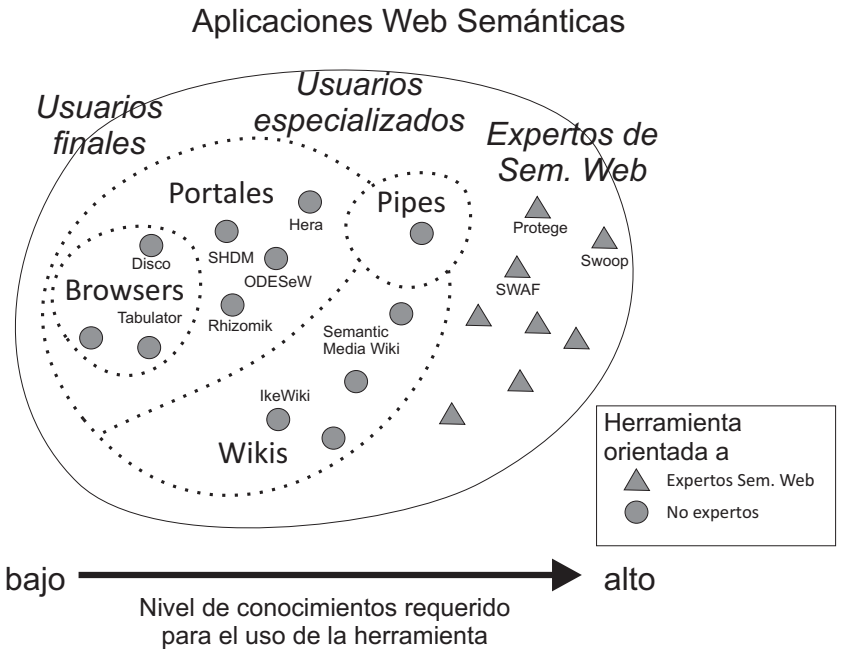

Figura 2.1: Clasificación de las aplicaciones web semánticas conforme al tipo de usuario al que van dirigidas.

La figura 2.2 muestra el punto de vista de los **desarrolladores** de aplicaciones, y cómo se relacionan las aplicaciones web semánticas con respecto a las aplicaciones web tradicionales. En esta figura, donde el eje horizontal muestra valores crecientes de funcionalidad y el eje vertical muestra valores crecientes de capacidad colaborativa, se puede ver que los primeros servidores de páginas web tenían los menores valores de funcionalidad y capacidad cooperativa. La aparición de los CGI's, y más tarde de los servlets, aumentó la funcionalidad que, junto a la creación de aplicaciones multiusuario (concepto de sesión) permitió crear las aplicaciones web actuales.

Tras la "frontera tecnológica" que suponen las tecnologías de la Web Semántica se encuentran los dos grandes grupos de aplicaciones web semánticas. Los portales semánticos añadieron funcionalidad a las aplicaciones web, pero con un escaso incremento en la capacidad colaborativa. Los wikis destacan por su capacidad colaborativa, pero la funcionalidad está centrada en la anotación, y es más reducida que la que ofrecen los portales semánticos. Los Semantic Web Pipes están muy cerca de los portales semánticos ya que tienen funcionalidad similar, pero mayor capacidad colaborativa debido a la compartición y reutilización de funcionalidad. Todas estas aplicaciones se sitúan en la "frontera del usuario

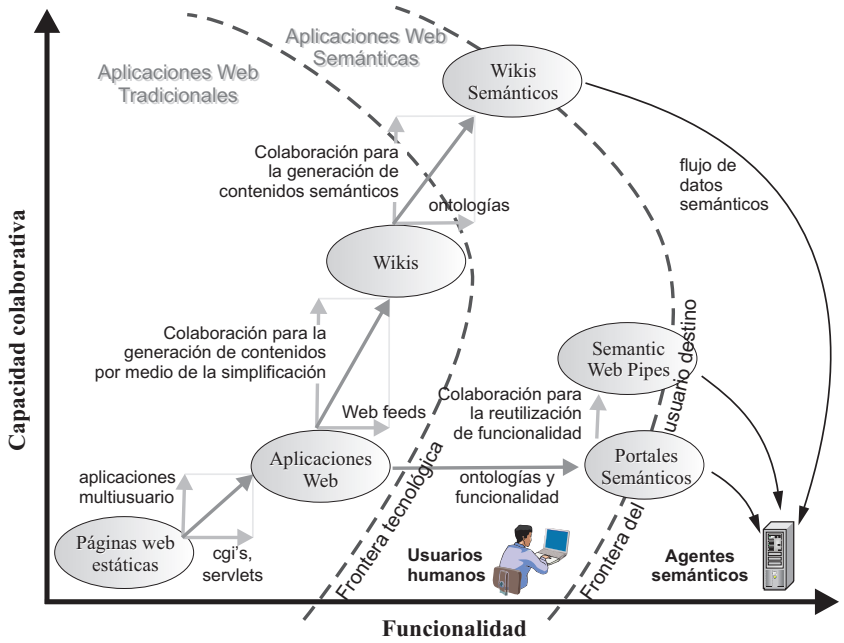

Figura 2.2: Tipos de aplicaciones web semánticas conforme a los desarrolladores.

destino" que divide a las aplicaciones dirigidas a personas de las dirigidas a agentes semánticos, indicando que están dirigidas a personas pero que a su vez generan información semántica susceptible de ser usada por agentes semánticos.

En las siguientes secciones se muestran las perspectivas que tienen de las aplicaciones web semánticas los diferentes perfiles de desarrolladores y usuarios identificados en esta sección.

2.2. Web Semántica para desarrolladores

Esta sección se centra en el punto de vista del desarrollador, tanto del desarrollador experto en tecnologías de la web semántica (perfil A) como del desarrollador de aplicaciones web (perfil B y/o C) de aplicaciones. Para el perfil A se describen las herramientas para visualización y edición de ontologías y datos semánticos, con especial atención en una herramienta reciente llamada Semantic Web Pipes. Para el desarrollador de aplicaciones (perfil B y/o C) se muestran los enfoques integradores existentes que pretenden proporcionar una visión unificadora de los componentes necesarios para crear una aplicación web semántica, así como las plataformas de desarrollo de aplicaciones web semánticas y las

librerías disponibles.

2.2.1. Herramientas para expertos en tecnologías de la Web Semántica (perfil A)

Visualización y edición de ontologías y datos semánticos

Para crear las herramientas orientadas a usuarios expertos en Web Semántica (perfil a) se utilizan herramientas de visualización de datos semánticos como IsaViz [Pietriga, 2001], que renderiza los gráficos en formato SVG [SVG, 2003], ó ClusterMaps [Fluit *et al.*, 2002].

Se han propuesto algunas especificaciones que pueden ser usadas en aplicaciones web, como las Graph-StyleSheets (GSS) [2], usado por IsaViz, o Fresnel [Bizer *et al.*, 2006], usado por PiggyBank. Fresnel usa los conceptos de "lentes" y "formatos"[3] para la presentación de datos RDF, unos conceptos complejos para un desarrollador medio.

Semantic Web Pipes

Semantic Web Pipes [Morbidoni *et al.*, 2008] es un sistema muy reciente inspirado por Yahoo! Pipes [4]. Se trata de una aplicación web (`http://pipes.deri.org`), creada con la plataforma ZK[5], que permite crear aplicaciones web semánticas como workflows que conectan las entradas y salidas mediante diferentes operadores (servicios) que operan con datos semánticos. Estas aplicaciones generan nuevos operadores (servicios) que pueden ser usados por otros usuarios, por lo que se acercan a la filosofía wiki de colaboración y contribución de los plugins de la plataforma propuesta en esta tesis (para desarrolladores) así como los templates propuestos (para usuarios).

Por tratarse de una herramienta que permite crear nueva funcionalidad sin requerir conocimientos de desarrollo de aplicaciones o de las tecnologías de la Web Semántica, sino únicamente conceptos básicos de Web Semántica y SPARQL, esta herramienta también es adecuada para usuarios de la Web Semántica (perfil a) y se describe en la sección 2.3.1.

La figura 2.3 muestra un ejemplo de workflow creado con Semantic Web Pipes. Los operadores disponibles se muestran en la parte izquierda de la figura.

[2] Véase `http://www.w3.org/2001/11/IsaViz/gss/gssmanual.html`

[3] En `http://www.w3.org/2005/04/fresnel-info/manual/#includingCSS` se puede encontrar un ejemplo completo aplicado a FOAF:Person

[4] Véase `http://pipes.yahoo.com`

[5] Véase `http://docs.zkoss.org/wiki/Introduction`

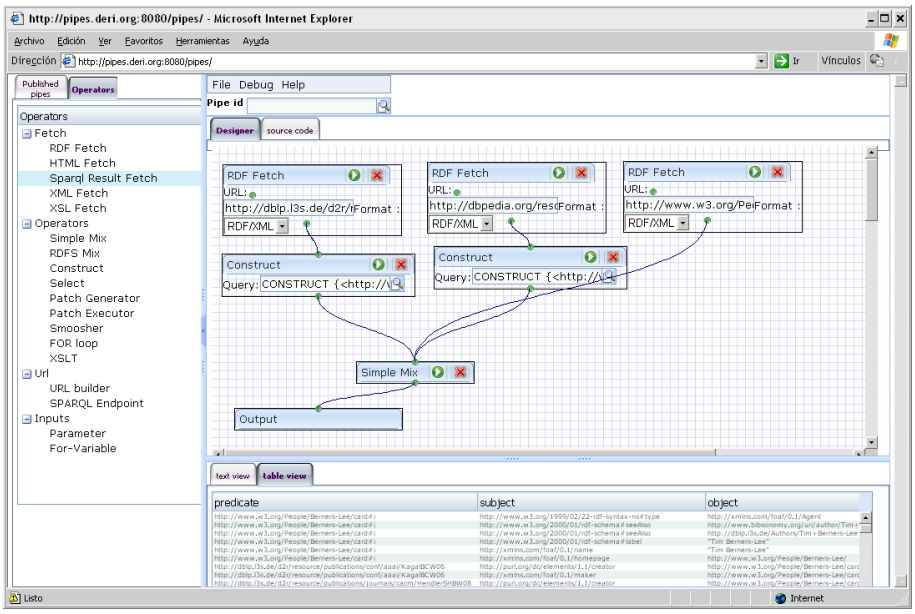

Figura 2.3: Ejemplo de workflow creado con Semantic Web Pipes.

Estos operadores (servicios) de Semantic Web Pipes no están centrados en la presentación o la entrada de datos, sino sólo en su transformación. En este sentido, una de las herramientas propuestas en esta tesis podría controlarse a través de un nuevo operador, de forma que se pudiera producir un interfaz más amigable (o adaptado al usuario) que la usada originalmente por Semantic Web Pipes. La creación de nuevos componentes es relativamente sencilla, y requiere conocimientos de java, XML, y del entorno ZK.

2.2.2. Herramientas para expertos en tecnologías Web (perfiles B y C)

Enfoque integrador del Semantic Web Framework (SWF)

Hasta la fecha, el único enfoque integrador propuesto en la literatura para fomentar el desarrollo modular de aplicaciones semánticas es el Semantic Web Framework, descrito a continuación.

El proyecto *Knowledge Web*[6] fue una red de excelencia financiada por el VI Programa Marco de la Unión Europea entre 2004 y 2008. Esta red agrupaba varias universidades, centros de investigación y empresas privadas. Dentro de este proyecto, uno de los objetivos

[6]Véase `http://knowledgeweb.semanticweb.org/`

fue establecer unas directrices para la creación de aplicaciones web semánticas. Durante 2007 y 2008, varias de las instituciones participantes de Knowledge Web definieron el Semantic Web Framework (SWF) en un documento titulado "Architecture of the Semantic Web Framework V2" [García-Castro *et al.*, 2008] como un primer paso para el desarrollo a gran escala de aplicaciones web semánticas, con el objetivo de ayudar a los desarrolladores a crear aplicaciones web semánticas y a reducir el coste de estos desarrollos.

SWF es un modelo de plataforma basada en componentes en la que se describen los tipos existentes de tecnologías de la Web Semántica, sus funcionalidades, y su dependencia de dichas tecnologías. La figura 2.4 muestra los 33 componentes identificados en SWF.

Aunque no existe una implementación de SWF, se puso a prueba la descripción del sistema por medio de ocho casos de prueba realistas. Para cada caso de prueba se indican los componentes necesarios y sus tecnologías asociadas. El documento muestra también cómo se puede implementar cada componente con las herramientas (alrededor de 200) existentes en ese momento (véase apéndice III de [García-Castro *et al.*, 2008]). Por ejemplo, la tabla 2.4 muestra las herramientas [7] existentes (en la fecha de finalización del documento, febrero de 2008) de los componentes englobados en "Data & Metadata Management" (primera columna de la figura 2.4). Este conjunto de componentes son los más cercanos a los proporcionados por las herramientas creadas en el marco de esta tesis.

Semantic Web Application framework (SWAF)

El término *plataforma* (framework) a veces se utiliza para referirse a un conjunto de librerías o clases usadas que implementan la estructura típica de cierto tipo de aplicaciones. Cuanto mayor sea la cantidad de código reutilizable que se almacena en estas librerías, mayor será el ahorro de trabajo y de tiempo para el desarrollador de la aplicación.

Sin embargo, otra interpretación del término *plataforma* la define como un conjunto de componentes software extensibles y configurables que modelan y resuelven cierto tipo de problemas. Según esta definición, una plataforma de desarrollo añade a estos componentes un "engine" que los coordina y ejecuta. Para cada aplicación concreta, los desarrolladores deben extender uno o varios de estos componentes. En esta sección se considera la segunda acepción del término. En cierto modo, este concepto tiene una gran relación con el del SWF anteriormente descrito, pues se podría considerar que una plataforma es una

[7] Conviene hacer notar que no todas las herramientas de un componente dado cubren toda la funcionalidad descrita del componente. La identificación del grado de cobertura quedó fuera del ámbito del documento.

Componente	#	Herramienta
Information directory manager	6	Aduna Metadata Server, Alvis, Beagle++, Gnowsis, Haystack, OWLIM
Ontology repository	15	KAON2, Jena, Sesame, Ontology Server, RDF Server, Knowledge zone, Onthology, OntoSelect, DAML Ontology Library, SchemaWeb, ONTOSEARCH2, Protégé Ontologies Library, OntStore, RDFPeer, RDF2GO
Data repository	3	DSpace, Lucene, Zebra
Alignment repository	2	COMA++, Alignment API and Alignment Server
Metadata registry	17	3store, AllegroGraph, Boca, Brahms, Hawk, The open metadata registry (prototypes 1-3), OASIS ebXML Registry, Oyster, Oyster2, Kowari, RDFGateway, RDF2GO, RDFStore, SemWeb, YARS

Tabla 2.4: Ejemplo de herramientas que pueden implementar el conjunto de componentes "Data & Metadata Management" del SWF.

implementación del SWF.

A diferencia del desarrollo de aplicaciones web tradicionales, en el que existen desde hace tiempo plataformas de desarrollo (por ejemplo el producto WebLogic de Oracle o WebSphere de IBM), el desarrollo de aplicaciones web semánticas carece por lo general de este tipo de plataformas. La única plataforma de desarrollo actualmente disponible es SWAF (Semantic Web Application framework) [Oren *et al.*, 2007a], que se compone de ActiveRDF [Oren *et al.*, 2007b], una librería para gestión de RDF, y una extensión (plugin) que se integra fácilmente con Ruby on Rails. Esta extensión permite al desarrollador generar el código fuente de vistas y controladores de elementos de ontologías, de forma que la creación de aplicaciones web se ve simplificada mucho, especialmente en la parte de generación del interfaz web. ActiveRDF se inspira en el sistema SHDM, descrito en la sección 2.3.3.

SWAF genera acciones y vistas para crear, mostrar, editar y borrar instancias, así como acciones (recuperar, listar y buscar) para manejar fuentes de datos remotas. El código de estas vistas es ERB [8], un sistema de plantillas con una sintaxis parecida a la de JSP (código entre tags $<\%$ y $\%>$), que embebe código Ruby en la salida HTML. Aunque el código de las vistas puede ser especializado por clases derivadas, el código de las vistas no puede ser modificado de forma sencilla.

[8] Véase `http://www.ruby-doc.org/stdlib/libdoc/erb/rdoc/`

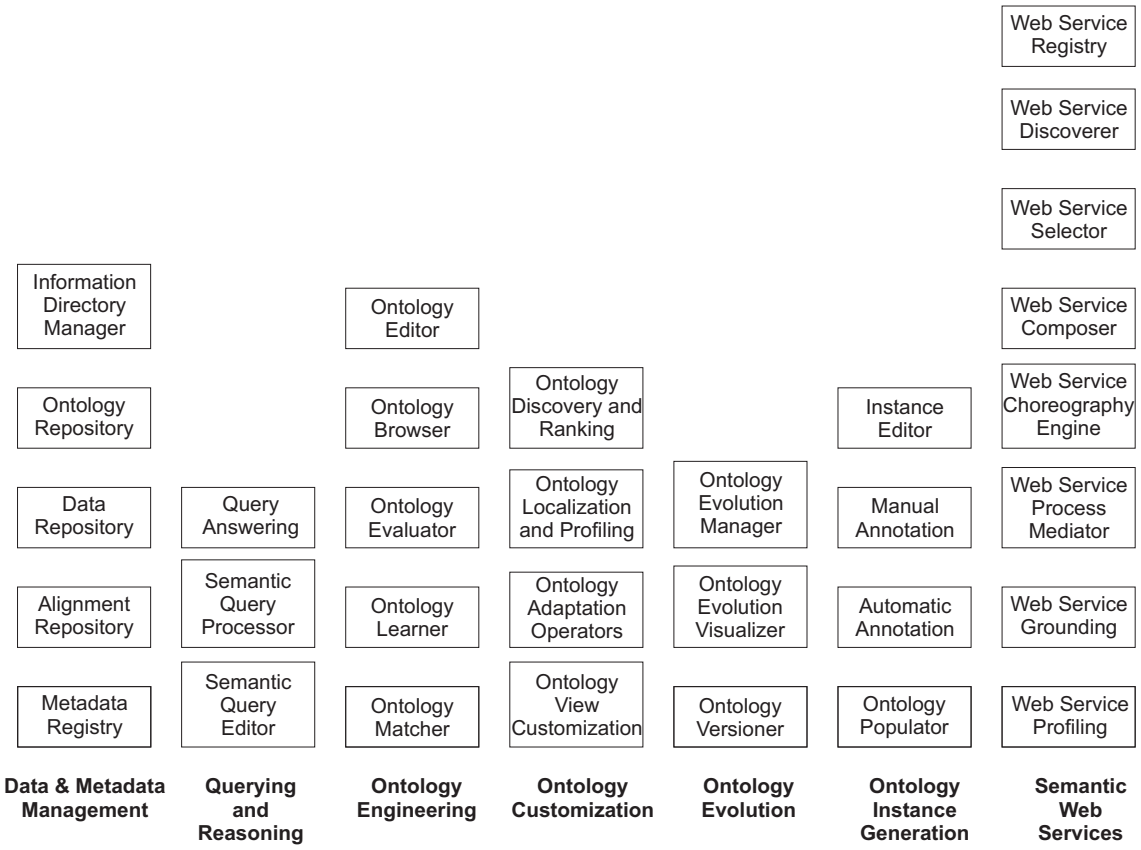

Figura 2.4: Componentes del Semantic Web Framework V2 [García-Castro *et al.*, 2008].

Dentro de ActiveRDF, el proyecto SWORD [9] diseñó un plugin para Rails para que las vistas se traten como plantillas, y por tanto puedan ser manipuladas, pero este proyecto se cerró antes de terminar la implementación.

Librerías

A continuación se describen brevemente las librerías de uso más común a disposición de los desarrolladores con conocimientos de las tecnologías de la Web Semántica.

Librerías básicas de gestión de ontologías y datos semánticos

Sin ser un análisis exhaustivo, entre las librerías de "propósito general" más utilizadas para la construcción de aplicaciones semánticas destacan Jena [10] y Sesame [11]. Estas librerías

[9] Véase `http://wiki.activerdf.org/SWORD/`

[10] Véase `http://jena.sourceforge.net/`

[11] Véase `http://www.openrdf.org/documentation.jsp`

gestionan la creación de ontologías en los lenguajes más comunes (RDF, RDFS, OWL, etc), con persistencia de datos (memoria, bases de datos, ficheros), y permiten realizar consultas al modelo de datos en SPARQL, RQL y otros lenguajes.

Todas estas librerías usan el lenguaje Java. Los desarrolladores de aplicaciones web que utilicen otros lenguajes (Phyton, PHP, Ruby, .NET) deben usar adaptadores para usarlas.

Librerías para modelado orientado a objetos

Como indican Oren et al en [Oren *et al.*, 2007a] las librerías como Jena y Sesame se basan exclusivamente en las clases definidas en la ontología y no en las instancias de un conjunto de datos semánticos. Esto es, no generan instancias de clases java que repliquen la(s) ontología(s) y los datos semánticos.

Algunos desarrolladores prefieren manejar un modelo orientado a objetos a partir de una ontología dada, en vez de pensar en términos de tripletas como propone RDF. Las siguientes librerías generan código Java a partir de una ontología expresada en RDF, o en los lenguajes más comunes.

- **ActiveRDF** [12] [Oren *et al.*, 2007a]. Para lenguaje Ruby. Genera instancias de clases java que replican la(s) ontología(s) y los datos semánticos.
- **RDFReactor** [13] [Völkel, 2006] y **Jastor** [14] [Kalyanpur *et al.*, 2004] permiten generar código Java a partir de ontologías OWL, concretamente una clase Java por cada clase de la ontología.
- **Elmo** [15], además, puede generar una ontología a partir de un conjunto de clases java.

Librerías para creación de interfaces de usuario web

A continuación se muestran las más destacadas por su orientación Web 2.0 y su soporte para multiples navegadores.

- **Google AJAX APIs** [16]. Se proporcionan librerías en distintos lenguajes de programación (e.g. PHP, Phyton, Java, Javascript) para acceder a funcionalidad proporcionad por Google. Desde creación de gráficas de todo tipo, mapas, tablas, etc., hasta hosting de librerías como Dojo, Yahoo User Interface Library, o Prototype. Orientado a desarrolladores con conocimientos medio-alto de tecnologías Web.

[12] Véase `http://www.activerdf.org/`
[13] Véase `http://semanticweb.org/wiki/RDFReactor`
[14] Véase `http://jastor.sourceforge.net`
[15] Véase `http://openrdf.org/doc/elmo`
[16] Véase `http://code.google.com/apis/ajax/`

- **MIT Simile Web Widgets** [17]. El widget más conocido es Timeline, utilizado para mostrar líneas temporales interactivas, pero hay otros widgets como TimePlot, TimeGrid, o Exhibit. Se distribuye en forma de librería javascript. Orientado a diseñadores web básicos con conocimientos de Javascript.

- **OAT Framework** [18]. Múltiples elementos básicos como gráficos de barras, varios tipos de tablas, árboles, y algunos elementos para RDF, como un visualizador (basado en SVG) y un navegador [19]. Se distribuye en forma de librería javascript. Orientado a diseñadores web básicos con conocimientos de Javascript.

2.3. Web Semántica para usuarios finales

En esta sección se describen las herramientas existentes en la actualidad para los usuarios definidos en la sección 2.1.2, a saber: expertos en Web Semántica, usuarios con conocimientos de la Web, y usuarios finales.

2.3.1. Herramientas para usuarios expertos en Web Semántica (perfil a)

En el contexto de este trabajo se necesita que usuarios no expertos en tecnologías web puedan conocer la estructura de los elementos de una ontología dada. A continuación se muestran las aproximaciones existentes.

Visualización y edición de ontologías y datos semánticos

En el área de la visualización de datos se distinguen dos grupos principales de herramientas: (1) aquellas orientadas a manejar ontologías escritas en lenguajes como RDF, RDFS u OWL, y (2) las orientadas a manejar datos semánticos (i.e. individuos RDF). Las primeras incluyen la creación de ontologías por medio de editores, algunos de ellos de naturaleza colaborativa. Destacan, por orden cronológico, Tadzebao y Webonto [Domingue, 1998], Protege [Noy *et al.*, 2001], OntoEdit [Sure *et al.*, 2002], SWOOP [20] y NeOn

[17] Véase `http://code.google.com/p/simile-widgets/`
[18] Véase `http://demo.openlinksw.com/DAV/JS/demo/index.html`
[19] Véase `http://demo.openlinksw.com/DAV/JS/rdfbrowser2/index.html`
[20] Véase `http://www.mindswap.org/2004/SWOOP`

Toolkit [21].

Wikis Semánticos

Los wikis aparecieron en 1995 como aplicaciones web con las que el usuario podía crear contenidos **textuales** de un modo muy sencillo. El más popular es MediaWiki [22], con el que se ha construido Wikipedia [23]. En estos sistemas, usuarios sin conocimientos de tecnologías web pueden ver cualquier página del wiki ("modo vista"), y modificar el contenido cambiando a "modo edición". Un conjunto muy sencillo de etiquetas de marcado (markup tags), mucho más sencillo que las de HTML, permite dar formato al texto, crear tablas, o crear enlaces a otros wikis o a páginas web externas al wiki. Además de este aspecto de amigabilidad, los wikis se caracterizan por un sistema de control de versiones que permite restaurar cualquier estado anterior de la página wiki. Más aún, los wikis tienen un sistema de control de acceso que impide que dos usuarios modifiquen el mismo contenido a la vez. Adicionalmente, los wikis suelen tener otras características, tales como: (1) visualización diferenciada de enlaces a páginas wiki inexistentes, (2) visualización de enlaces inversos (páginas que enlazan con la página actual), e (3) indexación de contenidos para permitir una búsqueda rápida por palabra clave. Todas estas características hacen de los wikis una herramienta colaborativa amigable y han demostrado ser responsables de su éxito.

Un wiki semántico [Oren *et al.*, 2006] añade la siguiente funcionalidad a un wiki tradicional: (1) anotación de contenidos, (2) representación formal de contenidos, y (3) navegación por contenidos.

Por ejemplo, el texto "Cervantes nació en Alcalá de Henares en 1547" puede ser anotado con información como: (1) Cervantes → *profesión* → Escritor, (2) Cervantes → *nacido en* → Alcalá de Henares, (3) Alcalá de Henares → *localizado en* → Madrid, y (4) Cervantes → *fecha de nacimiento* → 1547. Uno de los sistemas más extendidos es Semantic Media Wiki[24], aunque hay otros como: SemperWiki[25], IkeWiki[26], o Makna[27]. En febrero de 2009

[21] Véase `http://www.neon-project.org`
[22] Véase `http://www.mediawiki.org`
[23] Véase `http://www.wikipedia.org`
[24] Véase `http://sourceforge.net/projects/semediawiki`
[25] Véase `http://www.semperwiki.org`
[26] Véase `http://ikewiki.wastl.net:8080/`
[27] Véase `http://www.apps.ag-nbi.de/makna`

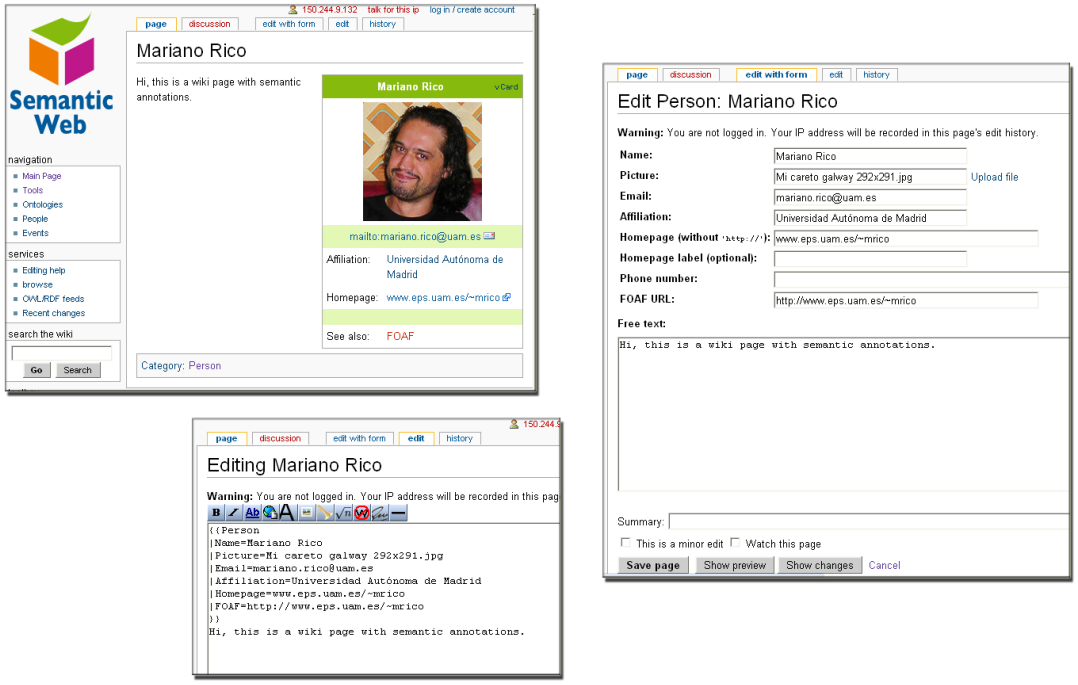

Figura 2.5: Ejemplo de anotación semántica en Semantic Media Wiki.

se encontraban activos 12 wiki-engines semánticos [28]. En todos ellos, el usuario debe establecer la relación entre dos términos seleccionándola de entre los múltiples elementos de la(s) ontología(s) del dominio. En el ejemplo anterior, el sistema debe tener definidas las relaciones *profesión*, *nacido en*, *localizado en*, y *fecha de nacimiento*. La figura 2.5 muestra un ejemplo [29] de anotación semántica en Semantic Media Wiki.

Aunque algunos de los términos mencionados pueden ser incluidos en wikis no semánticos como *categorías*[30], la ventaja de los wikis semánticos es que (1) permiten usar ontologías externas para extender el vocabulario de términos disponibles, y (2) permiten que el contenido de una página wiki sea el resultado de una consulta semántica. Por ejemplo, se podría tener una página wiki titulada "Escritores" que se rellene automáticamente con el resultado de la siguiente consulta: "de las instancias de la clase Escritor, muestra su nombre y apellidos, ordenados por apellido" (expresada con la sintaxis adecuada). Un

[28] Se puede ver una lista actualizada de proyectos activos en la que se muestran sus características en `http://semanticweb.org/wiki/Semantic_Wiki_State_Of_The_Art`

[29] Véase `http://semanticweb.org/wiki/Mariano_Rico`

[30] Véase `http://en.wikipedia.org/wiki/Implementation`

ejemplo más complejo podría ser una página wiki que muestre “Mujeres polacas galardonadas con el premio Nobel antes de los 40 años”. En el caso de wikis no semánticos, no tiene sentido crear *categorías* específicas para rellenar automáticamente estos ejemplos.

Semantic Web Pipes para usuarios

Aunque los Semantic Web Pipes están orientados a expertos en tecnologías de la Web Semántica y, por tanto, se describen en la sección 2.2.1, un experto en Web Semántica puede reutilizar fácilmente las aplicaciones (workflows de datos y operadores) para crear nueva funcionalidad o adecuar la funcionalidad existente a sus necesidades.

Las aplicaciones se denominan **Pipes**, y esta herramienta permite que los usuarios “clonen” pipes creados por otros usuarios. Estos nuevos pipes pueden ser modificados por el usuario, probados, y puestos a disposición de la comunidad de usuarios.

2.3.2. Herramientas para usuarios con conocimientos de la Web (perfiles b y c)

Herramientas para la creación de contenidos

Desde los orígenes de la Web han existido herramientas para asistir a los usuarios en la tarea de crear contenidos para la Web. Los primeros navegadores podían mostrar páginas web y editar sus contenidos, pero esta funcionalidad se desligó y ahora suele requerir de herramientas distintas, de una parte el navegador (e.g. Firefox, Internet Explorer), y de otra las herramientas de autor (e.g. Dreamweaver).

La aparición de las tecnologías asociadas al movimiento Web 2.0, esencialmente AJAX, propició la aparición de herramientas de autor en formato aplicación web. Este es el caso de herramientas gratuitas como Google Pages, Microsoft Spaces, o Yahoo PageBuilder. Las aplicaciones web tienen la ventaja, comparadas con las aplicaciones de escritorio, de que no tienen que instalarse y de que su actualización es transparente al usuario. Esto elimina la barrera que supone la instalación de aplicaciones para los usuarios menos experimentados (perfil d), y aumenta el número de usuarios que pueden usar estas herramientas.

Aunque estas herramientas suelen ocultar al usuario el código cliente (HTML, CSS, etc.), algunas permiten que al usuario incluir funcionalidad externa mediante el copiado y pegado de código fuente (típicamente HTML). Por ejemplo, los Google Gadgets pueden ser insertados en cualquier página web mediante esta técnica. Una vez que se ha seleccionado

y configurado un Google Gadget, se obtiene un bloque de código HTML que el usuario puede insertar en cualquier herramienta de autor que permita editar el código fuente. Los conocimientos necesarios para realizar esta tarea se limitan a conocer la estructura básica de los tags HTML. Por tanto, no se dirige estrictamente a usuarios finales (tipo d), pero se acerca bastante.

También existen entornos comerciales para el desarrollo de contenidos para la Web, como Macromedia Dreamweaver o complejos sistemas de gestión de contenidos (CMS) como Vignette.

Herramientas para visualización de ontologías

La mayoría de aproximaciones existentes están dirigidas a usuarios con conocimientos técnicos de ontologías, ya que no sólo deben conocer el significado de términos como *clase*, *relación*, *propiedad* o *instancia*, sino que deben conocer la semántica de primitivas de representación como *range*, *type*, *domain*, *funtional property* o *inverse functional property*, entre otras.

En la propuesta presentada en esta tesis se ha necesitado proporcionar a usuarios con bajos conocimientos de Web Semántica información acerca de la estructura de los componentes de una ontología, prescindiendo de detalles avanzados. A continuación se describen las aplicaciones más destacadas aunque el tipo de usuarios al que va dirigido no es el adecuado a los intereses de ese trabajo.

- **OWLDoc** [31] es una herramienta java que genera documentación al estilo"javadoc", es decir, páginas web enlazadas entre sí por los nombres de los términos, a partir de una ontología OWL. Esta herramienta es un plugin de Protege-OWL, por lo que no es una aplicación web.

- **Ldontospec** [32] es similar a OWLDoc, pero escrito en lenguaje Ruby. Se ha usado para documentar la ontología de programas de televisión usada por la BBC (`http://www.bbc.co.uk/ontologies/programmes/`).

- **Literate** [Parsia, 2008] describe en notación compacta (Notación Manchester) [33] el contenido de ontologías OWL. Se ha aplicado al borrador de la especificación de

[31] `http://www.co-ode.org/downloads/owldoc/`
[32] Véase `http://github.com/metade/ldontospec`
[33] Borrador W3C Diciembre 2008. Véase `http://www.w3.org/TR/owl2-manchester-syntax/`

OWL2 [34].

Navegadores de la Web Semántica

Los navegadores de la Web Semántica muestran al usuario representaciones de los datos contenidos en una fuente de datos semánticos. La mayor parte de los navegadores se implementan como aplicaciones web, para ser utilizadas por navegadores Web convencionales.

Siguiendo la filosofía inicial de los navegadores, en la que la comunicación es únicamente entre el cliente (navegador) y el servidor, se desarrolló Tabulator [Berners-Lee *et al.*, 2006]. Sin embargo, esta aproximación adolece de limitaciones severas: el navegador accede a Tabulator como a una página web cualquiera en `http://www.w3.org/2005/ajar/tab`, por tanto, cualquier intento de acceder a otro host que no sea a `w3.org` lanzará una violación de seguridad. En el caso de Firefox, se puede configurar el navegador para que permita esta operación, pero en Internet Explorer es imposible a día de hoy.

Por tanto, la aproximación seguida habitualmente es evitar que la navegación la haga el navegador. La navegación la hace el host en el que reside la aplicación, y el navegador únicamente muestra la información a la que va accediendo el host. Esta aproximación es la que siguen aplicaciones web como DISCO [35]o OntoWiki [36]. La figura 2.6 muestra un ejemplo de fuente semántica mostrada en Tabulator. La figura 2.7 muestra a OntoWiki.

En todos estos navegadores de la Web Semántica la forma de mostrar los datos semánticos es fija, y no se puede adaptar a las necesidades del usuario. La propuesta presentada en esta tesis está dirigida a permitir que la presentación se pueda adaptar al tipo de dispositivo que utiliza el usuario, sus deficiencias visuales (e.g. daltonismo o reducida agudeza visual), o sus preferencias estéticas.

2.3.3. Herramientas para usuarios finales (perfil d)

En esta sección se describen las herramientas disponibles para usuarios sin conocimientos de Web Semántica ni especialización en tecnologías web. Los usuarios con mínimos conocimientos de web, por ejemplo los que saben crear páginas web sencillas, han sido descritos en la sección 2.3.2. Aunque los usuarios descritos en esta sección no tengan

[34] Véase `http://www.w3.org/2007/OWL/wiki/Primer`
[35] Véase `http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/`
[36] Véase `http://demo.ontowiki.net/`

Figura 2.6: Navegador de la Web Semántica Tabulator.

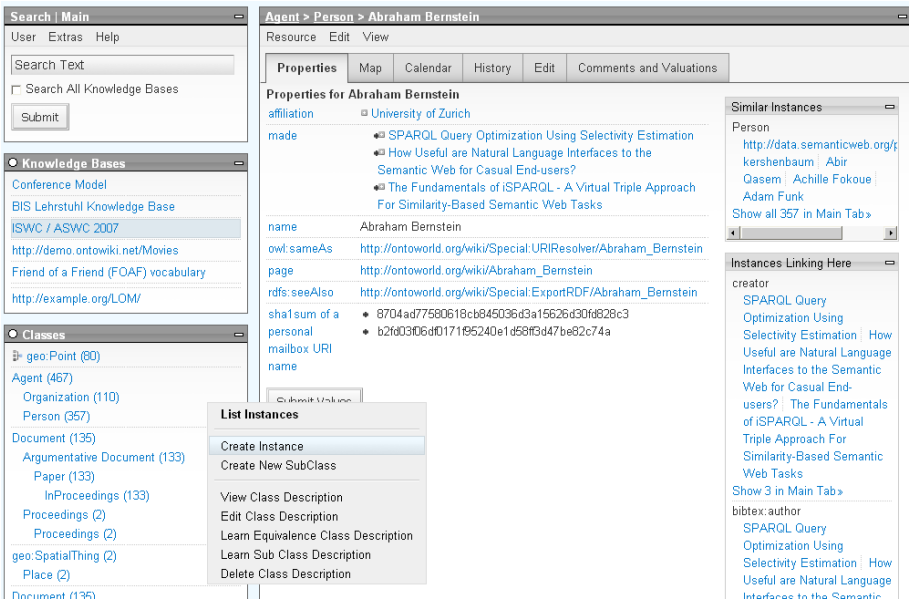

Figura 2.7: Navegador de la Web Semántica OntoWiki.

conocimientos de las áreas citadas, no significa que no tengan ningún conocimiento (se asume mundo abierto), pueden ser especialistas de otras áreas y capaces de utilizar una herramienta web como los portales web semánticos descritos a continuación.

Portales Web Semánticos

Lausen *et al.* describe [Lausen *et al.*, 2005a] los portales web semánticos como portales web (sitio web que recoge información de un grupo de usuarios con intereses comunes) basados en tecnologías de la web semántica en el que una comunidad de usuarios **comparte** e **intercambia** información. Este autor indica que la principal ventaja de estos sistemas es que pueden modelar la estructura del portal mediante una o varias ontologías. Al ser dichas ontologías una representación de un conocimiento consensuado, son idóneos para intercambiar información en una comunidad de interés y permiten un procesamiento automático de la información. Los portales web tradicionales (no semánticos) solventan este problema mediante diversos métodos de estructuración, como tipo de contenido, vista, metadatos propietarios, etc. Pero esto suele generar incompatibilidad entre portales y confunde al usuario.

El usuario interactúa con el sistema como en cualquier aplicación web tradicional, pero generando y editando información semántica. Aunque existen plataformas capaces de desarrollar este tipo de portales, e.g. ODESeW [Corcho *et al.*, 2006a] (2006), los inconvenientes de estos sistemas, comparado con las arquitecturas basadas en componentes extensibles, es que (1) son monolíticos, en el sentido de que la funcionalidad está centralizada, y (2) extender su funcionalidad es una tarea técnicamente compleja. Un ejemplo de esta complejidad se puede encontrar en [Keller *et al.*, 2004].

A continuación se describen los sistemas más destacados, por orden cronológico inverso.

Rhizomik

Rhizomik [García *et al.*, 2008a; García *et al.*, 2008b] es una plataforma que permite crear portales web semánticos. Usa la infraestructura Rhizomer [García y Gil, 2006] y un wiki-engine. Rhizomer es una infraestructura que permite presentar datos semánticos, navegar por ellos, y la edición de los mismos. Tanto la presentación como la edición se realizan de forma automática. La presentación de datos mejora otras presentaciones automáticas de grafos RDF gracias al paradigma Rhizomic, mediante el cual se puede presentar información semántica "cercana" (en el grafo) aunque se interpongan nodos anónimos, de forma

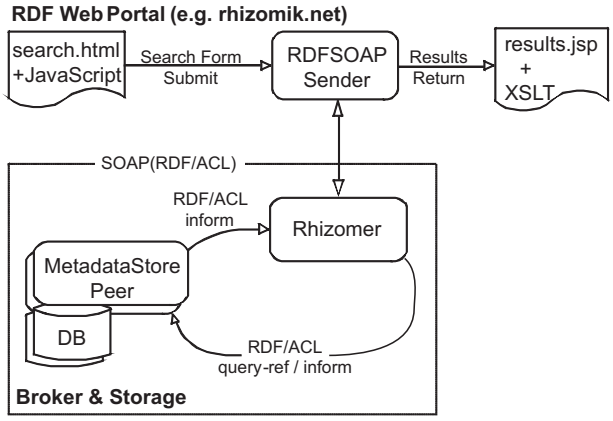

Figura 2.8: Arquitectura del portal semántico Rhizomik.

que se muestra más información que normalmente no se presenta. La figura 2.8 muestra la arquitectura del portal semántico Rhizomik. El código fuente de Rhizomik o de Rhizomer no es público aún, por lo que la evaluación no puede ser exhaustiva.

ODESeW

ODESeW [Corcho *et al.*, 2006b] comenzó siendo un front-end del workbench de ontologías WebODE [Arpírez *et al.*, 2001], pero evolucionó posteriormente hacia un portal web semántico. Siguiendo el patrón de diseño MVC, proporciona vistas reutilizables fácilmente extensibles. Las tecnologías requeridas para crear este tipo de vistas son: JSP 2.0 (con Tag Extension y EL (Expression Language)) y J2EE Java Beans. El sistema proporciona las siguientes vistas genéricas:

- Vistas para renderización individual (Term View) de conceptos, atributos, relaciones e instancias.
- Vistas amplias (Upper Term View), que permiten mostrar conjuntos de conceptos, atributos, relaciones e instancias.
- Otras vistas: árboles de conceptos de una ontología, listas de instancias, etc.

SHDM

Semantic Hypermedia Design Method (SHDM) [Lima y Schwabe, 2003b] [Lima y Schwabe, 2003a] es otro entorno para la creación de aplicaciones web semánticas. SHDM incluye en el modelo de la aplicación web el modelo semántico de la aplicación. El modelo de la aplicación también se ve enriquecido, por ejemplo, ampliando el modelo de navegación

mediante navegación por facetas. El entorno de creación de modelos se ha implementado sobre tecnología .NET [Ricci y Schwabe, 2006].

La ventaja de los sistemas basados en modelos [Nunes y Schwabe, 2006] es que buena parte del código de la aplicación puede ser generado automáticamente, y que mediante los adaptadores adecuados se pueden generar versiones adaptadas al dispositivo (como hace Paterno). Sin embargo, estos sistemas se basan en un modelo, que, obviamente, no puede abarcar todos los conceptos posibles. Por ejemplo, en [Fialho y Schwabe, 2007] se muestra cómo ampliar SHDM para incluir el concepto de "animated multimedia transitions", esto es, la transformación de una interfaz web inicial (incluso con elementos de formularios) en una final con elementos diferentes.

Hera

Hera [Vdovjak *et al.*, 2003] es otro entorno de desarrollo dirigido por modelos semánticos. En este caso se modela la entrada de un consulta por parte de un usuario, acerca de un dominio específico (por ejemplo, un museo), y el sistema genera una interfaz web en la que la navegación se hace a la medida de la consulta. Algunas partes del entorno son genéricas, pero otras dependen del dominio y deben rehacerse para cada dominio de aplicación. Considera la adaptación del interfaz a las características del dispositivo navegador utilizado y a algunas características del usuario (modelado con CCPP [37]).

La generación del interfaz web se realiza por medio de hojas de estilo XSLT. Cada aplicación tendrá un dominio específico y deberá especificarse una XSLT para convertir cada instancia del modelo conceptual a una instancia del modelo de aplicación, así como otra XSLT para pasar del modelo de aplicación a cada tipo concreto de dispositivo [Houben *et al.*, 2004].

La sección 5.3.1 muestra una comparativa de las características de algunas aplicaciones web semánticas representativas capaces de manejar plantillas para presentar datos semánticos, tales como wikis semánticos, navegadores de la Web Semántica, y portales semánticos. Se ha seleccionado una aplicación o infraestructura representativa de cada grupo: Semantic Media Wiki como wiki semántico, Fresnel (usado por el navegador Longwell), y Rhizomer (usado por el portal semántico Rhizomic), respectivamente.

[37] Véase apéndice D.3

2.4. Conclusiones y limitaciones del estado del arte

En este capítulo se han analizado los tipos de aplicaciones web semánticas atendiendo a los perfiles implicados en la creación y el uso de estas aplicaciones. Se han mostrado las herramientas de que disponen usuarios y desarrolladores agrupadas por tipos de conocimientos, esencialmente conocimientos de la Web Semántica y de la Web tradicional.

En cuanto a los **desarrolladores de aplicaciones**, las herramientas actuales requieren conocimientos avanzados de tecnologías de la Web Semántica y de tecnologías de creación de aplicaciones web, y únicamente proporcionan una infraestructura software sobre la que crear aplicaciones web semánticas.

Las plataformas dirigidas por modelos (model-based o model-driven) aportan la comodidad de la generación de gran parte del código de la aplicación, pero deben disponer de mecanismos sencillos que permitan especificar la generación de la interfaz web sin necesitar conocer toda la herramienta como usuarios o desarrolladores. Por ejemplo, una solución muy extendida es generar la interfaz web mediante la aplicación de hojas de estilo XSLT a los datos semánticos, como es el caso de los sistemas SHDM, Hera, o Rhizomik. Esta tarea requiere que los diseñadores web tengan unos conocimientos muy especializados alejados de sus competencias habituales. Otra solución habitual es basar la generación del interfaz web en JSP (caso de ODESeW), lo cual vuelve a requerir de diseñadores web con unos conocimientos muy especializados fuera de sus competencias y añade el riesgo que supone mezclar código de aplicación con código de presentación. Una aproximación novedosa son los Semantic Web Pipes, en los que se puede lograr mucha funcionalidad a través de una interfaz web bastante intuitiva, así como compartir y reutilizar esta funcionalidad en un entorno colaborativo.

En cuanto a los **usuarios**, la aproximación de ocultar la complejidad de la semántica da lugar a los portales semánticos, en la que los usuarios no requieren ningún conocimiento adicional para navegar y manipular datos semánticos, y la aproximación de manipulación explícita de la semántica da lugar a los wikis semánticos, en los que se vuelven a requerir conocimientos en cuanto al significado de los términos ontológicos utilizados.

La tabla 2.5 muestra un resumen del nivel de competencias requerido para desarrolladores y usuarios de aplicaciones web semánticas. Por tanto, el primer paso debe ser facilitar a los desarrolladores el proceso de creación de aplicaciones web, reduciendo el número de competencias y el nivel de las mismas en las tecnologías implicadas en la creación de aplicaciones web semánticas. Adicionalmente, desligar a los desarrolladores de la tarea del

Tabla 2.5: Nivel de competencias requerido para el uso/desarrollo de herramientas por parte de usuarios/desarrolladores. En una escala de 1 a 5, en el caso de los desarrolladores indica el nivel de complejidad de las herramientas necesarias. En el caso de los usuarios, indica la complejidad de uso de la herramienta.

	Desarrollador				**Usuario**			
	A	B	C	D	a	b	c	d
Portales Web Semánticos								1
Navegadores de la Web Semántica						2	2	
Wikis Semánticos				3				
Semantic Web Pipes	3				2			
Presentación y edición de ontologías y datos semánticos	5				3			
Herramientas creación de contenidos						2	2	
Visualización de ontologías						2	2	
Librerías		5	5					
SWAF		2	2					

diseño no sólo reduce competencias, sino que permite que expertos en diseño web puedan realizar diseños más atractivos para ser usados por las aplicaciones web semánticas.

Asimismo, se deben diseñar mecanismos que permitan a los diseñadores gráficos crear diseños atractivos sin necesidad de dominar tecnologías propias de desarrolladores de software (caso de XSLT), ni tener que mezclar código de aplicación con estética (caso de JSP).

Por analogía con la web tradicional, al igual que la Web 2.0 se centra en que sea el usuario final el que aporte contenidos de todo tipo, y ha obligado a un rediseño de las aplicaciones web para poder explotar el poder creativo de los usuarios, la Web Semántica puede delegar parte de su desarrollo a los usuarios, no en tareas relacionadas con la complejidad intrínseca de la lógica, pero sí en tareas de diseño gráfico o creación y explotación de datos semánticos.

Capítulo 3

Objetivo y propuestas

En este capítulo, y a la vista de lo mostrado en los capítulos anteriores, se muestran los objetivos (tesis) de este trabajo, las propuestas para alcanzar los objetivos, las pruebas de su consecución, así como las suposiciones realizadas y las limitaciones de las soluciones aportadas. Finalmente se muestran las contribuciones de la tesis.

3.1. Introducción

En matemáticas, la demostración por método directo consiste en el encadenamiento lógico de proposiciones, de manera que, de las hipótesis es posible llegar a la tesis. En este caso, la hipótesis es el supuesto que se acepta como cierto y que sirve de base para el razonamiento. La tesis es lo que se pretende demostrar. Los teoremas matemáticos (verdades no evidentes pero demostrables) son enunciados hipótesis + tesis. Por ejemplo, el teorema de Pitágoras "dado un triángulo rectángulo de catetos a y b, la hipotenusa mide $\sqrt{a^2+b^2}$" se compone de la hipótesis "dado un triángulo rectángulo de catetos a y b" y de la tesis "la hipotenusa mide $\sqrt{a^2+b^2}$".

En las ciencias experimentales, sin el apoyo de la lógica matemática y, por tanto, sin la capacidad de demostración, se recurre a las teorías. Es comúnmente aceptado que la mejor manera de abordar la solución de un problema es la que propone el llamado "método científico". Según [Walker y Walker, 1963], este método intenta la validación de un modelo mediante (1) postular un modelo basado en las observaciones o mediciones experimentales existentes, (2) verificación de las predicciones de este modelo con respecto a las observaciones o mediciones ulteriores, y (3) ajuste o sustitución del modelo, conforme lo requieran las nuevas observaciones o mediciones, y vuelta al punto (1). De esta forma se van refinando las teorías, hasta llegar a modelos tan avanzados como la gravitación desde la perspectiva de la Teoría de la Relatividad, o el Modelo Estándar en física de partículas.

El método científico ha sido utilizado desde hace mucho tiempo. En el siglo XVII, Robert Boyle propuso las siguientes etapas del método científico [Cárdenas, 2005]:

1. Reunión de los hechos.
2. Construcción de una *hipótesis* relacionada con los hechos.
3. *Probar* la hipótesis con nuevas experiencias.
4. Si estas desaprueban la hipótesis, construir una nueva con los hechos antiguos y los nuevos

5. Repetir hasta encontrar la solución

Sin embargo, actualmente [Cegarra Sánchez, 2004] se ha cambiado la terminología y se considera que, a efectos prácticos, el método científico consta de las siguientes etapas:

1. Planteamiento del *objetivo*
2. Reunión de los datos conocidos
3. Organización de los datos
4. *Propuesta* de una posible solución
5. *Prueba* de la solución
6. Presentación de los resultados

3.2. Objetivo

Según [Cegarra Sánchez, 2004], es aconsejable plantear un *objetivo general* y uno o varios *objetivos inmediatos*. El primero pretende abarcar un campo más amplio que los segundos, dentro del mismo tipo de problema, y no depende tanto de la evolución de la investigación.

Objetivo general 1
Aumentar el uso de ontologías y datos semánticos en la Web Semántica.

Objetivo inmediato 1
Proporcionar métodos y herramientas que faciliten el desarrollo de aplicaciones web semánticas a los desarrolladores.

Objetivo inmediato 2
Proporcionar métodos y herramientas que faciliten el uso de datos semánticos a los usuarios finales.

3.3. Propuestas

Propuesta 1
Considerando la suposición 1 se propone crear herramientas que reduzcan (1) los tiempos de desarrollo, (2) las dependencias entre los desarrolladores, y (3) el número de tecnologías y el nivel de conocimientos que deben tener los desarrolladores de aplicaciones web semánticas.

Propuesta 2
Los usuarios de las aplicaciones web semánticas creadas con las herramientas propuestas deben asignar unos valores aceptables de calidad. La calidad viene determinada por las suposiciones 2 y 3.

Propuesta 3
Crear aplicaciones web semánticas orientadas a usuarios especializados. Estas aplicaciones deben ser capaces de permitir que los usuarios aporten su conocimiento sin necesidad de tener conocimientos de las tecnologías de la Web Semántica. En concreto se propone, como ejemplo de usuarios especializados, a los expertos en creación de interfaces de usuario web.

Propuesta 4
Crear aplicaciones web semánticas orientadas a usuarios finales que permitan a estos usuarios usar y crear datos semánticos sin necesidad de tener conocimientos de las tecnologías de la Web Semántica.

3.4. Pruebas

Las pruebas descritas a continuación deben concluir que las propuestas descritas cumplen los objetivos descritos.

Prueba 1
Se proporciona a un conjunto de desarrolladores de aplicaciones web semánticas la infraestructura propuesta para la creación de aplicaciones web semánticas. Los resultados experimentales muestran que los tiempos de desarrollo se reducen, que las dependencias entre los desarrolladores se reducen, que los requerimientos para desarrollar este tipo de aplicaciones se reducen, y que los desarrolladores necesitan menos herramientas para crear este tipo de aplicaciones.

Esta prueba debe concluir que la propuesta 1 satisface el objetivo inmediato 1. La infraestructura propuesta se describe en el capítulo 4, y los experimentos con desarrolladores se encuentran en la sección 6.1.

Prueba 2
Los usuarios de las aplicaciones creadas con la infraestructura propuesta asignan valores elevados de usabilidad y satisfacción de usuario.

Esta prueba debe concluir que la propuesta 2 satisface el objetivo inmediato 1. Las herramientas creadas con la infraestructura propuesta se describen en el capítulo 5. La evaluación experimental con usuarios de las herramientas creadas con la infraestructura propuesta se muestran en la sección 6.2.

Prueba 3
Se proporciona a un conjunto de diseñadores web una aplicación web semántica que permite a éstos crear plantillas web que manejan datos semánticos, sin requerir conocimientos de las tecnologías de la Web Semántica. Estas plantillas pueden ser usadas fácilmente por creadores de aplicaciones web. Los resultados experimentales muestran que la usabilidad de la herramienta es alta, al igual que la satisfacción del usuario con la interfaz.

Prueba 4
Se crea una aplicación que explota las plantillas creadas por los diseñadores web mediante una plataforma orientada a usuarios finales creada por una empresa comercial. Todas las aplicaciones de la plataforma se configuran y utilizan de forma sencilla y similar. Se considera que no es necesaria una evaluación experimental que demuestre la facilidad de su uso por usuarios finales (suposición 4).

Estas dos últimas pruebas deben concluir que las propuestas 3 y 4 satisfacen el objetivo inmediato 2.

3.5. Suposiciones

Las suposiciones que se describen a continuación permiten contextualizar las soluciones aportadas y la relevancia de las contribuciones de la tesis, determinando el contexto en el se puede asegurar que las propuestas validan los objetivos.

Suposición 1
Simplificar la creación de la parte servidora y cliente de las aplicaciones web semánticas facilita que los expertos en tecnologías de la Web Semántica creen aplicaciones web semánticas.

Suposición 2
La calidad de una aplicación web se puede determinar por la medida de la usabilidad de la misma.

Suposición 3
La calidad de una aplicación web se puede determinar por la medida de la satisfacción del usuario con respecto a la interfaz de usuario de la misma.

Suposición 4
Las aplicaciones creadas con la infraestructura de Google Gadgets están orientadas a usuarios finales.

3.6. Limitaciones

Las limitaciones de las soluciones aportadas permiten determinar líneas futuras de investigación. Se pueden agrupar en tres tipos: limitaciones por las herramientas utilizadas, limitaciones por la implementación, y limitaciones por la experimentación.

3.6.1. Limitaciones por las herramientas utilizadas

Limitación 1
Las aplicaciones desarrolladas sobre la plataforma propuesta adolecen de las limitaciones del sistema gestor de wikis (wiki-engine) utilizado. En concreto, las facilidades para la creación de formularios tienen algunos errores que impiden un uso eficiente de ciertos elementos del formulario como selectores múltiples (checkboxes) o únicos (radioboxes).

3.6.2. Limitaciones por las herramientas creadas

Limitación 2
La plataforma desarrollada está limitada por las características de su diseño. En concreto, la implementación actual evita almacenar datos semánticos en memoria, por lo que cada vez que se añade un dato se lee el modelo de disco, almacenándolo en memoria, se añade

el dato, y se vuelve a escribir el modelo actualizado en disco, liberándolo de memoria. Esto funciona bien para modelos de tamaño medio (miles de instancias), pero se vuelve inmanejable para modelos grandes. La solución pasa por soluciones de almacenamiento en base de datos, pero requieren complicar el proceso de instalación.

3.6.3. Limitaciones por la experimentación

Limitación 3

El número de participantes en los experimentos es un compromiso entre precisión en la medida y la dificultad de encontrar participantes cualificados.

Limitación 4

El número de participantes que ocupaban el papel de diseñadores web profesionales es bajo dada la dificultad en encontrar participantes con altos conocimientos de tecnologías web cliente.

3.7. Contribuciones de la tesis

3.7.1. Separación de roles en la creación de aplicaciones web semánticas

Como resultado del análisis de las partes implicadas en la creación de una aplicación web semántica, aparecen diferentes roles para desarrolladores y usuarios finales. La figura 3.1 muestra una separación clara entre desarrolladores (devel1 y devel2) y usuarios finales (user1, user2, user3) debido a las competencias requeridas.

Se identifican dos tipos diferentes de desarrolladores. El devel1 tiene el rol de "desarrollador de aplicaciones web semánticas", proporcionando elementos para la creación de aplicaciones web semánticas (F-plugins en la figura). El devel2 representa a otro tipo de desarrollador que no contribuye con plugins, sino que aprovecha los datos semánticos creados por las aplicaciones del devel1.

Para dar soporte a estos desarrolladores, se ha creado un sistema que liga los desarrollos realizados por los expertos en tecnologías de la Web Semántica (devel1) y los de los desarrolladores tradicionales de aplicaciones Web (devel2). Este sistema se ha llamado Fortunata y se detalla en secciones posteriores.

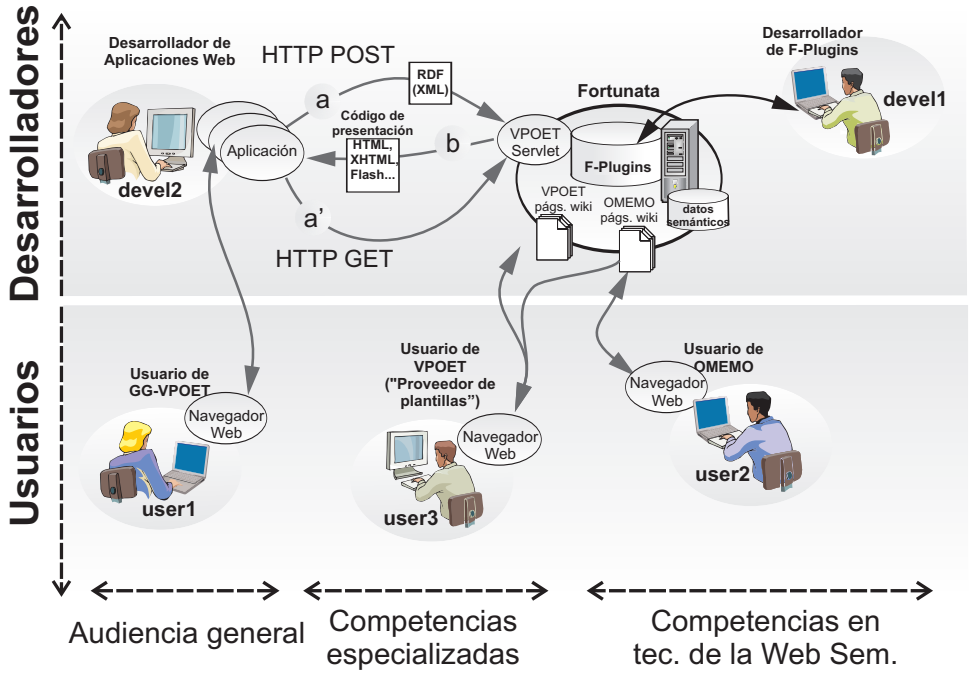

Figura 3.1: Roles de los participantes en la metodología propuesta

A nivel de usuarios, el vínculo entre los dos extremos, expertos en tecnologías de la Web Semántica (user2) y usuarios finales (user1), lo proporcionan los diseñadores web (user3). Estos usuarios deben disponer de un sistema que oculte al máximo las complejidades de las tecnologías de la Web Semántica y a la vez permita utilizarlas. Para lograrlo se ha proporcionando a los diseñadores web:

- un entorno colaborativo, de forma que la información aportada por un usuario puede ser aprovechada por los demás, dividiendo así una tarea laboriosa o compleja en pequeñas tareas sencillas.
- formularios, un concepto familiar para cualquier usuario de aplicaciones informáticas
- macros, similares al concepto bien conocido de "función" en las hojas de cálculo, habitual entre este tipo de usuarios

3.7.2. Creación de aplicaciones web semánticas por desarrolladores

De los requerimientos anteriormente citados, el primero (entorno colaborativo) lo proporcionan aplicaciones como los wikis. El segundo (formularios) lo proporciona sólo un

reducido número de wikis, y el tercer requerimiento (macros) lo deben proporcionar los desarrolladores de las aplicaciones.

En 2006 existían únicamente dos wiki-engines que permitían extensiones y formularios y por tanto podían servir bien para alcanzar estos objetivos: JSPWiki[1] y Twiki[2]. El primero usa tecnologías Java, mientras que el segundo se basa en Perl. Por consideraciones de compatibilidad con otras tecnologías utilizadas, se ha seleccionado JSPWiki como plataforma sobre la que construir esta metodología.

Después de un periodo de experimentación con diversos wiki-engines se decidió aprovechar esta tecnología. Esta aproximación tecnológica no requiere desarrolladores con experiencia en múltiples lenguajes y tecnologías, sino sólo fundamentos de los wikis, y conocimientos básicos de tecnologías de la web semántica y un lenguaje de programación. Para este tipo de desarrolladores, y para un wiki-engine concreto, se ha creado un entorno software al que se ha llamado Fortunata. Este software explota los plugins, piezas software que extienden una funcionalidad dada. En este caso, nuestros plugins extienden la funcionalidad de un wiki open-software. Las aplicaciones diseñadas con este paradigma permiten a los desarrolladores crear funcionalidad de una manera descentralizada.

El desarrollo tradicional de aplicaciones centraliza el código fuente. Por tanto, extender la funcionalidad típicamente requiere acceder al código fuente, modificarlo, y compilarlo. El resultado es una nueva versión de la aplicación. Sin embargo, los plugins permiten que los miembros de una comunidad de desarrolladores contribuyan creando nueva funcionalidad con un grado mínimo de dependencia. Cuando un desarrollador ha creado y probado un nuevo plugin, el código fuente se envía al administrador del wiki. Si el código es considerado válido y seguro será compilado y añadido al wiki-engine. A diferencia de los entornos de desarrollo tradicionales, esta adición no requiere comprobar dependencias o compilar el código de toda la aplicación. Incluso, en el sistema propuesto, esto se puede realizar mientras el wiki está funcionando.

Las tecnologías de la Web Semántica proporcionan una ventaja adicional: una integración de datos más simple. Las aplicaciones basadas en Fortunata se componen de un conjunto de plugins que gestionan una fuente de datos semántica. Estas aplicaciones pueden integrar fácilmente datos semánticos de otras aplicaciones basadas en Fortunata, como muestran los experimentos de la sección 6.1.1.

[1] Véase `http://jswpwiki.org`
[2] Véase `http://twiki.org`

3.7.3. Aplicaciones Web Semánticas para usuarios

Se han creado dos aplicaciones web semánticas usando la infraestructura Fortunata: OMEMO y VPOET.

- OMEMO permite que usuarios sin conocimientos de lenguajes de ontologías puedan saber qué componentes (clases, propiedades, y relaciones) se declaran en una ontología dada. OMEMO genera automáticamente páginas wiki con información acerca de cada elemento y comprueba periódicamente, accediendo a la URL, si se ha producido algún cambio en la ontología o ya no es accesible. En la figura 3.1, el user2 representa a este tipo de usuario.

- VPOET permite a los usuarios finales, denotados en este contexto como "proveedores de plantillas" (TPs), crear plantillas de visualización para un elemento de ontología dado. Estas plantillas puede ser creadas por cualquier usuario con conocimientos básicos de HTML o Javascript usando simples macros. Estos TPs pueden obtener información acerca del elemento de ontología leyendo las páginas wiki generadas por OMEMO, o leyendo otras páginas creadas manualmente que referencien a las páginas de OMEMO. En la figura 3.1, user3 representa a este tipo de usuario.

Además de crear las plantillas de visualización, los TPs indican las características de sus plantillas usando formularios, especificando detalles tales como el comportamiento en caso de cambio de tamaño de fuente, tamaño (preferido, mínimo, máximo), o colores dominantes.

Como cualquier otra aplicación basada en Fortunata, toda la información generada se publica en forma de datos semánticos, de forma que puede ser usada por sistemas abiertos, mediante un canal HTTP GET/POST. La figura 3.1 muestra este canal en el caso de VPOET, y cómo es explotado por desarrolladores como devel2. Como caso de uso, se ha utilizado este canal creando un Google Gadget llamado GG-VPOET. Usuarios finales como user1 pueden usar GG-VPOET para renderizar fuentes de datos semánticos indicando una plantilla existente.

Otras aplicaciones pueden explotar este canal. Por ejemplo, se está usando este canal para solicitar la plantilla más adecuada para un perfil de usuario dado. Este perfil de usuario experimental contiene datos semánticos con información acerca de las limitaciones visuales del usuario y el dispositivo utilizado.

3.7.4. Evaluaciones experimentales

Se ha evaluado Fortunata en cuanto a la usabilidad de las herramientas que se pueden generar con esta herramienta, así como en cuanto a las facilidades que proporciona para la colaboración entre desarrolladores (sección 6.1.1.

Las herramientas creadas usando Fortunata, OMEMO y VPOET, han sido evaluadas en cuanto a su usabilidad y la satisfacción del usuario (sección 6.2).

Capítulo 4

Creación de aplicaciones web semánticas por desarrolladores

En este capítulo se describe Fortunata, una solución para desarrolladores de aplicaciones web semánticas. Esta plataforma simplifica la creación de este tipo de aplicaciones centrando los esfuerzos del desarrollador en la creación de plugins embebidos dentro de un wiki, reduciendo las competencias requeridas y los tiempos de desarrollo. Una aplicación web semántica basada en Fortunata se compone de plugins y de páginas wiki. Estos plugins se pueden reutilizar, lo que permite que los desarrolladores puedan crear de forma colaborativa una aplicación web semántica, con un grado mínimo de dependencias entre ellos. La evaluación experimental (véase sección 6.1) confirma la reducción de las competencias requeridas y los tiempos de desarrollo.

4.1. Introducción

Como se ha descrito en el capítulo 2, de una parte, los desarrolladores de aplicaciones web que quieren incorporar tecnologías de la Web Semántica encuentran una barrera de adopción muy alta debido a la complejidad de estas tecnologías. Para estos desarrolladores se han creado dos aplicaciones, descritas en el capítulo 5, que facilitan la incorporación de datos semánticos a las aplicaciones web. Por otra parte, los expertos en tecnologías de la Web Semántica encuentran complejas las tecnologías implicadas en la creación de aplicaciones Web. Fortunata va dirigido a este segundo grupo de desarrolladores, simplificando la creación de interfaces web de usuario por medio de las tecnologías wiki.

Para comprender la dificultad que supone para un experto en tecnologías de la Web Semántica crear una aplicación Web, supongamos que un desarrollador como el descrito quiere convertir su aplicación local en una aplicación web. La aplicación local está escrita en Java, usando la librería Jena, con una funcionalidad sencilla: lee un conjunto de datos semánticos (accesibles en Internet), presenta un interfaz al usuario para que introduzca cierta información, realiza un conjunto de cálculos, y escribe los resultados como datos semánticos conforme a una ontología propia creada *ad hoc*.

Para poder crear la versión web de esta aplicación, deberá realizar las tareas que se muestran en la tabla 4.1. Esta tabla muestra las tareas y las tecnologías necesarias para poder llevarlas a cabo. Comparado con la versión local de la aplicación, que sólo requiere conocimiento de Java y Jena, resulta evidente que el esfuerzo resulta muy alto.

Fortunata[1] es una plataforma diseñada para facilitar la creación de aplicaciones web

[1]Se puede encontrar una descripción detallada, con ejemplos prácticos, en `http://ishtar.ii.uam.`

Tarea	Conocimientos
Crear y mantener un servidor web	Servlet engines (e.g Tomcat, JRun), Servlets 2.4 (estructura y despliegue de archivos war)
Crear la parte servidora	JSP, JSF
Crear la parte cliente	HTML, CSS, javascript, DOM, AJAX, etc.
Si la aplicación es compleja, organizar la parte cliente y servidora mediante una arquitectura software	Patrón de diseño MVC

Tabla 4.1: Tareas y conocimientos técnicos para desarrollar aplicaciones web para un desarrollador Java típico.

semánticas a desarrolladores con conocimientos de tecnologías de la Web Semántica pero con escasos conocimientos de las tecnologías Web. Aunque facilita especialmente la creación del interfaz web, también está orientada a reducir el nivel de conocimientos requeridos de las tecnologías de la Web Semántica, de forma que los conocimientos requeridos sean los menores posibles. Concretamente, Fortunata se compone de una librería java que delega (1) las tareas de presentación por medio de un wiki-engine sobre el que se apoya, y (2) la gestión y publicación de datos semánticos por medio de un conjunto sencillo de métodos para la gestión de ontologías y datos semánticos.

Esta no es la primera aproximación que intenta combinar semántica y tecnologías Web. Al menos, en parte, esto ha sido tratado en Wikis Semánticos (véase sección 2.3.1), Portales Semánticos (véase sección 2.3.3), y más recientemente en los Semantic Web Pipes (véase sección 2.3.1). Sin embargo, hay importantes diferencias con respecto a estas aproximaciones que se detallan a continuación.

- Los Wikis Semánticos [Oren *et al.*, 2006] se centran en la creación colaborativa de datos semánticos y su publicación al estilo de los wikis, normalmente combinado con texto en lenguaje natural. Sin embargo, Fortunata se centra en la creación colaborativa de aplicaciones que exploten esos datos, por medio de plugins para wikis (llamados Fortunata-plugins, o simplemente F-plugins). Por tanto, mientras que los wikis semánticos se centran en crear contenidos por medio de usuarios finales,

es/fortunata. El código fuente de Fortunata, y el de las aplicaciones OMEMO y VPOET se encuentra en `http://code.google.com/p/fortunata`

Fortunata se centra en crear aplicaciones por medio de desarrolladores.

- Los Portales Semánticos [Lausen *et al.*, 2005a] también se centran en la creación colaborativa de datos semánticos, pero a diferencia de los wikis semánticos, son más rígidos y obligan a usar modelos de conocimiento (ontologías) que serán mostrados a través de formularios. Estos portales suelen presentar los datos semánticos en forma de tablas que sólo pueden configurar los desarrolladores del portal. Fortunata, además de centrarse en la creación de aplicaciones en vez de en contenidos, proporciona a los desarrolladores la capacidad de transformar los datos desde que el usuario los introduce cuando rellena un formulario hasta la publicación de los datos semánticos.

- Semantic Web Pipes [Morbidoni *et al.*, 2008], al igual que Fortunata, se centra en desarrolladores de aplicaciones que gestionan datos semánticos. Las aplicaciones se crean como workflows que conectan las entradas y salidas de diferentes "operadores" que tratan las fuentes de datos o manipulan los datos. Estas aplicaciones son consideradas nuevos "operadores", de forma que pueden ser reutilizados por otros desarrolladores, compartiendo esta filosofía de colaboración contributiva con Fortunata. Sin embargo, a diferencia de Fortunata (en realidad, de una de sus aplicaciones, VPOET), los Semantic web Pipes no se centran en la presentación (o la entrada) de los datos, sólo en su transformación, proporcionando unos interfaces muy sencillos para entrada de datos y una presentación tabular para presentar los datos semánticos. En este sentido, las plantillas de VPOET podrían complementar la funcionalidad de los Semantic Web Pipes.

Comparado con estas plataformas de creación de aplicaciones web semánticas, la contribución de Fortunata es combinar las **ventajas** de los wikis semánticos (usando su sencilla sintaxis), portales semánticos (uso de formularios para introducir datos que serán convertidos en datos semánticos), y Semantic Web Pipes (transformación de datos); minimizando las **desventajas** de los wikis semánticos (edición no controlada o muy compleja), de los portales semánticos (tratamiento complejo, o imposible, de los datos introducidos por los usuarios), y de los Semantic Web Pipes (escasa capacidad de presentación y recogida de datos). En cuanto a funcionalidad y capacidad colaborativa, la ubicación de Fortunata con respecto a estas plataformas se muestra en la figura 4.1.

En esta figura se muestran las aplicaciones creadas con Fortunata como un conjunto de elementos que rodea un núcleo común. Los elementos son los plugins, y el núcleo la

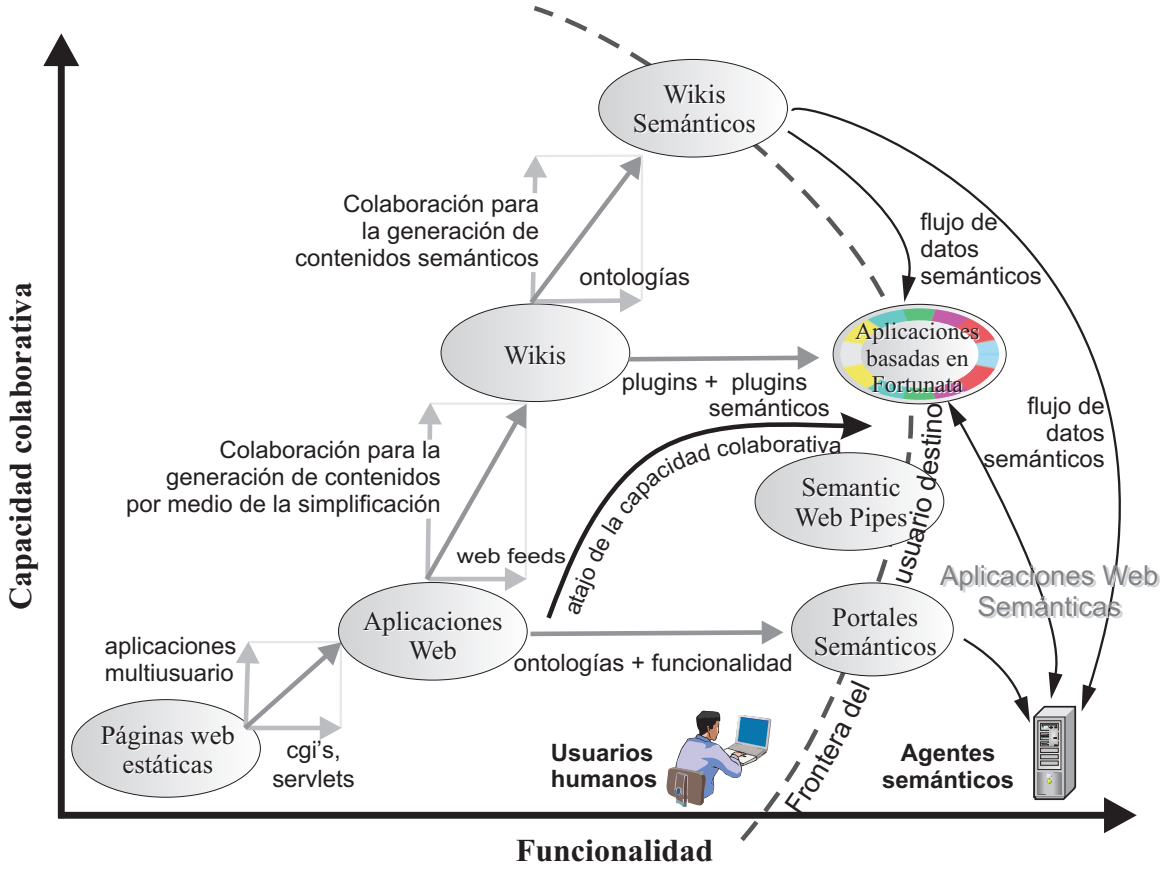

Figura 4.1: Situación de Fortunata en comparación con otras plataformas para creación de aplicaciones web semánticas.

plataforma Fortunata. Una aplicación creada con Fortunata se compone de un conjunto de plugins que integran datos semánticos de fuentes (externas, propias, o de otras aplicaciones basadas en Fortunata) que permiten la transformación de los datos de diversas formas, y/o proporcionan capacidades para solicitar o mostrar datos al usuario.

A diferencia de las técnicas tradicionales de desarrollo, en las que el código fuente está centralizado (típicamente en un repositorio común), las aplicaciones basadas en plugins se crean de una manera menos centralizada. Por ejemplo, si se desea ampliar la funcionalidad de un portal semántico o de un wiki semántico, el desarrollo tradicional requiere acceder al código fuente, modificarlo, y compilarlo, lo que genera una nueva versión de la aplicación. Si se usan plugins, los miembros de una comunidad de desarrolladores pueden contribuir a la creación de nueva funcionalidad con un menor grado de dependencia (por ejemplo, no

tienen que acceder al código de los otros para ampliar la funcionalidad de la aplicación). Cuando un desarrollador ha creado y probado un plugin nuevo, el código fuente es enviado al administrador del sitio web en el que está instalada Fortunata. Si el código es considerado válido y seguro, se compila y se añade al wiki-engine, y se pone a disposición de cualquier usuario o desarrollador.

Siguiendo el ejemplo anterior, después de un periodo breve de entrenamiento sobre Fortunata, el desarrollador puede crear su prototipo en poco tiempo (como se muestra en la sección de evaluación 6.1). El API de Fortunata oculta los detalles más propensos a errores, permitiendo a los desarrolladores centrar sus esfuerzos en proporcionar la funcionalidad requerida. Lógicamente, la creación de aplicaciones más avanzadas que las que se han creado probablemente requiera competencias más avanzadas en tecnologías de la web semántica, pero tareas básicas y proclives a errores como la presentación web o la publicación de datos semánticos y/o ontologías se delegan en Fortunata.

4.2. Fortunata, plataforma de funcionalidad colaborativa

Esta sección describe técnicamente la arquitectura, las características que proporciona Fortunata, y un caso de uso para desarrolladores.

4.2.1. Arquitectura y características de Fortunata

Como se ha mencionado en la introducción, Fortunata intenta combinar las ventajas de los wiki-engines y de los sistemas de gestión de ontologías, proporcionando a los creadores de aplicaciones web semánticas las siguientes características:

- **Presentación al estilo wiki**. Los desarrolladores no requieren competencias en tecnologías web cliente (e.g., HTML, CSS, o AJAX), tan sólo sintaxis wiki que puede aprenderse en un breve periodo de tiempo (unos 10 minutos para la sintaxis básica). Esto se debe a que Fortunata proporciona a los desarrolladores un conjunto sencillo de métodos capaces de mostrar “texto wiki”.
- **Gestión de datos semánticos**. Los sistemas de gestión de ontologías como Jena, Sesame, etc. suelen proporcionar a los desarrolladores unas APIs muy ricas y extensas que permiten crear y manejar ontologías y datos semánticos. Fortunata proporciona un conjunto muy simple de métodos, fáciles de aprender para los desarrolladores.

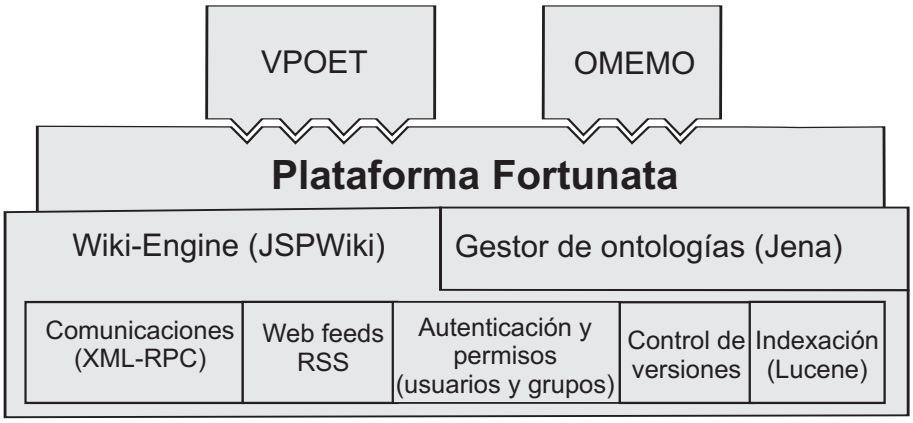

Figura 4.2: Arquitectura de Fortunata.

- **Utillaje**. Algunas limitaciones de JSPWiki, como el intercambio de datos entre páginas wiki, o un tipo unificado de mensajes de error, se salvan usando esta funcionalidad adicional.

Para proporcionar todas estas características, Fortunata ha sido creada sobre la funcionalidad proporcionada por el wiki-engine JSPWiki y la funcionalidad del sistema de gestión de ontologías Jena, como se muestra en la figura 4.2.

Se seleccionó JSPWiki, de entre los muchos wiki-engine disponibles [2], debido a su capacidad para crear formularios y su extensibilidad por medio de plugins, así como por estar basado en tecnologías bien establecidas como Java Server Pages (JSP).

A continuación se describen los distintos componentes software de esta plataforma, justificando su utilización en Fortunata.

JSPWiki [3] es un wiki-engine open-source escrito en el lenguaje Java que permite el uso de plugins para extender su funcionalidad. Algunos plugins pertenecen a la librería "core" (mantenidos por la comunidad de desarrolladores de JSPWiki), mientras que otros son "contribuciones" (mantenidos por los creadores de los plugins). La sencillez para extender la funcionalidad fue una de la razones para usar este wiki-engine.

Además de la extensibilidad, JSPWiki proporciona otras características que pueden ser explotadas por los desarrolladores que usen Fortunata así como por los usuarios de las aplicaciones creadas con Fortunata. Estas ventajas se describen brevemente como sigue:

[2] Véase `http://www.wikimatrix.org/` o `http://en.wikipedia.org/wiki/Comparison_of_wiki_software`

[3] Véase `http://jspwiki.org`

- Web feeds. Basados en el estándar RSS, proporcionan a los usuarios un mecanismo de suscripción a las páginas wiki. Por ejemplo, un usuario suscrito a una página wiki dada (que puede ser generada a partir de una fuente de datos semánticos, y que por tanto represente un cambio en los datos de la fuente) será notificado cada vez que la página cambie.

- Control de acceso y permisos. Permite gestionar el acceso al wiki a individuos y grupos, así como los permisos para ver, modificar, o borrar (entre otros) una página wiki, funcionalidad muy común en cualquier aplicación web (semántica).

- Control de versiones. Permite volver una página wiki a cualquier estado previo. Esta característica anima a los usuarios a modificar las páginas wiki con la garantía de que se puede deshacer cualquier cambio.

- Gestión de enlaces. Proporciona mecanismos para identificar enlaces "nulos" (enlaces que apuntan a páginas wiki que no existen), enlaces "inversos" (páginas wiki que enlazan con la página wiki actual), o enlaces "huérfanos" (páginas wiki no enlazadas por ninguna página wiki).

- Indexación. Implementada por la herramienta Lucene, permite búsqueda por palabra clave sobre el conjunto de las páginas gestionadas por el wiki-engine, independientemente de si han sido creadas manualmente o automáticamente (por ejemplo, por aplicaciones como OMEMO).

Jena [4] es una librería Java que proporciona a los desarrolladores un entorno de programación (API) para la gestión de ontologías y datos semánticos. Permite gestionar ontologías descritas en distintos lenguajes, como OWL y RDFS, así como variados modelos de persistencia y razonamiento. Fortunata reduce esta diversidad de modelos a un pequeño conjunto de opciones. Por ejemplo, la implementación que usaron los desarrolladores que evaluaron Fortunata (véase sección 6.1) usaba un modelo de persistencia basado en ficheros, algo bastante habitual en sistemas wiki, y ontologías OWL.

Con esa configuración, los métodos proporcionados por Fortunata evitan a los desarrolladores tareas como la gestión de ficheros, modelos Jena, etc. El desarrollador sólo tiene que proporcionar código para la creación de cada dato semántico (instancia) y el código

[4] Véase `http://jena.sourceforge.net`

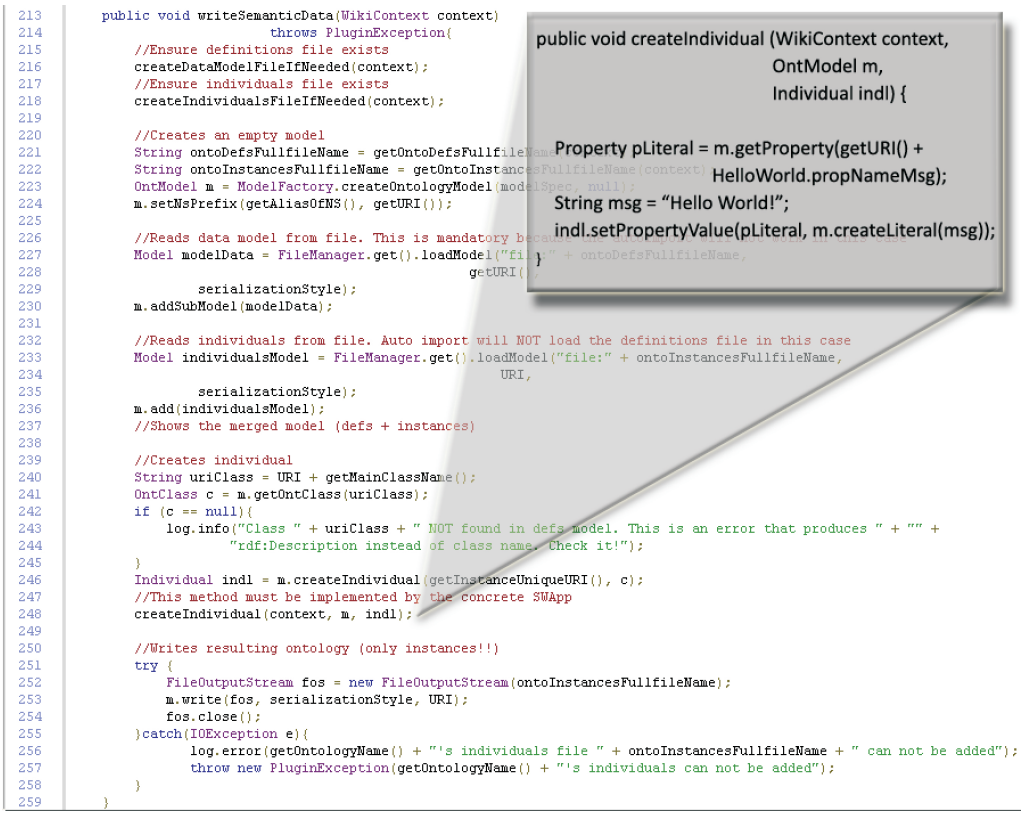

Figura 4.3: Comparativa del código requerido para almacenar (y publicar) una instancia. El recuadro muestra el código que tiene que proporcionar un desarrollador que utilice Fortunata para la aplicación de ejemplo “Hello Word”. El código mostrado en el fondo lo proporciona Fortunata.

de creación de la ontología (clases y propiedades). La figura 4.3 muestra el código que tiene que proporcionar un desarrollador para crear una instancia de la aplicación de ejemplo “Hello World” proporcionada en el manual de desarrolladores [5]. Conviene comparar esas 3 líneas con la cantidad de código necesario para proporcionar esta funcionalidad (en este caso proporcionada por Fortunata), mostrado en el fondo de la figura.

4.2.2. Plugins como elementos colaborativos

En esta sección se muestra el proceso de creación de plugins para JSPWiki, y a continuación el proceso de creación de plugins para Fortunata (F-plugins).

[5] Véase `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=HelloWorldSWApp`

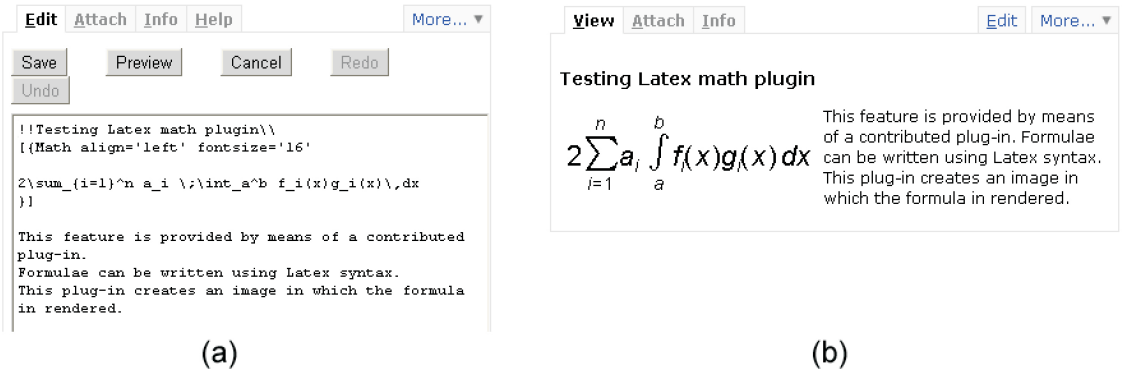

Figura 4.4: Ejemplo de uso de un plugin de JSPWiki que genera una imagen a partir de una fórmula LaTeX. La página wiki que lo contiene muestra (a) el "modo edición", en el que se muestra cómo se invoca el plugin, y (b) el "modo vista", que muestra el resultado.

Contribución de funcionalidad por medio de plugins en JSPWiki

Los plugins de un wiki son piezas de código que extienden la funcionalidad básica del wiki-engine: la edición de páginas wiki. Los plugins automatizan acciones en una página wiki, y presentan el resultado de la acción en "modo vista". Ejemplos de estos plugins en JSPWiki son: TableOfContents, que genera una tabla de contenidos a partir del contenido de la página donde se encuentra el plugin, o ReferringPages, que muestra la lista de todas las páginas que refieren a la página donde se encuentra el plugin. En la fecha de escritura de este documento (abril de 2009), JSPWiki proporciona 25 plugins "core", como los mencionados anteriormente, y alrededor de cien "contributed".

La figura 4.4 muestra un ejemplo de cómo invocar un plugin disponible en el sistema desde una página wiki (parte izquierda de la figura), y el resultado de la invocación con un conjunto específico de parámetros (parte derecha). El texto para invocar el plugin se encuentra entre"[{" y "}]". Este plugin contiene argumentos para especificar el tamaño de letra, la alineación, así como un cuerpo en el que se especifica una fórmula en formato LaTeX. El resultado de esta invocación es la página wiki en "modo vista" que se muestra en la derecha. Este plugin se ejecuta automáticamente cada vez que se muestra ("modo vista") la página wiki que lo contiene. El resultado de la ejecución es una imagen que muestra la fórmula.

En JSPWiki, los formularios están relacionados con los plugins. Cada vez que se envía (submit) un formulario, el formulario envía sus datos al plugin indicado. La ejecución del

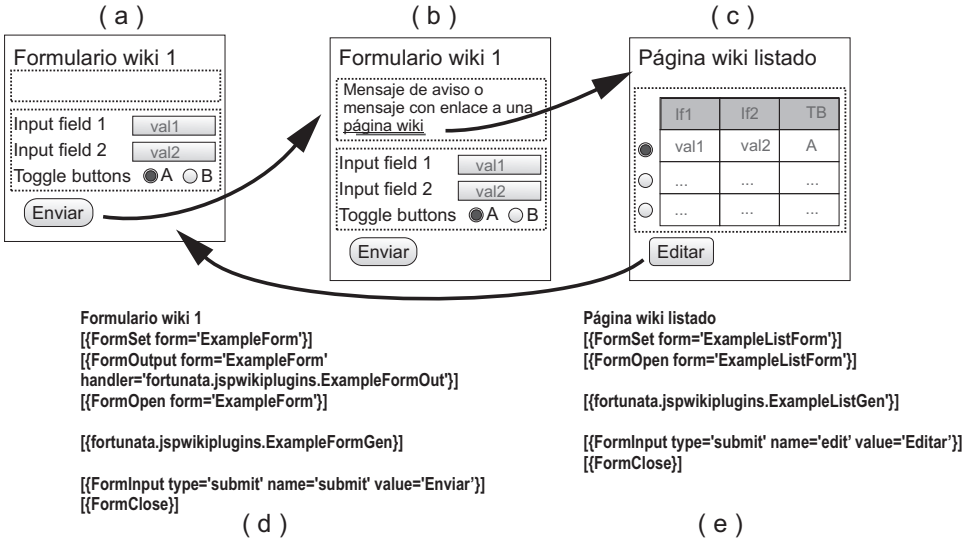

Figura 4.5: Ciclo interactivo tipo usado en las aplicaciones basadas en Fortunata.

plugin permite:

- leer la información proporcionada por el usuario en el formulario
- hacer las operaciones adecuadas
- mostrar el resultado al usuario

A diferencia de las aplicaciones web tradicionales, en las que los resultados se muestran en una nueva página, en JSPWiki no existen esas transiciones. El resultado se muestra en la misma página wiki, como se muestra en las partes (a) y (b) de la figura 4.5.

Esta limitación se resuelve usando la estrategia mostrada en la figura 4.5. Esta figura se compone de 5 partes: (a)-(e). El wiki engine convierte el código wiki mostrado en la parte (d) en el formulario mostrado en (a). Una vez que se introducen los datos, el usuario pulsa el botón “Enviar” y el plugin `ExampleFormOut` es invocado (línea 4 en (d)).

El panel de mensajes (línea discontinua en (b) en la parte superior del formulario) muestra el resultado de la validación de los datos: un mensaje de error si no son válidos, o un mensaje confirmando la ejecución. Además, ese mensaje de confirmación incluye un enlace a la página wiki que muestra los datos introducidos. Esta página wiki (c) ha sido generada por el plugin `ExampleFormOut`, y los datos mostrados son generados dinámicamente por otro plugin (`ExampleListGen`, en la línea 4 de (e)). Hay que destacar la simplicidad del

código wiki. Este aspecto es esencial, ya que el texto wiki debe ser generado por el desarrollador del plugin. En este caso, el desarrollador del plugin `ExampleFormOut` debe generar una página wiki con el código mostrado en (e).

Para crear un plugin JSPWiki, los desarrolladores sólo tienen que crear una clase Java que implemente el interfaz *WikiPlugin* (véase figura 4.6). Este interfaz tan sólo requiere la implementación del método `execute()`. Este método será invocado por el wiki-engine en la ejecución del plugin, típicamente cuando un usuario ve la página wiki que lo contiene. Dentro de ese método, los desarrolladores tienen acceso a los parámetros del plugin y sus valores, así como a todos los datos de sesión, permisos, y configuración del wiki-engine; y deben incluir el código que realiza la funcionalidad del plugin, típicamente invocando métodos de una librería propia.

Desarrollo de F-plugins

Al igual que los plugins de JSPWiki, los F-plugins se implementan por medio de una clase Java (véase figura 4.6) que implementa el interfaz *WikiPlugin* (perteneciente a la librería de JSPWiki), pero adicionalmente extiende la clase `FortunataPlugin` (de la librería Fortunata)) proporcionando a los desarrolladores métodos útiles (e.g. `renderWikiText()`) para la gestión de formularios y presentación de texto wiki. Una aplicación web semántica se representa por medio de una clase derivada de la clase abstracta *FortunataSWApplication*, que proporciona a los desarrolladores con métodos útiles a la vez que les obliga a implementar los métodos `createIndividual()` y `fillDataModel()` relativos a la persistencia de los datos y la(s) ontología(s). Todos los plugins que componen una aplicación web semántica basada en Fortunata comparten una de estas clases *FortunataSWApplication*. En este ejemplo, la figura muestra la clase `AddVisualization`. Esta clase es un F-Plugin y, consecuentemente, hereda los métodos implementados en la clase base *FortunataPlugin* y se ve forzada a implementar tres métodos (uno del interfaz *WikiPlugin* y dos de la clase `FortunataPlugin`). Este plugin contiene una instancia de la clase `Vpoet`, que implementa dos métodos de la clase `FortunataSWApplication` relativos a la gestión de los datos semánticos.

El proceso para crear y contribuir un F-plugin se detalla en la parte superior de la figura 4.7, y sigue el procedimiento habitual de las arquitecturas basadas en wikis. Primero, el desarrollador debe crear el F-plugin localmente (pasos 1 a 3) y realizar un adecuado número de tests para comprobar que funciona correctamente (paso 4). Entonces debe publicar

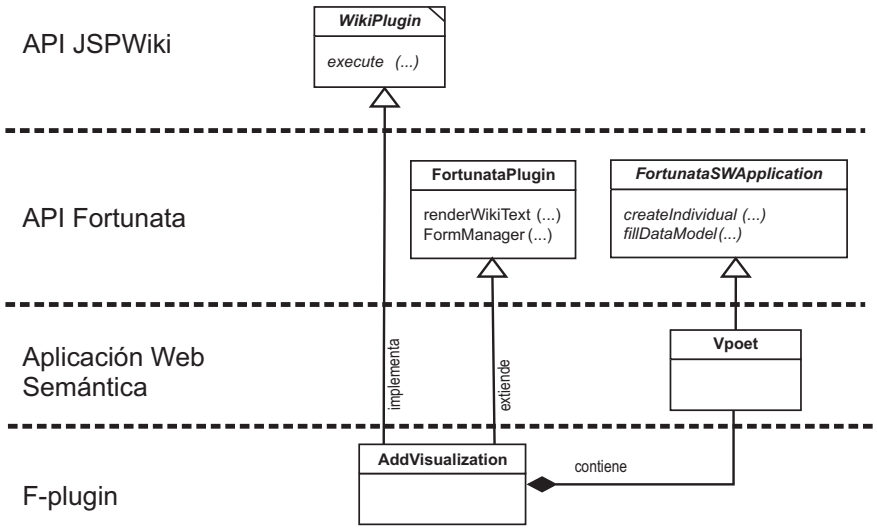

Figura 4.6: Diagrama de clases de las aplicaciones basadas es Fortunata. La capa "Fortunata API" los principales métodos que un desarrollador debe implementar. En este ejemplo, el F-plugin contiene una instancia de la clase Vpoet. Las clases abstractas y los interfaces estás escritos en cursiva.

(paso 5) el código fuente del plugin y la documentación de su uso. La parte inferior de la figura muestra el proceso necesario para crear nueva funcionalidad reutilizando la funcionalidad inicial siguiendo el esquema de la "colaboración contributiva". Se compone de los siguientes pasos: instalación de un F-plugin existente (paso 6), lectura y comprensión de sus ontologías asociadas (leyendo manualmente los ficheros OWL, usando un editor de ontologías estándar, o por medio de una aplicación especializada en este tipo de tareas como OMEMO) con el fin de encontrar los elementos que deben ser añadidos, eliminados o modificados, o bien decidir si se debe importar y usar un nuevo conjunto de ontologías (paso 7), la creación local del plugin ampliado (pasos 8 a 10), tests locales (paso 11) y publicación (paso 12). Esta forma de realizar contribución de código muestra la baja complejidad del proceso de reutilización y contribución de F-plugins, y por tanto la simplicidad del proceso colaborativo.

La tabla 4.2 resume las tareas de desarrollo asociadas al desarrollo de una aplicación web semántica típica, y compara las competencias necesarias para realizar estas tareas cuando se usa (a) un proceso de desarrollo típico, y (b) un proceso de desarrollo basado en

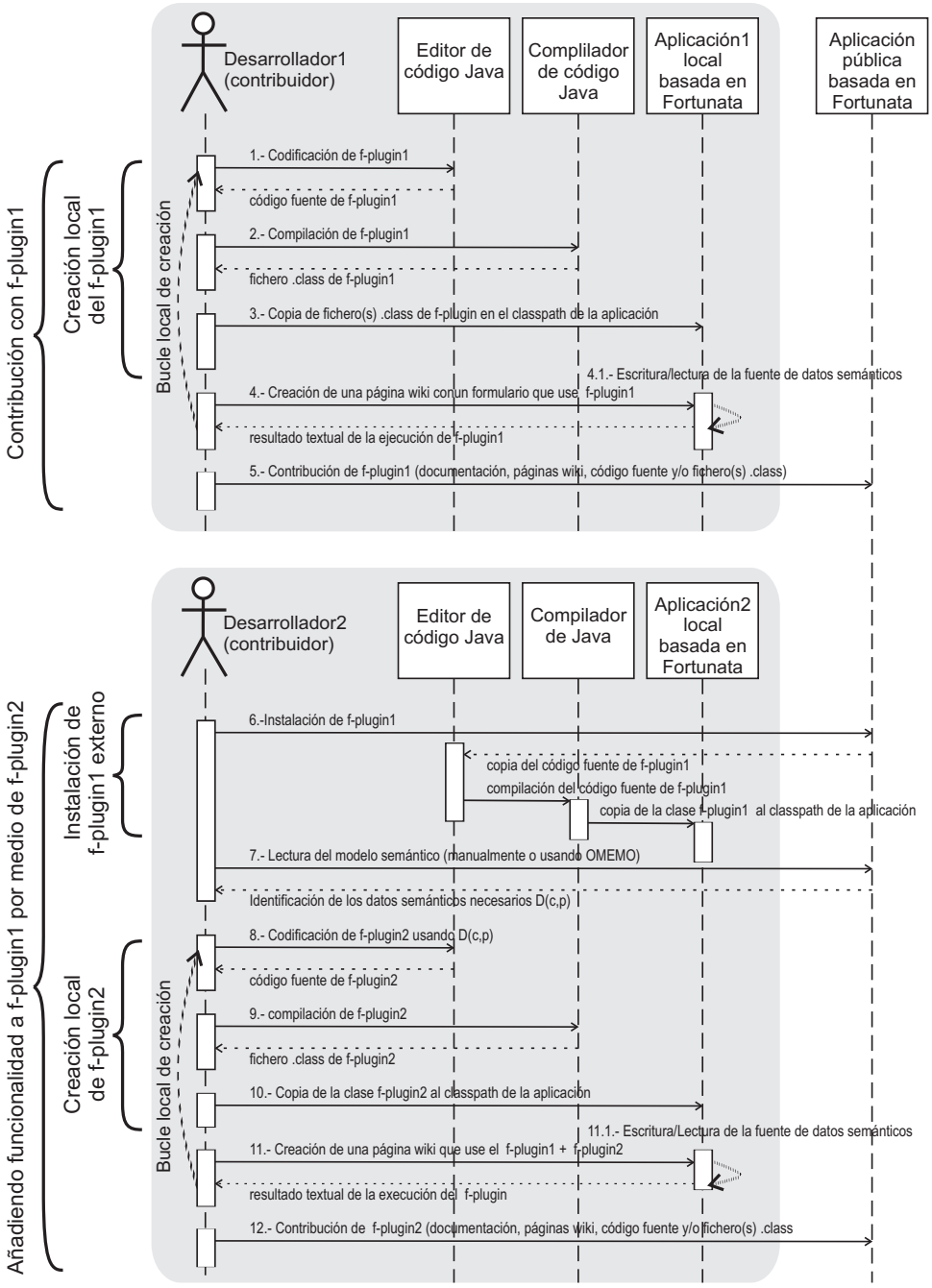

Figura 4.7: Diagrama de secuencia de una contribución. Desarrollador1 crea y prueba (localmente) el f-plugin1. Una vez finalizado, se contribuye este plugin. Developer2 aprovecha esta contribución y extiende la funcionalidad por medio de f-plugin2, que también acaba siendo contribuido.

Tabla 4.2: Comparación entre el desarrollo tradicional de aplicaciones web semánticas y el desarrollo basado en Fortunata.

Tarea	**Desarrollo tradicional**	**Desarrollo basado en Fortunata**
Creación de páginas	CMS (Contents Managing System)	Wiki
Creación of formularios	HTML, CSS, DOM, AJAX	Formularios wiki
Creación de aplicaiones web	tecnologías JSP, JSF, .NET	Wiki-engine
Permisos, autenticación	Administrador (servidor de) Aplicaciones Web	Wiki-engine
Creación de ontologías	Jena	Fortunata
Creación de datos semánticos	Jena	Fortunata
Publicación de ontologías y datos semánticos	Jena + Administración Servidor Web	Fortunata

Fortunata. El proceso de desarrollo tradicional requiere más competencias (más herramientas de desarrollo y roles) que el proceso de desarrollo basado en Fortunata. Este es uno de los principales resultados de la comparación realizada con desarrolladores reales, como se describe en las diferentes evaluaciones realizadas (sección 6.1).

4.3. Conclusiones

En este capítulo se ha mostrado una plataforma de programación simple y fácilmente extensible, llamada Fortunata, que facilita la creación de aplicaciones web semánticas de manera colaborativa. Esta colaboración no se centra en compartir un código fuente centralizado, sino en la donación (contribución) de funcionalidad por medio de plugins de un wiki-engine. Las ventajas de Fortunata se pueden resumir en (1) entorno de desarrollo sencillo y colaborativo, (2) mecanismos para reutilización de funcionalidad contribuida, y (3) mínimas dependencias entre desarrolladores.

Para lograr estos objetivos, Fortunata aprovecha la funcionalidad de un wiki-engine (JSPWiki) y un gestor de ontologías (Jena), simplificando su uso mediante la simplificación del interfaz de programación (API), lo que permite que desarrolladores con menores

competencias en tecnologías de la Web Semántica puedan crear F-plugins. Fortunata ha sido evaluado con el fin de comprobar que las aplicaciones construidas con esta plataforma tienen unos valores razonables de usabilidad. Esa evaluación permitió mejorar el API y sirvió para generar una guía de usabilidad para desarrolladores de aplicaciones basadas en Fortunata.

Con esta plataforma se han creado dos aplicaciones (véase capítulo 5) llamadas OMEMO y VPOET que han sido probadas con usuarios reales. Una tercera, llamada MIG 5.4.3 se encuentra aún en fase de desarrollo y pruebas. Los experimentos realizados con estas aplicaciones muestran que Fortunata es una plataforma válida para crear aplicaciones web semánticas reales basadas en componentes reutilizables que reduce los tiempos de desarrollo y las competencias requeridas a los desarrolladores de las mismas.

Sin embargo, estas aplicaciones también muestran las limitaciones de Fortunata, debidas principalmente a las limitaciones heredadas de JSPWiki. Algunas de estas últimas se han podido superar, como es el caso de la navegación desde formularios, pero otras dependen de su resolución en futuras versiones de este wiki-engine. También hay limitaciones debidas al modelo de aplicación web semántica propuesto, por ejemplo, la persistencia de los datos semánticos se realiza por medio de ficheros, con un mal rendimiento cuando hay muchos datos semánticos (por encima de decenas de miles de tripletas). El API de Fortunata puede incorporar modelos alternativos que aumenten el rendimiento antes situaciones como las descritas, aunque estos no han sido actualmente desarrollados.

Capítulo 5

Uso de datos semánticos por usuarios finales

Este capítulo muestra dos aplicaciones web semánticas, llamadas OMEMO y VPOET, creadas con la plataforma Fortunata. Estas aplicaciones han permitido comprobar la aplicabilidad de la plataforma y corroborar con usuarios reales la evaluación de Fortunata. Han servido para completar la funcionalidad del API de Fortunata, detectar limitaciones del wiki-engine, y comprobar que Fortunata proporciona una infraestructura útil para el desarrollo de este tipo de aplicaciones.

OMEMO genera páginas wiki que describen los componentes de una ontología dada pero, a diferencia de otras herramientas que proporcionan una funcionalidad similar, esta descripción es más sencilla, y está orientada a mostrar la estructura de los componentes de que consta la ontología. Esta descripción es la que necesitan los usuarios de VPOET para crear plantillas capaces de presentar o editar datos semánticos de cada uno de estos componentes. Estas plantillas pueden ser usadas por los desarrolladores web, sin requerir conocimientos de las tecnologías de la web semántica, de forma sencilla mediante mensajes sobre HTTP. Como ejemplo de uso de este repositorio de plantillas se ha creado un Google Gadget que permite que los usuarios finales puedan añadir datos semánticos a sus páginas web.

5.1. Introducción

En el capítulo 4 se mostró la plataforma Fortunata, que permite a los desarrolladores con conocimientos de las tecnologías de la Web Semántica crear aplicaciones sin requerir conocimientos avanzados de dichas tecnologías ni conocimientos especializados de las tecnologías web cliente. Por tanto, aunque Fortunata reduce el nivel de competencias de los desarrolladores, sigue requiriendo unos conocimientos básicos de las tecnologías de la Web Semántica. Estos desarrolladores deben ser capaces de usar un API que, aunque oculta mucha de la complejidad de Jena, obliga al desarrollador a utilizar clases y métodos de Jena, por lo que el número de desarrolladores cualificados para crear este tipo de aplicaciones, aunque puede aumentar gracias a Fortunata, sigue siendo reducido.

El desarrollador común no tiene conocimientos de las tecnologías de la Web Semántica, por lo que, idealmente debe disponer de herramientas que le permitan manejar datos semánticos en sus aplicaciones web sin necesidad de dominar estas tecnologías. Al igual que los portales web semánticos ocultan a sus usuarios la gestión de ontologías y datos semánticos, VPOET y OMEMO consiguen que los desarrolladores web puedan manejar

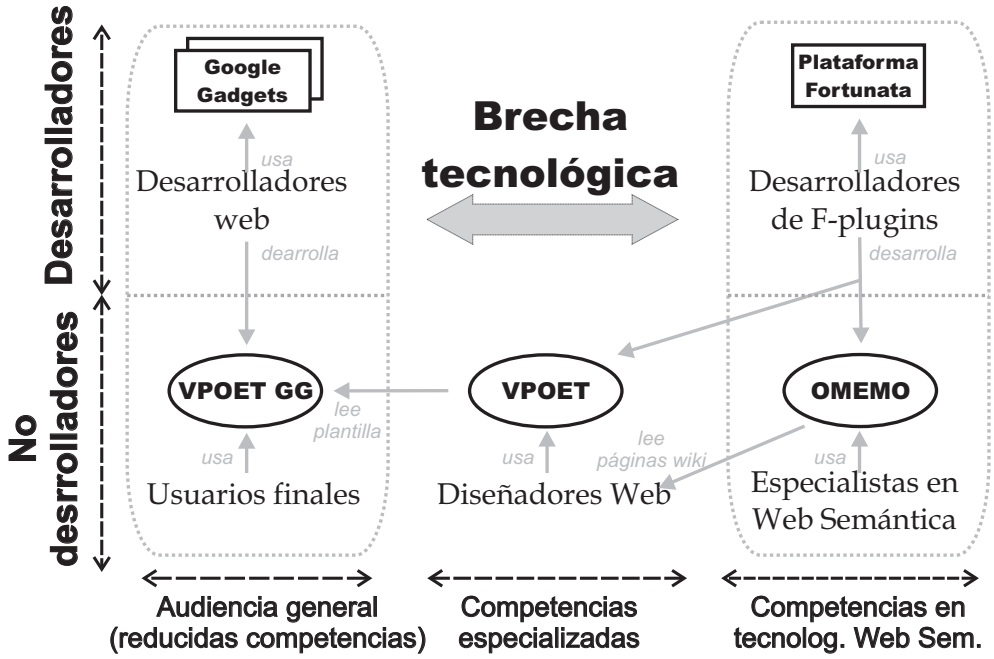

Figura 5.1: Esquema de la estrategia seguida para reducir la brecha tecnológica existente entre los expertos en tecnologías de la Web Semántica y los usuarios y desarrolladores no expertos en estas tecnologías.

ontologías y datos semánticos sin necesidad de conocer las tecnologías subyacentes.

En la sección 3.7.1 se bosquejó la estrategia seguida para acercar las posiciones de dos tipos de desarrolladores con competencias muy diferentes. La figura 5.1 muestra esta estrategia de forma esquemática, indicando las competencias y perfiles de las partes implicadas. Fortunata acerca a la Web a los desarrolladores de la Web Semántica, y VPOET (con la ayuda de OMEMO y de los diseñadores web) acerca la Web Semántica a los desarrolladores de aplicaciones Web.

Aunque OMEMO es una aplicación independiente, ha sido diseñada para ser explotada por los usuarios de VPOET. A continuación se muestra una visión general y un ejemplo de uso de cada una de estas aplicaciones.

5.2. Ontologies for MEre MOrtals (OMEMO)

OMEMO es una aplicación basada en Fortunata, dirigida a usuarios sin conocimientos de lenguajes de ontologías. Usando OMEMO, los usuarios pueden saber qué componentes

(clases, propiedades, y relaciones) se definen en una ontología dada.

Cualquier usuario es libre de añadir al repositorio de OMEMO cualquiera de las ontologías existentes en la red (escritas en RDFS o OWL), pero deberá saber la URL de la ontología, su namespace, o el lenguaje es el que está escrita. Consideramos que un usuario típico de OMEMO no debe tener esos conocimientos, sino que es más propio de especialistas en tecnologías de la Web Semántica (parte derecha de la figura 5.1), por lo que esa tarea se reserva para este último tipo de usuario.

Cuando un usuario de OMEMO añade una nueva ontología, se genera automáticamente un conjunto de páginas wiki. Conviene mencionar que una ontología puede tener diversas versiones, distinguibles unas de otras por su fecha de publicación. Por ejemplo, supongamos que OMEMO conoce dos de las versiones de la ontología FOAF (20050403 y 20050603).

El proceso de generación de páginas resulta en una página wiki para la ontología, con enlaces a sus distintas versiones y todos sus componentes, así como una página por cada clase, propiedad o relación. La infraestructura proporcionada por JSPWiki se encarga de proporcionar los enlaces directos o inversos entre todas estas páginas. Siguiendo este ejemplo, la figura 5.2 muestra una sección de la página wiki generada para la clase `Person` (versión 20050403). El punto ❹ indica que hay otra versión de la ontología FOAF, permitiendo acceder a sus páginas. La propiedad *interest* es de tipo `Document` ❸ (definido en el mismo FOAF), pero la propiedad *firstName* ❶ es de tipo `Literal` ❷ (definido en la ontología RDFS). En el caso de que la ontología RDFS esté almacenada en OMEMO (como así es), aparecerá un enlace (subrayado continuo) a la página wiki de `Literal`. Caso de no estar almacenada aparecerá como un enlace a página inexistente (subrayado discontinuo). La lista numerada de la figura 5.2 muestra el nombre de las páginas wiki apuntadas. La sintaxis de esos nombres aporta una serie de ventajas que se detallan más adelante.

Hay que destacar que las páginas generadas son indexadas automáticamente por JSPWiki usando Lucene [1], como si fueran páginas creadas manualmente; por lo que las facilidades para la búsqueda proporcionadas por JSPWiki también incluyen al texto de estas páginas. Cualquier página creada manualmente puede apuntar a cualquiera de las mencionadas anteriormente. La única restricción de las páginas generadas es que no se pueden editar. El texto wiki de estas páginas indica a JSPWiki que no son editables, por lo que el botón “Editar” desaparece de la página wiki. A continuación se muestra un caso práctico

[1] Véase Lucene Project Page en `http://lucene.apache.org/`

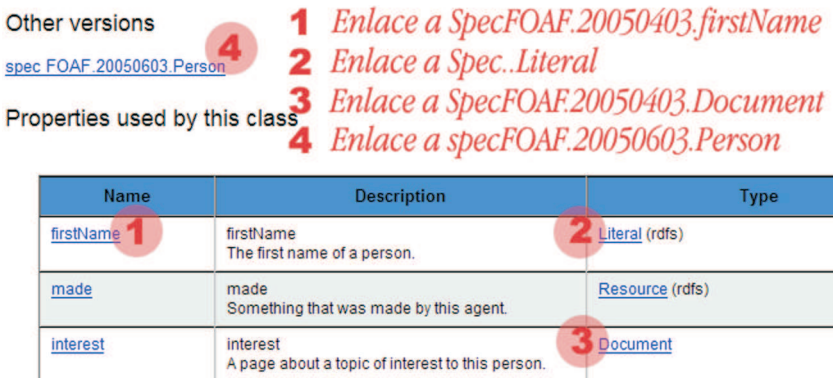

Figura 5.2: Captura de pantalla de la página wiki generada por OMEMO para la clase `Person` (versión 20050403)

de utilización de OMEMO.

5.2.1. Caso de uso de OMEMO

Siguiendo la figura 5.1, supongamos un especialista en Web Semántica usa su navegador web para ver las páginas wiki generadas por OMEMO, en busca de detalles acerca de las ontologías almacenadas en el sistema, o añadir ontologías nuevas. La parte izquierda de la figura 5.3 muestra la página wiki "Add Ontology" con el formulario para añadir ontologías, mientras que la parte derecha muestra la página wiki "Stored Ontologies". Esta última página tiene algunas características que merecen destacarse:

- Cada ontología muestra la última comprobación ("last check"), esto es, cuánto tiempo hace que se comprobó su vigencia. Un temporizador comprueba periódicamente si la ontología original (ubicada en la URL mostrada por la segunda columna en la parte derecha de la figura 5.3) ha cambiado. En ese caso, se descarga la nueva versión y se generan nuevas páginas.

- "RDF Icon" tiene un enlace a los datos RDF usados para construir esta página.

Para añadir una nueva ontología, el especialista en Web Semántica rellena el formulario con algunos datos obligatorios tales como (1) URL en al que se encuentra la ontología, (2) sobrenombre (alias) de la ontología, y (3) la fecha de la versión. Una vez que todos estos campos son rellenados y se pulsa el botón "Submit", el proceso de generación comprueba

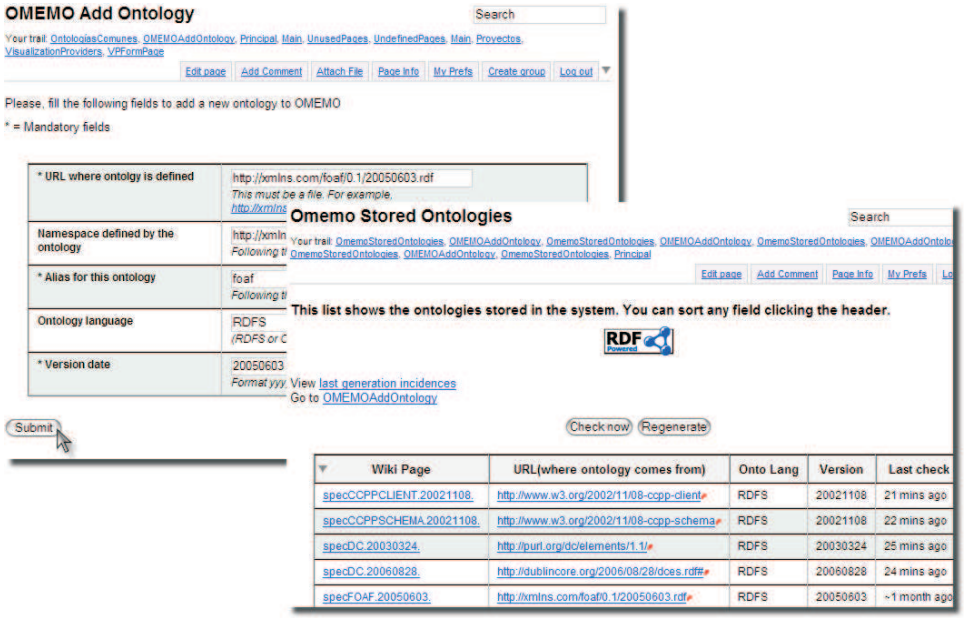

Figura 5.3: Páginas wiki de OMEMO. Izda.: formulario para añadir ontologías. Dcha.: lista de ontologías gestionadas por OMEMO

algunos posibles conflictos y, si todo es correcto, se genera un nuevo conjunto de páginas wiki. El proceso completo, incluyendo las comprobaciones más importantes es el siguiente:

1. Se establece una conexión HTTP con la URL especificada. El fichero que contiene la ontología es descargado y almacenado en una carpeta temporal.

2. Se analiza la ontología para detectar los *namespaces* utilizados. Esto permite enlazar diferentes ontologías entre sí. Por ejemplo, la ontología `FOAF` refiere a la ontología `RDFS` en la definición de la propiedad `FOAF:firstName`, especificando que es un `RDFS:Literal` (véase figura 5.2, primera fila de la tabla).

3. Comprobación 1: Conflicto con prefijos tales como prefijos en blanco (*namespaces* que no definen prefijo), los prefijos duplicados (la ontología O_1 defines el prefijo p para el *namespace* n, y la ontología O_2 define el mismo prefijo para un *namespace* diferente), y los *namespaces* sobreescritos (la ontología O_1 define el *namespace* n con el prefijo p_1, y la ontología O_2 define ese *namespace* con el prefijo p_2).

4. Comprobación 2: Versiones de ontologías. La ontología O_1 en la URL_1, y la ontología O_2 en la URL_2, definen el mismo *namespace*, pero su contenido es diferente. Por

ejemplo, la ontología FOAF tiene (entre otras) las versiones 20050**603** y 20050**403**. En la versión **603** se define las propiedades *isPrimaryTopicOf* y *birthday*, que no existen en la versión **403**.

5. Comprobación 3: Ontologías duplicadas. La misma ontología puede tener diferentes URLs. Este es el caso de la ontología Dublin Core[2] (DC), que se puede encontrar en dos URLs distintas: `http://purl.org/dc/elements/1.1` y `http://dublincore.org/2006/08/28/dces.rdf#`. Sin esta comprobación, esto generaría dos entradas diferentes (dc.20030324 y dc.20060824), pero el contenido sería el mismo.

Se han implementado soluciones para la mayor parte de estos problemas, y se ha creado una sintaxis que identifica unívocamente a cada componente, lo que resulta útil para asignar el nombre a cada una de las páginas wiki generadas. Esta sintaxis tiene la forma `spec.prefix.version.comp`, y se puede ver un ejemplo de su uso en la lista numerada de la figura 5.2.

5.3. Visualization Prviders for Ontology ElemenTs (VPOET)

VPOET es una aplicación basada en Fortunata compuesta por 4 páginas wiki y siete F-plugins. VPOET permite crear plantillas web para cualquier componente de una ontología. Estas plantillas pueden estar diseñadas para mostrar datos semánticos (plantillas de salida) o para solicitar datos a través de un formulario (plantillas de entrada).

Aunque VPOET puede ser usado por cualquier usuario con conocimientos básicos de las tecnologías web cliente, ha sido creado para permitir que diseñadores web profesionales puedan crear diseños atractivos capaces de obtener (de un usuario) o mostrar (a un usuario) datos semánticos. Los usuarios de la aplicación VPOET se denominan en este contexto "Proveedores de Plantillas", y es un término equivalente a "diseñador web". Ambos serán usados indistintamente en este documento.

Desde el punto de vista de un proveedor de plantillas, esta aplicación es como cualquier otra aplicación web, con elementos de formularios tales como campos de texto, selectores, o botones. Los proveedores de plantillas sólo tienen que leer un tutorial on-line [3] para

[2] Véase `http://dublincore.org/`

[3] Véase el tutorial en `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=VPOETTutorial`

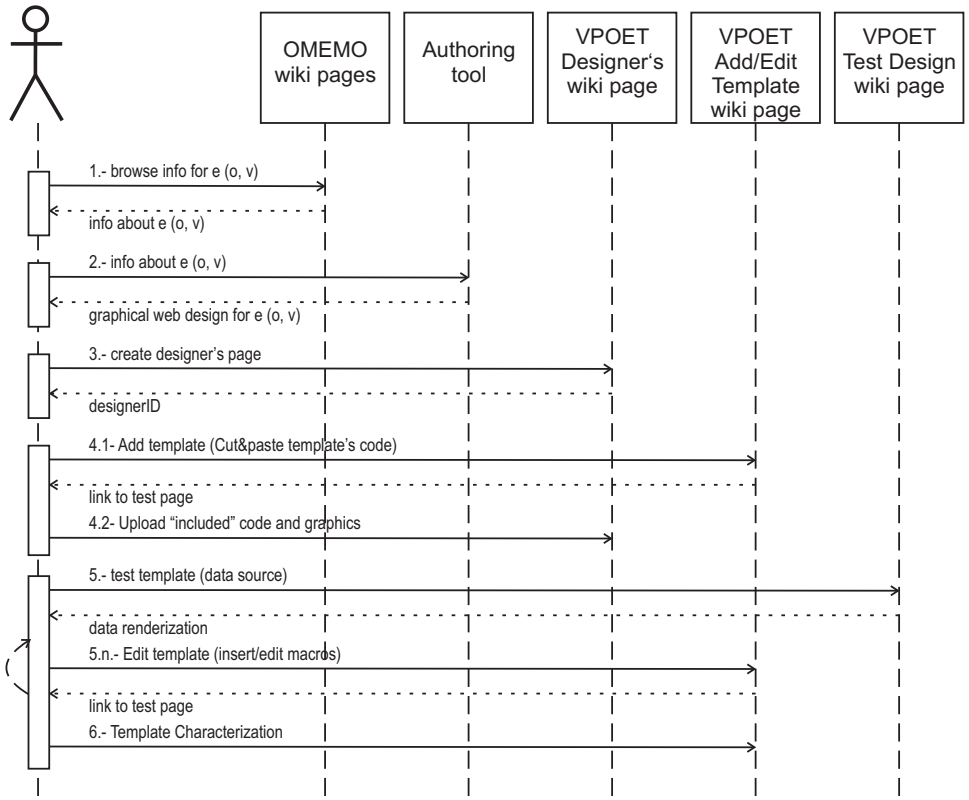

Figura 5.4: Diagrama de secuencia para los usuarios de VPOET

empezar a crear plantillas. Sin embargo, por ser una aplicación basada en Fortunata, la información introducida por el usuario es publicada como datos semánticos. La sección 5.4 muestra cómo estos datos semánticos pueden ser explotados por los desarrolladores de aplicaciones web para manejar datos semánticos.

5.3.1. Creación de plantillas de salida

El proceso de creación de una plantilla comienza seleccionando un componente de una ontología y, como muestra el diagrama de secuencia de la figura 5.4, se compone de los siguientes pasos:

1. Obtener información acerca de la estructura del componente seleccionado. Esto es, saber de qué sub-componentes consta el componente seleccionado. El proveedor de plantillas obtiene esta información leyendo las páginas wiki generadas por OMEMO para la ontología a la que pertenece el componente seleccionado.

2. Hacer un diseño gráfico para la web (diseño web) usando herramientas de autor (e.g. Dreamweaver) a partir de la información obtenida en el punto anterior.

3. Elegir un identificador de proveedor de plantillas (`designerID`) y crear una página wiki en VPOET con este identificador. Esta página contendrá información acerca del proveedor de la plantilla (TP) y de las plantillas que ha almacenado en VPOET.

4. El diseño gráfico se compone de un conjunto de archivos (imágenes, y código cliente como HTML, CSS o Javascript). El código cliente debe ser "copiado y pegado" en los campos adecuados de la página "Add design" de VPOET. Las imágenes y los archivos "included" por los ficheros HTML, CSS o Javascript deben ser adjuntados a la página wiki del proveedor de plantillas o subidos a cualquier servidor web. En cualquier caso, el código cliente debe tener los enlaces correctos a estos ficheros.

5. Comienza un ciclo de pruebas usando fuentes de datos semánticos (típicamente externas a VPOET) que contengan instancias del componente seleccionado.

   a) Los paths absolutos (e.g. a imágenes o a ficheros Javascript) deben ser sustituidos usando una macro específica (`OmemoBaseURL`). La tabla 5.1 muestra las macros más relevantes disponibles en VPOET, los argumentos que requiere cada una, y una breve explicación de su utilidad.

   b) Los datos semánticos son insertados en el diseño usando estas macros específicas.

   c) Se prueba el diseño con distintas fuentes de datos semánticos de pruebas (proporcionadas por terceras partes).

   d) Este ciclo termina cuando el diseño produce visualizaciones. satisfactorias para todas las fuentes de datos semánticos de pruebas.

6. El diseño es caracterizado por su creador, proporcionando datos acerca de propiedades del diseño tales como política de resize (tamaño fijo, expansiva, contractiva), colores principales, o comportamiento frente a cambios del tamaño de letra.

La mayor parte del esfuerzo requerido para crear una plantilla se localiza en el ciclo de pruebas, especialmente en la inserción de macros. Aunque pueda parecer un conjunto de macros muy reducido, ha demostrado ser suficiente para cubrir las necesidades de los

Tabla 5.1: Algunas de las macros disponibles para los proveedores de plantillas de VPOET.

Macro	Argumentos	Descripción
`OmemoGetP`	`propName, pre, post, sep`	Se sustituye por el valor de la propiedad `propName`
`OmemoBaseURL`	Sin argumentos	Se sustituye por la URL del servidor en el que se ejecuta OMEMO
`OmemoConditionalVizFor`	`propName, designerID, designID`	Muestra el valor de la propiedad `propName` sólo si tiene algún valor
`OmemoGetLink`	`relationName`	Se sustituye por un enlace capaz de mostrar componentes del tipo al que apunta la relación `relationName`

proveedores de visualización que han usado la aplicación. Se puede encontrar información adicional acerca del uso de estas macros en una sección específica [4] del tutorial de VPOET. En la subsección 5.3.3 se proporciona un ejemplo de uso detallado.

Reutilización de plantillas

VPOET ha sido diseñado para permitir a los proveedores de plantillas reutilizar plantillas propias y ajenas, de forma que es posible la composición de plantillas. Esto se logra de dos formas distintas: (1) mostrando condicionalmente una propiedad (macro `OmemoConditionalVizFor`) y (2) enlaces capaces de mostrar los componentes destino de una relación (macro `OmemoGetLink`). Se describen en detalle a continuación.

Cuando una propiedad dada no tiene valor, la macro `OmemoGetP` se expande a la cadena "N.A". Para evitar este efecto, la macro `OmemoConditionalVizFor` sólo muestra el valor de la propiedad si existe, por medio de la plantilla indicada en la macro. La macro `OmemoGetLink` ofrece una funcionalidad similar, pero para relaciones que enlazan unas instancias con otras. Por ejemplo, si se crea una plantilla que muestre direcciones de correo electrónico mediante "ofuscación de Javascript" (una técnica que impide que pueda ser leída, muy usada para evitar correo basura), se puede reutilizar esta plantilla cuando una propiedad sea de tipo email.

La figura 5.6 muestra un ejemplo avanzado de código fuente de una plantilla para el

[4] Véase `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=MacrosInVPOET`

Figura 5.5: Ejemplo avanzado de código fuente de una plantilla VPOET. Los rectángulos gruesos enmarcan las macros, y los finos la reutilización de plantillas.

componente `FOAF:Person`. Las macros están enmarcadas dentro de un rectángulo grueso, y las plantillas reutilizadas se han enmarcado con un rectángulo fino. La figura 5.5 muestra a esta plantilla en acción, mostrando una fuente de datos concreta. Se resaltan los ID de cada instancia `FOAF:Person` para mostrar que puede tratar tanto con instancias públicas (identificables por una URI con una URL accesible) como anónimas (“Anonymous” en la figura).

Figura 5.6: La plantilla de la figura 5.5 en uso. Se resalta el ID de cada instancia de tipo `FOAF:Person` de una fuente de datos semánticos.

Comparativa de las macros frente a los lenguajes de script

Esta sección describe brevemente algunas aplicaciones web semánticas representativas capaces de manejar plantillas para presentar datos semánticos, tales como wikis semánticos, navegadores de la Web Semántica, y portales semánticos. Se ha seleccionado una aplicación o infraestructura representativa de cada grupo: Semantic Media Wiki como wiki semántico, Fresnel (usado por el navegador Longwell), y Rhizomer (usado por el portal semántico Rhizomic), respectivamente.

Semantic Media Wiki permite que los usuarios de este wiki creen plantillas por medio de una sintaxis *ad hoc* con funciones de parseo [5]. La parte izquierda de la tabla 5.3

[5]Se puede ver un ejemplo de código fuente de una plantilla y ejemplos de uso en `http://en.wikipedia.org/wiki/Template:Infobox_Settlement`

muestra el código fuente de la plantilla [6], y la parte derecha de la figura muestra la renderización de los dato semánticos por medio de esa plantilla. El creador de la plantilla debe conocer la sintaxis wiki, elementos básicos de programación, y los elementos básicos de los componentes de las ontologías.

Fresnel [Pietriga *et al.*, 2006] is una infraestructura para crear plantillas para datos semánticos, usada por un navegador facetado de la Web Semántica llamado Longwell. Sin embargo, como muestra la tabla 5.4, la sintaxis de Fresnel [7] requiere conocimientos de las tecnologías de la Web Semántica que limitan severamente el número de diseñadores disponibles.

Rhizomer [García y Gil, 2006] es una infraestructura para navegar y editar datos semánticos. La presentación de los datos RDF se consigue mediante hojas de estilo XSLT (Extensible Stylesheet Language Transformations). Como se puede ver en la tabla 5.5, los diseñadores web requieren que usen Rhizomer requieren conocimientos avanzados de XSLT.

La tabla 5.2 resume las características de cada tipo de plantillas y las competencias requeridas a los creadores de plantillas para las aplicaciones semánticas consideradas.

[6] Edítese la página `http://semanticweb.org/wiki/Template:Person` para ver el código fuente de la plantilla entre `<includeonly>` y `</includeonly>`. Nota: el código previo a este bloque muestra un ejemplo de uso.

[7] Véase `http://www.w3.org/2005/04/fresnel-info/manual/`

	SMW	**Fresnel**	**Rhizomik**	**VPOET**
Lenguaje de programación	Sintaxis wiki, Programación, HTML/CSS	OWL, Fresnel ont., CSS	XSLT, OWL/XML	HTML, CSS, Javascript, macros
Permite				
Reutilización de plantillas	No	Sí	Sí	Sí
Renderización condicional	Sí	Sí	Sí	Sí
Imágenes	No	No	Sí	Sí
Dinamismo (Web 2.0)	No	No	Sí	Sí
Orientado a	Programadores promedio	Desarrolladores de la Web Semántica + CSS	Desarrolladores de la Web Semántica + XSLT	Diseñadores web + Macros
Competencias				
Tecs. Sem. Web	Bajo	Alto	Muy alto	Bajo
Tecs. Cliente Web	Bajo	Alto	Muy alto	Bajo-Muy alto
Otras	Sintaxis wiki	Alto	XSLT	
Pros	Requisitos medios	OWL puro	Solución genérica	Bajos requisitos
Contras	Sintaxis propensa a errores	Muy complejo para un diseñador web	Muy complejo para un diseñador web	

Tabla 5.2: Comparativa de las características de las macros de VPOET frente a otros entornos para la creación de plantillas

```
{| cellspacing="0" cellpadding="5" style="position:relative; margin: 0 0 0.5em 1em;
   border-collapse: collapse; border: 1px solid #aaa; background: #fff; float: right;
   clear: right; width: 20em"
! colspan="2" style="background: #86ba0c; color: white" |<span style="font-size: 80%;
                    float: right; ">{{#ask: [[{{FULLPAGENAME}}]]
| format=vcard
| ?Name
| ?Affiliation=organization
| ?Email
| ?Foaf:phone=workphone
| ?Homepage
| searchlabel=vCard
}}</span> [[Name::{{{Name|{{PAGENAME}}}}}]]
|-
{{#ifeq:{{{Picture|}}}|||{{Tablelongrow|Value=[[Image:{{{Pic
       }}}|150px|{{{Name|{{PAGENAME}}}}}]]|Color=white}}}}
|-
{{#ifeq:{{{Email|}}}|||{{Tablelongrow|Value={{Mailbox|{{{Ema
       }}}}}|Color=#e4f8b6}}}}
|-
{{#ifeq:{{{Affiliation|}}}|||{{Tablerow|Label=Affiliation:|
   Value=[[member of::affiliation::{{{Affiliation}}}]]}}}}
|-
{{#ifeq:{{{Homepage|}}}|||{{Tablerow|Label=Homepage:|
   Value=[[homepage::http://{{{Homepage}}}|{{{Homepage label
   {{{Homepage}}}}}]]}}}}
|-
{{#ifeq:{{{Phone|}}}|||{{Tablerow|Label=Phone:
   |Value=[[foaf:phone::{{{Phone}}}]]}}}}
|-
<!--  *** Events ***  -->
{{Tablelongrow|Align=Left|Font size=80%|
  Value={{#ask:[[has PC member::{{PAGENAME}}]]
| format=list
| sort=start date | order=desc
| sep=,_
| intro=PC member of:_ }}|Color=#e4f8b6}}
|-
{{Tablelongrow|Align=Left|Font size=80%|
  Value={{#ask:[[has OC member::{{PAGENAME}}]]
| format=list
| sort=start date | order=desc
| sep=,_
| intro=OC member of:_ }}|Color=#e4f8b6}}
|-
{{#ifeq:{{{FOAF|}}}|||{{Tablerow|Label=See also:|
Value=[[see also::{{{FOAF}}}|FOAF]]}}}}|Color=white}}
|}[[Category:Person]]
```

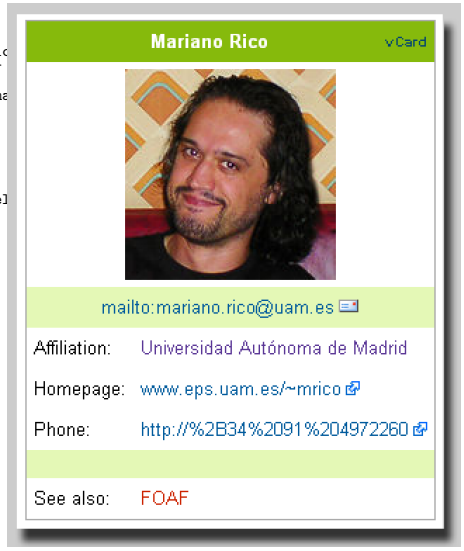

Tabla 5.3: Plantilla de Semantic Media Wiki. Izquierda: Código de la plantilla. Derecha: Ejemplo de renderización.

```
:foafGroup rdf:type fresnel:Group ;
     fresnel:stylesheetLink <http://www.example.org/example.css> ;
     fresnel:containerStyle "background-color: white;"
                            ^^fresnel:stylingInstructions ;

:foafPersonFormat rdf:type fresnel:Format ;
     fresnel:classFormatDomain foaf:Person ;
     fresnel:resourceStyle "background-color: gray;"
                           ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .

:nameFormat rdf:type fresnel:Format ;
     fresnel:propertyFormatDomain foaf:name ;
     fresnel:propertyStyle "border-top: solid black;"
                           ^^fresnel:stylingInstructions ;
     fresnel:labelStyle "font-weight: bold;"
                        ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .

:urlFormat rdf:type fresnel:Format ;
     fresnel:propertyFormatDomain foaf:homepage ;
     fresnel:propertyFormatDomain foaf:mbox ;
     fresnel:value fresnel:externalLink ;
     fresnel:propertyStyle "border-top: solid black;"
                           ^^fresnel:stylingInstructions ;
     fresnel:labelStyle "font-weight: bold;"
                        ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .

:depictFormat rdf:type fresnel:Format ;
     fresnel:propertyFormatDomain foaf:depiction ;
     fresnel:label fresnel:none ;
     fresnel:value fresnel:image ;
     fresnel:propertyStyle "border-top: solid black;"
                           ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .
```

Tabla 5.4: Plantilla de Fresnel (usado por el navegador Longwell). Izquierda: código de la plantilla. Derecha: ejemplo de renderización.

```
<?xml version="1.0"?>
<?xml-stylesheet href="./xsltdoc.xsl" type="text/xsl" media="screen"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <xsl:import href="rdf2html-functions.xsl"/>
  <xsl:param name="language">en</xsl:param>
  <xsl:output media-type="text/xhtml" version="1.0" encoding="UTF-8"
              indent="yes"/>
  <xsl:strip-space elements="*"/>
  <xsl:template match="/">
    <xsl:apply-templates select="rdf:RDF"/>
  </xsl:template>
  <xsl:template match="rdf:RDF">
    <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8"/>
      <title>Rhizomik - ReDeFer - RDF2HTML</title>
      <link href="http://rhizomik.net/style/rhizomik.css" ty
            rel="stylesheet" />
      <link href="http://rhizomik.net/style/rhizomer.css" ty
            rel="stylesheet" />
    </head>
    <body>
      <!-- div id="browser">
        <a href="javascript:history.back()">back</a> - go to
        <a href="javascript:history.forward()">forward</a>
      </div -->
      <xsl:if test="count(child::*)=0">
        <p>No data retrieved.</p>
      </xsl:if>
      <xsl:for-each select="child::*">
        <xsl:sort select="@rdf:about" order="ascending"/>
        <xsl:call-template name="rdfDescription"/>
      </xsl:for-each>
... 224 lines more ...
</xsl:stylesheet>
```

me a Person

depiction	http://www.geocities.com/marianorico/mi_careto.gif
family_name	Rico
givenname	Mariano
homepage	http://www.ii.uam.es/~mrico/
knows	a Person: mbox mailto:andreas.harth@deri.org; name Andreas Harth; seeAlso http://sw.deri.ie/~aharth/foaf.rdf
knows	a Person: fistName O'Murchu; mbox_sha1sum 712b83520bf846cdeb5b8158d541108b12a01447; name Ina O'Murchu
knows	http://www.cs.man.ac.uk/~ocorcho/foaf.rdf
knows	Uldis_Bojars
knows	me
mbox	mailto:Mariano.Rico@uam.es
name	Mariano Rico Almodóvar
workInfoHomepage	http://www.ii.uam.es/~mrico/
workplaceHomepage	http://www.ii.uam.es/~mrico/

Tabla 5.5: Plantilla en Rhizomic (usado por el portal semántico Rhizomik). Izquierda: template code. Derecha: ejemplo de renderización.

5.3.2. Creación de plantillas de entrada

La plantillas de entrada de VPOET son formularios HTML que convierten los valores introducidos en los elementos del formulario (campos de texto, botones de selección, etc.) en datos semánticos RDF. Como las plantillas de salida, las plantillas de entrada se asocian a componentes de ontologías (que normalmente se componen de otros sub-componentes). Por ejemplo, una plantilla de entrada para el componente `FOAF:Person` debe proporcionar un formulario con campos de texto para `firstName`, `knows`, etc. El proceso de creación de una plantilla de entrada es similar al de las plantillas de salida. Las principales diferencias son las siguientes:

- El código fuente de las plantillas de entrada no contiene macros, pero debe seguir las siguientes convenciones: (1) el código HTML debe contener un formulario HTML con el atributo `action` apuntando a un servlet específico proporcionado por VPOET, (2) el formulario debe contener un campo oculto llamado "ontoelem", cuyo valor es el nombre del componente de la ontología para el que se crea esta plantilla (en sintaxis OMEMO), (3) los controles del formulario (campos de texto, selectores, etc.) deben tener su atributo *nombre* con el valor del componente de la ontología con

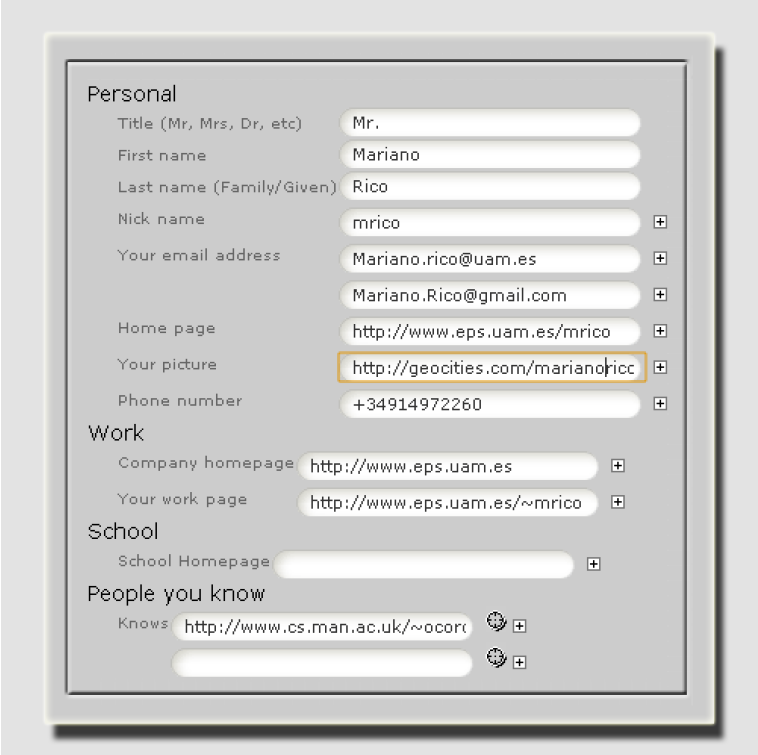

Figura 5.7: Ejemplo de plantilla de entrada para la clase `FOAF.20050403.Person`.

que se relaciona, en notación OMEMO.

- La fase de pruebas no utiliza fuentes de datos semánticos. Sin embargo, como el servlet de VPOET genera datos semánticos en cierta URL, esta fuente puede ser mostrada por un template de salida. Por tanto, se recomienda crear una plantilla de salida por cada plantilla de entrada para poder realizar las pruebas. Los errores más típicos, como nombres erróneos de propiedades, pueden ser detectados con esta estrategia.

La figura 5.7 muestra un ejemplo de plantilla de entrada para la clase `FOAF.20050403.Person` (en notación OMEMO). La figura muestra un icono "+" en la parte derecha de algunos campos. Esto indica que la propiedad es multivaluada y, por tanto, puede tener varios valores asignados. El icono "diana" mostrado a la derecha del campo asignado a la relación `knows` abre un "navegador de instancias", una ventana que permite a los usuarios seleccionar un conjunto de instancias de la clase destino correspondiente. Si no se desea usar el "navegador de instancias" se puede introducir directamente la URI en el campo de texto.

Tabla 5.6: Plantilla simple de ejemplo.Izda.: Código HTML inicial. Dcha.: código procesado por un navegador web

Código	Visualización
<table style='background: #F0F0F0; border ;padding: 1px; border: thin solid #DDDDDD; margin-left: 1em'> <tr> <td>Name:</td><td>this is the name</td> </tr> <tr> <td>Given Name:</td><td>this is the givenname</td> </tr> <tr> <td>Family name:</td><td>this is the family_name</td> </tr> <tr> <td>Home page:</td><td>http://somewhere.com</td> </tr> <tr> <td>Depiction:</td><td><img src='http://somewhere.com/mi_image.gif'></td> </tr> <tr> <td>Knows:</td><td>list of known people</td> </tr> </table>	Name: this is the name Given Name: this is the givenname Family name: this is the family_name Home page: http://somewhere.com Depiction:  Knows: list of known people

5.3.3. Caso de uso de VPOET

Esta subsección muestra en detalle un ejemplo de uso de VPOET. Se considera el caso de un proveedor de plantillas al que se le ha encargado crear una plantilla de salida para el componente `Person` de la ontología `FOAF` (versión 20050403).

El primer paso consiste en conseguir información acerca de los componentes de `Person`. Para ello, el proveedor de plantillas dirige su navegador web hacia la aplicación web OMEMO, y busca ese componente utilizando las facilidades proporcionadas por el buscador de texto de OMEMO. Como se describió en la sección anterior, el proveedor de plantillas observa que hay varias versiones de `Person`. Comprueba que la que debe utilizar es la versión 20050403 y navega hasta la página mostrada en la figura 5.2. Empieza el diseño de la plantilla considerando tan sólo unos pocos componentes, tales como *name*, *family_name*, *homepage*, *depiction*, y *knows*. Observa que *name* y *family_name* son literales, que *homepage* es una URL que apunta a una imagen, y que *knows* es una relación que apunta a `Person`.

Con toda esta información, el proveedor de plantillas utiliza su herramienta de autor preferida (por ejemplo, Dreamweaver) para crear un fichero HTML. La tabla 5.6 muestra el código HTML y su visualización en un navegador web estándar.

El siguiente paso es cargar este diseño en VPOET. Dirige su navegador web a VPOET

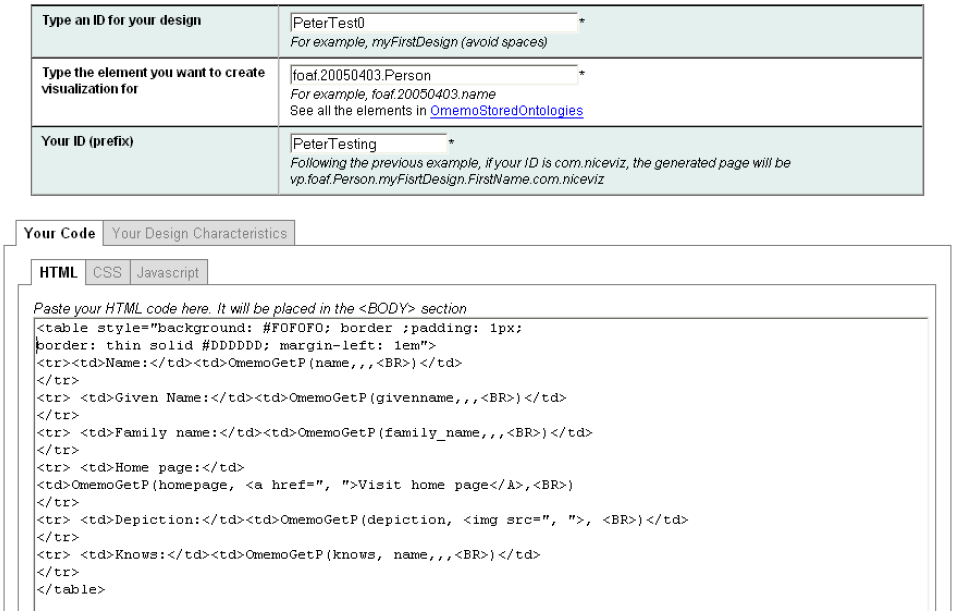

Figura 5.8: Creación de una plantilla en VPOET

y pincha en el enlace “Add a new design”. La figura 5.8 muestra la página wiki de VPOET que permite crear plantillas. Debe copiar y pegar el código al área de texto HTML tal y como muestra la figura. Una vez hecho ya puede pulsar el botón “Submit” para que la plantilla quede almacenada.

Aunque la plantilla se encuentre almacenada, aún no está lista para mostrar una fuente de datos semánticos. El proveedor de plantillas debe sustituir algunas partes del código de su diseño inicial por macros. En la figura 5.8 se puede ver el código mostrado en la parte izquierda de la tabla 5.6 tras la inserción de las macros. En este ejemplo tan sencillo sólo se usa una de las macros del sistema (`OmemoGetP`).

VPOET proporciona al proveedor de plantillas facilidades para probar la plantilla creada, mostrando cómo será presentada una fuente de datos semánticos dada a través de la plantilla. Una página wiki de VPOET muestra las diversas fuentes de datos semánticos disponibles, con comentarios acerca de las características que las hacen particularmente interesantes como casos de prueba. Por ejemplo, propiedades multivaluadas, o enlaces a imágenes inexistentes. Para este caso de prueba se ha usado el fichero FOAF del W3C de Tim Berners-Lee, que se encuentra en `http://www.w3.org/People/Berners-Lee/card`. El resultado de usar esta fuente de datos semánticos con la plantilla anterior se muestra en

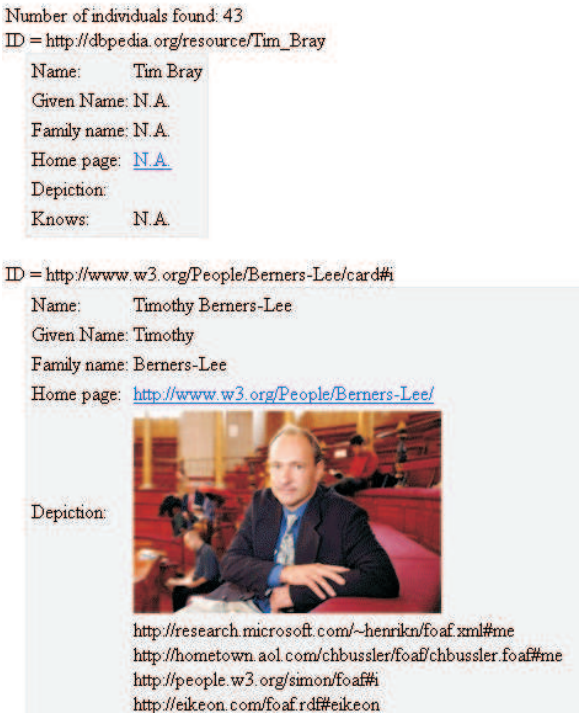

Figura 5.9: Fuente de datos semánticos renderizada usando la plantilla simple de ejemplo

la figura 5.9. Observando la imagen de la figura, el proveedor de templates descubre que en esa fuente de datos hay 43 instancias de `Person`, cada una con su ID. Este ID puede corresponder a una instancia dentro de otra fuente de datos, por lo que se puede recorrer un árbol de fuentes de datos relacionadas, al estilo de los navegadores web semánticos. Puede repetir este test especificando una instancia concreta de la fuente de datos. En este caso, sólo se muestra esa instancia.

Una vez que el proveedor de plantillas da por terminado el ciclo de pruebas, deberá caracterizar la plantilla, proporcionando información acerca de si es una plantilla de salida o de entrada, así como (1) datos técnicos (tamaño preferido, tamaño mínimo, tamaño máximo, o si soporta cambios en el tamaño de la fuente sin penalizaciones en el diseño), (2) colores (dominantes y secundario para frente y fondo), y (3) estética (dominante y secundaria, elegidas de una lista en la que se encuentran conceptos tales como *minimalista*, *recargada*, o *siniestra*, entre otros. Esta información es almacenada en el sistema y publicada automáticamente gracias a las facilidades proporcionadas por Fortunata.

La plantilla que se ha creado en este ejemplo debe ser caracterizada como plantilla de

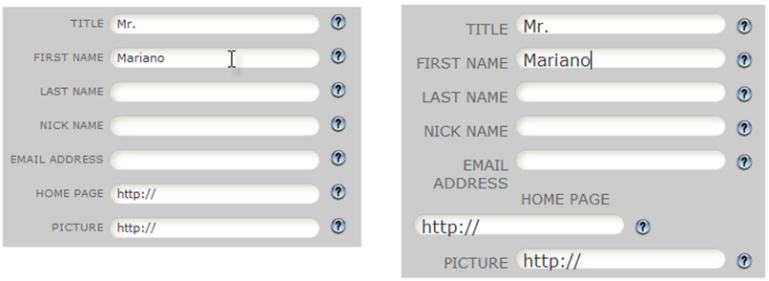

Figura 5.10: Ejemplo de plantilla de entrada para FOAF:Person. Izda.: Tamaño de fuente normal. Dcha.: esta plantilla no escala bien si se incrementa el tamaño de la fuente

salida, con tamaño variable (adaptado a su contenido). Un ejemplo de plantilla de entrada se muestra en la parte izquierda de la figura 5.10. En la parte derecha se puede ver el efecto de aumentar el tamaño de la fuente, una capacidad implementada en la mayor parte de los navegadores web. El resultado es una colocación defectuosa de los últimos campos de texto, por lo que esta plantilla debe caracterizarse como "no soporta aumentos del tamaño de la fuente".

5.4. VPOET para desarrolladores de aplicaciones web

Siguiendo la estrategia mostrada en la figura 5.1, aunque la información almacenada en VPOET se publica como datos semánticos accesibles a través de una URL que puede ser usada por sistemas externos o especialistas de la Web Semántica, se ha creado un canal adicional para permitir a los desarrolladores de aplicaciones web acceder a esta información sin necesidad de tener conocimientos de las tecnologías de la Web Semántica. Este canal se ha implementado como un servlet que realiza consultas sobre los datos semánticos almacenados por VPOET, al que se puede acceder mediante mensajes HTTP GET/POST con un número variable de parámetros. Estos mensaje permiten consultas como "muestra los datos semánticos de la URL Y usando la plantilla de salida X y el proveedor Y". Esta consulta se codifica en HTTP GET mediante la URL `http://URL-to-servlet/VPoetRequestServlet?action=renderOutput&designID=X&provider=Y&source=Z`. La sintaxis completa se muestra en la tabla 5.7.

Aunque los SPARQL Endpoints [8] proporcionan acceso a funetes de datos semánticos

[8] Véase `http://www.w3.org/TR/rdf-sparql-protocol/`

Tabla 5.7: Parámetros aceptados en una petición HTTP GET/POST para obtener una visualización almacenada en VPOET.

Parámetro	**Valor**	**Explicación/Ejemplo**
action	`renderOutput`	Pedir una visualización para mostrar unos datos RDF dados del componente indicado por el param `object`
	`renderInput`	Pedir una visualización para solicitar datos al usuario del componente indicado por el param `object`
object	prefix.class[.ver] prefix.relation[.ver]	Ejemplo: foaf.Person Ejemplo: foaf.firstName
`source` (sólo GET)	URL	URL de la fuente de datos semánticos
`provider` (opcional)	ID	Identificador del proveedor de plantillas. Por ejemplo: user3.test
outputFormat	`HTML` `XHTML`	Valor por omisión XHTML es usado por móviles WAP 2.0
`userProfile` (opcional y sólo GET	URL	URL del fichero RDF con la descripción del perfil de usuario

mediante mensajes sobre HTTP de forma similar, están orientados a usuarios con conocimientos de las tecnologías de la Web Semántica ya que la consulta del SPARQL Endpoint se especifica en lenguaje SPARQL. En la estrategia seguida, ni los usuarios finales ni los desarrolladores de aplicaciones web tienen conocimientos de la tecnologías de la Web Semántica, por lo que el uso de consultas SPARQL se vuelve inviable.

Cuando se usa el método GET, el parámetro `source` debe se proporcionado para indicar dónde se encuentran los datos semánticos. Por el contrario, cuando se usa el método POST, el parámetro `source` no es necesario ya que los datos semánticos deber ir contenidos en el mensaje. Si el parámetro `provider` no se proporciona, VPOET devolverá la "mejor visualización" para el perfil de usuario apuntado por el parámetro `userProfile`. El método POST está dirigido a las consultas en las que los datos RDF se añaden al mensaje POST.

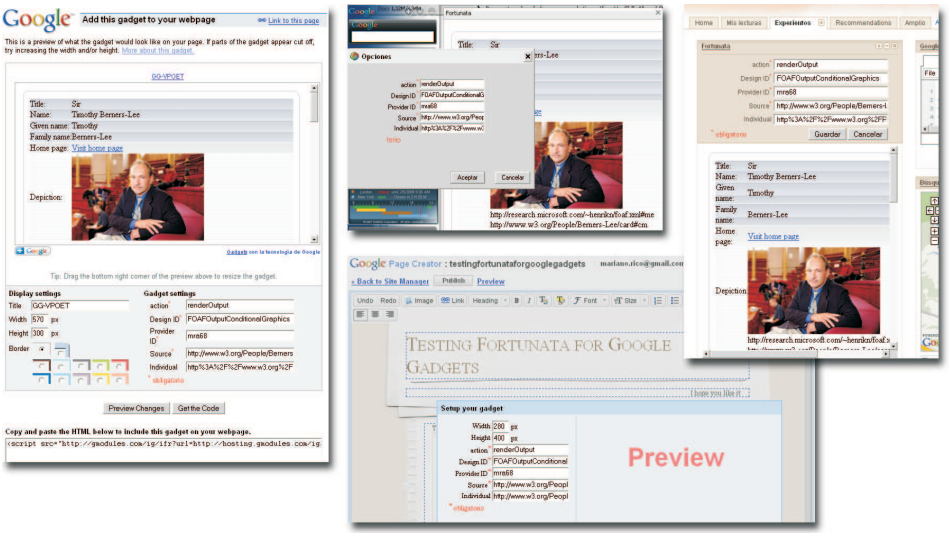

Figura 5.11: Uso de GG-VPOET en diferentes aplicaciones orientadas a usuarios finales. En sentido horario: página personal, Google Desktop, iGoogle, y Google Pages

5.4.1. Google Gadget VPOET (GG-VPOET)

Los mensajes sobre HTTP, con la sintaxis adecuada, pueden ser enviados a VPOET por otros programas escritos en cualquier lenguaje de programación, o por aplicaciones Javascript ejecutadas en un navegador web. Sin embargo, los navegadores se encuentran mucho más limitados que otras aplicaciones ya que sufren de restricciones de seguridad que les fuerzan a comunicarse únicamente con el servidor con el que mantienen la sesión. Por ejemplo, la página web que contiene Tabulator, el navegador para la Web Semántica creado por Tim Berners-Lee, se aloja en `http://dig.csail.mit.edu/2005/ajar/ajaw/tab`, y sólo se pueden solicitar datos a ese servidor. Es necesario configurar el navegador (sólo disponible para Firefox) para poder obtener datos de otros servidores. Obviamente este cambio de configuración es obligatorio, ya que Tabulator puede necesitar acceder a datos semánticos esparcidor por toda la Internet. Por contra, la aproximación propuesta no sufre este problema ya que las peticiones las centraliza Fortunata, y no el navegador directamente.

Para facilitar aún más el uso de las plantillas de VPOET y, como mostraba la figura 5.1, permitir que incluso los usuarios finales puedan usar estas plantillas, se ha creado un

Google Gadget llamado GG-VPOET [9]que se encarga de construir los mensajes HTTP adecuados. Un usuario final puede usar este gadget para generar datos semánticos (mediante plantillas VPOET de entrada) o mostrar datos semánticos (mediante plantillas de salida) en sus páginas web, o en diversos productor Google como iGoogle, Google Pages, or Google Desktop. Este gadget lo pueden configurar los usuarios finales fácilmente, proporcionando la misma información que se usaba durante la fase de test: designerID, designID, URL de la fuente de datos y, opcionalmente, el ID de una instancia concreta. La figura 5.11 muestra cómo se configura este gadget para su uso en diversos entornos, desde páginas personales hasta productos Google.

5.4.2. Mapeado de perfiles de usuario con plantillas semánticas de VPOET

Supongamos que VPOET contiene diferentes plantillas para el componente semántico `foaf.Person`, y que una aplicación externa solicita a través del canal HTTP la "más adecuada" para cierto perfil de usuario dado. En la figura 5.12 se muestra un posible proceso de mapeado. Cada ontología, identificada por un namespace, se muestra en una nube; con los componentes de la ontología y algunos individuos dentro de la nube. La parte izquierda de esta figura 5.12 muestra la ontología que describe el perfil del usuario, caracterizada por el namespace *a*. En este ejemplo, el usuario identificado por `user34` tiene el siguiente perfil: (1) usa como dispositivo de interacción un teléfono móvil que usa el protocolo WAP2, (2) prefiere estéticas sencillas, y (3) es daltónico (ceguera de color asociada a los colores rojo y verde).

La parte central de la figura 5.12 muestra algunas ontologías públicas conocidas. La ontología z_1 indica que el protocolo WAP2.0 se codifica en XHTML. Según la ontología z_3, "minimalista" y "simple" son estilos distintos pero cercanos. La ontología z_5 es una jerarquía de deficiencias visuales.

La parte derecha de la figura 5.12 muestra la ontología VPOET, con namespace *v*. En esta ontología, la plantilla identificada como `design67` está codificada en XHTML, su estética primaria es minimalista, y tiene por colores principal y secundario al rojo y al amarillo respectivamente. Usando sólo esta información, es imposible encontrar que `design67` es "la plantilla más adecuada" para el usuario `user34`. Se requiere una fuente

[9]Véase `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=VPOETGoogleGadget`

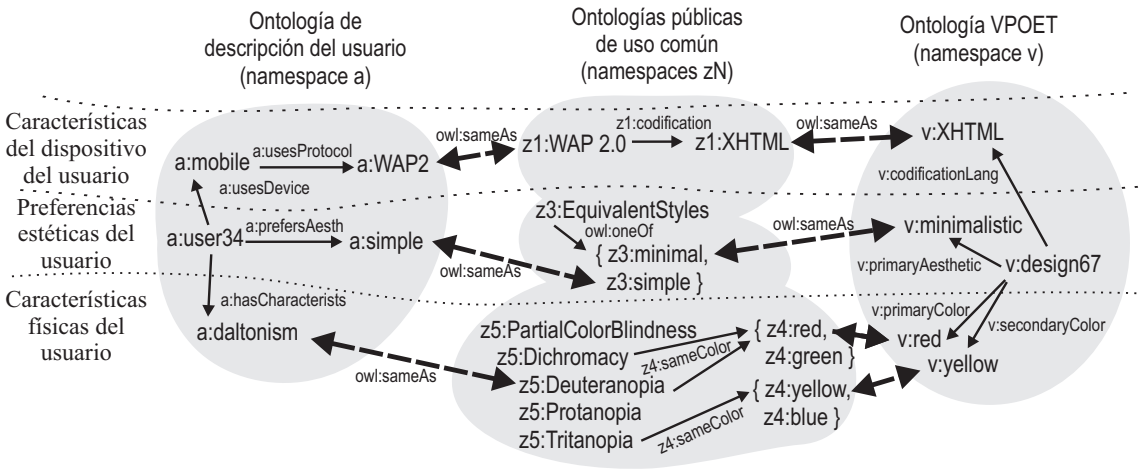

Figura 5.12: De izquierda a derecha: ontología de perfil de usuario, ontologías públicas, y la ontología de VPOET

.

adicional de datos semánticos que ligue componentes de las distintas ontologías. Estos enlaces suelen ser relaciones "sameAs"[10], mostradas como flechas discontinuas gruesas en la figura 5.12.

Añadiendo esta información semántica, un agente semántico podrá hacer una consulta semántica (por ejemplo, usando SPARQL), basada en el perfil de usuario, como esta: "seleccionar una plantilla con esta características: (1) codificada en XHTML, (2) con estética minimalista, y (3) con colores primario y secundario que no sean ni rojo ni verde tanto para el texto como para el fondo". Para este ejemplo, el resultado de la consulta sería el diseño `design67`. Añadiendo más restricciones a la consulta se conseguirá la plantilla "más adecuada" para un perfil de usuario dado.

5.4.3. Me InteractinG (MIG)

MIG [11] es una aplicación basada en Fortunata para la creación de perfiles de usuario dirigida a usuarios finales. Este perfil comprende detalles acerca de deficiencias visuales (como muestra la figura 5.13), el dispositivo utilizado por el usuario, y sus preferencias estéticas. Los datos introducidos por los usuarios son convertidos a datos semánticos y

[10]Técnicamente hay que distinguir entre owl:sameAs, owl:equivalentClass y owl:equivalentProperty para indicar igualdad entre individuos, clases, y propiedades/relaciones respectivamente.

[11]Véase `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=MIG`

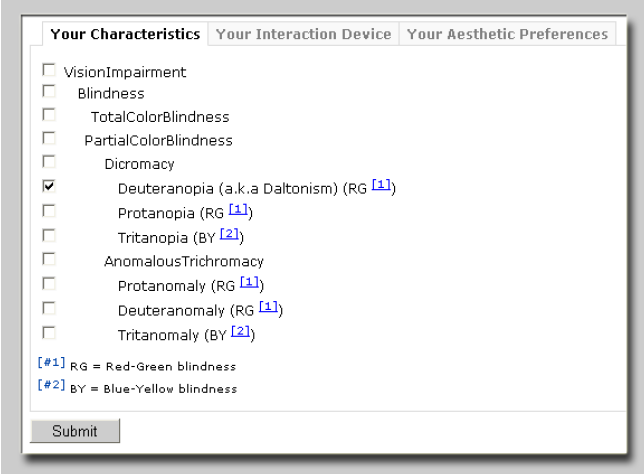

Figura 5.13: Características del usuario en MIG

publicados, como en cualquier aplicación basada en Fortunata.

Para lograr una descripción del perfil de un usuario se han probado algunas especificaciones que incluyen las deficiencias visuales de los usuarios, los distintos tipos de dispositivo de interacción que puede utilizar (PC, móvil, PDA...), así como sus preferencias estéticas. Para evitar crear una ontología *ad hoc*, se ha considerado reutilizar ontologías públicas bien conocidas. Concretamente, el dispositivo del usuario puede ser especificado usando la ontología CCPP [Klyne *et al.*, 2004] (en realidad una extensión de CCPP llamada UAProf[12]), las deficiencias visuales descritas por [Karim y Tjoa, 2006], y las características interactivas pueden ser modeladas con versiones ontológicas de MPEG-7 y MPEG-21 (véase trabajo de [Jabornig, 2006]). En MIG, las preferencias del usuario se han modelado como una taxonomía arbitraria con conceptos como *simple*, *barroco*, o *minimalista*. El perfil del usuario proporciona una lista ordenada de estéticas preferidas.

En MIG, la caracterización del dispositivo del usuario comprende detalles como el tamaño de pantalla, bits de color por pixel, y modelo de navegador. La mayor parte de los navegadores envían la cadena de "User-Agent" (UA) al servidor en cada mensaje HTTP. MIG explota esta característica para detectar el modelo de navegador, el sistema operativo, y el modelo de dispositivo cuando el usuario utiliza el dispositivo. MIG compara la cadena

[12] Véase `http://cms.openmobilealliance.org/technical/release_program/docs/copyrightclick.aspx?pck=UAProf&file=V2_0_1-20070625-A/OMA-TS-UAProf-V2_0-20060206-A.pdf`

UA enviada por el navegador del dispositivo del usuario con WURFL [13], un repositorio de UAs. Este repositorio almacena información de unos 9.000 dispositivos, con cientos de posibles capacidades (e.g. `is_wireless_device` o `resolution_width`). Si se encuentra la UA del usuario en el repositorio, la mayoría de los datos relativos a las capacidades del dispositivo son obtenidas de este repositorio.

Buscando la mejor plantilla

Como se indicó anteriormente, el modelo obtenido de la mezcla de la información semántica de VPOET, MIG, las ontologías comunes, y los elementos de unión, puede ser consultado por medio de lenguajes de consultas semánticas como SPARQL. Aunque los resultados de una consulta dada dependen de la información almacenada en el modelo, como vimos en el ejemplo, y la misma consulta puede devolver 0 o muchos resultados dependiendo de la existencia de elementos clave de unión, ¿qué sucede cuando muchos resultados cumplen la consulta?. La única manera que se dispone de restringir los resultados para alcanzar "la mejor" plantilla para un perfil de usuario dado es añadir más parámetros a la consulta SPARQL.

El problema de usar una única consulta SPARQL es que puede devolver muchos resultados, sin ningún criterio para su ordenación, o no devolver ninguno en absoluto. El primer caso denota una consulta demasiado relajada, y el segundo caso una consulta demasiado restrictiva.

La solución adoptada considera un conjunto de consultas SPARQL, ordenadas de menos restrictivas a más restrictivas, esto es, de pocos parámetros a muchos parámetros, conforme a un criterio de importancia. Por ejemplo, la primera consulta puede solicitar concordancia en el tamaño de la pantalla; la segunda, concordancia en el tamaño de la pantalla y el modelo de navegador, etc. Cuando la primera consulta se ejecuta, si devuelve más de un resultado se dispara la segunda consulta, reiterando este proceso hasta que no se devuelven resultados. La última consulta que devolvió resultados es considerada "la mejor", y cualquiera de las plantillas devueltas como resultado es utilizada para mostrar la fuente de datos dada.

[13] Véase `http://developer.openwave.com/dvl/tools_and_sdk/wurfl_and_wall`

5.5. Conclusiones

En este capítulo se han mostrado las aplicaciones que permiten que los desarrolladores de aplicaciones puedan incorporar de manera sencilla, e intuitiva, datos semánticos a sus aplicaciones. Utilizando la capacidad creativa de los diseñadores web, VPOET permite poner a disposición de terceras partes un repositorio semántico de plantillas para manejo de datos semánticos. Este repositorio es accesible mediante mensajes HTTP para permitir que los desarrolladores de aplicaciones web puedan acceder a estos datos independientemente del lenguaje en el que se implemente la aplicación web.

La evaluación experimental de VPOET y OMEMO confirma (véase sección 6.2) que la plataforma Fortunata no impone restricciones demasiado severas a las aplicaciones en términos de usabilidad o satisfacción del usuario con la interfaz. Sin embargo, no se ha realizado ningún experimento para evaluar la facilidad de uso del canal HTTP. Tampoco se ha evaluado la facilidad de uso del Google Gadget creado para que incluso los usuarios finales puedan incorporar datos semánticos a sus páginas web se manera sencilla, pero parece razonable suponer que la estrategia seguida consigue poner la Web Semántica al alcance de usuarios finales y desarrolladores de aplicaciones web.

La experimentación con perfiles de usuario abre una prometedora línea de investigación que podría ser explotada por los agentes semánticos del futuro próximo. También ha quedado abierta la especificación de una ontología que permita caracterizar a las plantillas de VPOET. Información acerca de navegadores soportados o parámetros técnicos que permitan identificar los dispositivos en los que pueda ser usada la plantilla permitirán ampliar la utilidad de las plantillas VPOET.

Capítulo 6

Evaluación experimental

Este capítulo muestra los experimentos llevados a cabo para evaluar las propuestas descritas en el capítulo 3. La evaluación de Fortunata se llevó a cabo por especialistas en usabilidad y por desarrolladores de aplicaciones. La evaluación por parte de los especialistas en usabilidad permitió corregir deficiencias en tiempo de diseño, previo a la creación de las herramientas OMEMO y VPOET, y tuvo como resultado una guía de recomendaciones para desarrolladores de aplicaciones basadas en Fortunata. La evaluación con desarrolladores permitió evaluar la capacidad colaborativa de Fortunata.

La evaluación de las hipótesis relacionadas con OMEMO y VPOET se llevó a cabo con usuarios que ejercían el rol de diseñadores gráficos. El objetivo de este estudio fue evaluar la usabilidad de las herramientas OMEMO y VPOET, así como medir el grado de satisfacción de los usuarios con respecto al interfaz de usuario. El motivo para realizar este experimento es el siguiente: aunque se han tratado de mostrar las ventajas de crear aplicaciones web semánticas basadas en Fortunata, dichas aplicaciones se encuentran limitadas por las capacidades interactivas provistas por el wiki-engine (JSPWiki) sobre el que se apoya Fortunata. Por tanto, hay que comprobar que las ventajas que obtienen los desarrolladores no suponen una penalización excesiva en la usabilidad o satisfacción de los usuarios. Evidentemente, una aplicación ad hoc *siempre puede ofrecer más y mejores elementos interactivos, pero el coste de desarrollo será mucho mayor. Las siguientes secciones describen en detalle el procedimiento experimental, mostrando el cuestionario de evaluación utilizado, los análisis realizados, así como los resultados más relevantes encontrados.*

6.1. Evaluación de la plataforma Fortunata

La usabilidad puede ser definida como "la facilidad de utilización y aceptación de un sistema para un tipo concreto de usuarios que llevan a cabo una tarea específica en un entorno específico" [Holzinger, 2005]. En el contexto de las aplicaciones web semánticas, la usabilidad es todavía un reto importante [Heath *et al.*, 2006].

Por tanto, en este trabajo se ha intentado medir y conocer cuán utilizables son las aplicaciones basadas en Fortunata. A este fin, se ha analizado la usabilidad de OMEMO y VPOET, como ejemplos del tipo de aplicaciones que pueden ser creadas con la plataforma Fortunata.

En el área de la usabilidad se identifican dos grupos principales de técnicas de evaluación de la usabilidad [Holzinger, 2005]:

- Métodos de Test, aplicados normalmente a aplicaciones que están funcionando (o al menos con prototipos iniciales de estas aplicaciones), que requieren usuarios reales de estas aplicaciones.

- Métodos de Inspección, normalmente usados durante la fase de diseño de las aplicaciones para identificar problemas potenciales de usabilidad, que no requieren usuarios reales sino evaluadores de usabilidad.

En esta evaluación se han aplicado métodos de inspección, ya que el objetivo ha sido identificar problemas potenciales de usabilidad de las aplicaciones durante su fase de diseño. Los Métodos de Inspección comprenden diferentes técnicas, como: Evaluación Heurística (HE), Repaso Cognitivo, y Análisis de Acción; que requieren evaluadores con diferentes niveles de competencias. El primero de ellos (HE) fue el seleccionado debido a que requiere evaluadores con menores competencias, lo que significa que los evaluadores no necesitan ser expertos en usabilidad, que son difíciles de conseguir. En realidad, un grupo de 3-5 evaluadores aplicando esta técnica identifican típicamente el 75-80 % de todos los problemas de usabilidad [Baker *et al.*, 2001].

Se preguntó a los evaluadores si la interfaz de usuario seguía unos principios de usabilidad bien conocidos, y los resultados de la evaluación se utilizaron para mejorar Fortunata y para crear una "guía de usabilidad" para los desarrolladores de aplicaciones basadas en Fortunata.

El segundo tipo de evaluación se centró en la medición de la capacidad de Fortunata en cuanto a colaboración contributiva. En este experimento se seleccionaron varios desarrolladores con competencias similares en tecnologías web cliente y con mínimos conocimientos de tecnologías de la Web Semántica, y fueron divididos en dos grupos identificados respectivamente como "A" y "B". Se propuso un objetivo común a ambos grupos, consistente en la creación de manera incremental de las siguientes tres aplicaciones web semánticas:

1. Una "Agenda personal", con una interfaz de usuario para proporcionar el nombre, dirección de correo electrónico, y teléfono.

2. Usando los datos del paso 1, crear la funcionalidad necesaria para proporcionar una interfaz de usuario para organizar reuniones de una persona dada.

3. Usando los datos del paso 2, añadir la funcionalidad necesaria para mostrar la gente implicada en una reunión en una fecha dada.

El primer grupo "A" no tenía ningún conocimiento de Fortunata, de forma que fueron libres de usar sus tecnologías y herramientas de desarrollo preferidas. La única restricción que debían seguir era la de cumplir las "ocho reglas de oro" de la usabilidad (descritas en la siguiente sección). El segundo grupo "B" recibió un breve entrenamiento (en la forma de un tutorial práctico de 20 minutos) acerca de la plataforma Fortunata, y se les pidió usar esta plataforma siguiendo la "guía de usabilidad" obtenida del primer experimento. El experimento finalizó rellenando un completo cuestionario con preguntas cuantitativas y cualitativas.

6.1.1. Preparación del experimento

Experimento de usabilidad

Se seleccionaron cinco evaluadores de usabilidad de la Universidad Autónoma de Madrid y se les pidió evaluar independientemente prototipos iniciales de OMEMOy VPOET. Durante las sesiones de evaluación cada evaluador usó las aplicaciones varias veces, comenzando desde distintos puntos, inspeccionando los elementos interactivos. Tuvieron que responder preguntas relacionadas como "las ocho reglas de oro" [Shneiderman y Plaisant, 2005], mostradas en la tabla 6.1.

El cuestionario se componía de diez preguntas de [Nielsen y Mack, 1994]. Es un cuestionario Likert 'basado en preguntas de elección forzada, en los que se presenta una frase y se debe responder indicando el grado de acuerdo o desacuerdo con la frase en una escala de 5 (o 7) puntos." [Brook, 1996], con siete posibles valores para las respuestas en el rango entre 1 (difícil) y 7 (sencillo). Además de las opciones, cada pregunta tenía un campo opcional de comentarios. La tabla 6.2 muestra las preguntas [1]. Adicionalmente, cada evaluador proporcionó una lista de recomendaciones para mejorar la usabilidad. Estos comentarios y recomendaciones se analizaron una vez que se llevaron a cabo las evaluaciones independientes.

[1] Disponible en `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=QuickUsabilityTestForEvaluators`

Tabla 6.1: Las "ocho reglas de oro" de la usabilidad.

ID	Regla	Descripción
1	Consistencia	Debe haber consistencia en las acciones, terminología (mensajes, menús, y ventanas de ayuda) y los gráficos (colores, distribución, y tipos de letra).
2	Usabilidad universal	Cada usuario tiene una necesidad, por tanto, el sistema debe proporcionar algún mecanismo para transformar los contenidos. No sólo usuarios discapacitados, sino diferencias entre principiantes-expertos (los principiantes necesitan explicaciones, los expertos necesitan atajos (shotcuts), o rangos de edad.
3	Reacción informativa	Cada acción del sistema debe producir una reacción. Para acciones comunes poco importantes, la reacción debe ser pequeña, pero las acciones infrecuentes o importantes deben producir una respuesta más grande.
4	Diálogos	Los diálogos deben ser diseñados para finalizar algo. Las secuencias de acciones se deben organizar en grupos, con un comienzo, desarrollo, y fin. Por ejemplo, el concepto de carrito en aplicaciones web, con visualización de etapas terminadas y pendientes.
5	Prevención de errores	El sistema debe evitar que los uaurios cometan errores, pero si se produce el error, el sistema debe proporcionar una solución simple, constructiva, y concreta.
6	Deshacer	Se debe permitir que los usuarios puedan deshacer acciones de forma sencilla. Todo debería poder deshacerse.
7	Control del locus interno	Soporte para el control interno del locus. Los usuarios expertos deben tener la sensación de controlar la herramienta. Los usuarios deben iniciar las acciones, no sólo responder a ellas.
8	Carga de memoria	Reducir la carga de memoria de corto plazo. Evitar múltiples ventanas, códigos, o secuencias complejas.

Tabla 6.2: Cuestionario para los evaluadores de la usabilidad de las aplicaciones basadas en Fortunata.

ID	Pregunta
1	*Visibilidad del estado del sistema.* El sistema debería mantener al usuario siempre informado acerca de lo que está sucediendo, mediante una respuesta adecuada en un tiempo razonable.
2	*Correspondencia entre el sistema y el mundo real.* El sistema debe hablar el lenguaje del usuario, con palabras, frases, y conceptos familiares al usuario, en vez de usar términos orientados al sistema. Debe seguir las convenciones del mundo real, haciendo que la información aparezca en un orden natural y lógico.
3	*Control del usuario y libertad.* A menudo, los usuarios eligen funciones del sistema por error, y deben tener claramente marcado una "salida de emergencia" para salir del estado no deseado sin tener que seguir un diálogo extenso. Debe permitir deshacer y rehacer.
4	*Consistencia y estándares.* Los usuarios no deberían tener que preguntarse si diferentes palabras, situaciones, o acciones, significan lo mismo. Siga las convenciones de la plataforma.
5	*Prevención de errores.* Es mucho mejor un diseño cuidadoso que prevenga que un problema suceda, que un buen mensaje de error. O bien se eliminan las condiciones propensas a errores, o bien se comprueban y se pide confirmación al usuario antes de ejecutar la acción.
6	*Reconocimiento mejor que recordar* Minimizar la carga de memorización del usuario haciendo visibles los objetos, las acciones, y las opciones. El usuario no debe tener que recordar información de un diálogo a otro. Las instrucciones de uso del sistema deben estar a la vista o ser fácilmente recuperables cuando sean necesarias.
7	*Flexibilidad y eficiencia de uso.* Los aceleradores (atajos) – invisibles para los usuarios noveles – suelen aumentar la velocidad de interacción de los usuarios expertos, de forma que el sistema puede atender tanto a los usuarios sin experiencia como a los expertos. Se debe permitir que los usuarios creen aceleradores para sus acciones más frecuentes.
8	*Estética y diseño minimalista.* Los diálogos no deberían contener información irrelevante o de escaso uso. Cada unidad adicional de información en un diálogo compite con el resto de unidades de información y disminuye su visibilidad relativa.
9	*Ayude a los usuarios a reconocer, diagnosticar, y recuperarse de los errores.* Los mensajes de error debe ser expresados en un lenguaje sencillo (sin códigos), indicar con precisión el problema, y sugerir una solución constructiva.
10	*Ayuda y documentación.* Aunque es mejor si el sistema puede ser usado sin documentación, puede ser necesario proporcionar ayuda y documentación. Toda esa información debe ser fácil de buscar, estar centrada en la tarea del usuario, mostrar una lista de pasos concretos que llevar a cabo, y no ser demasiado grande.

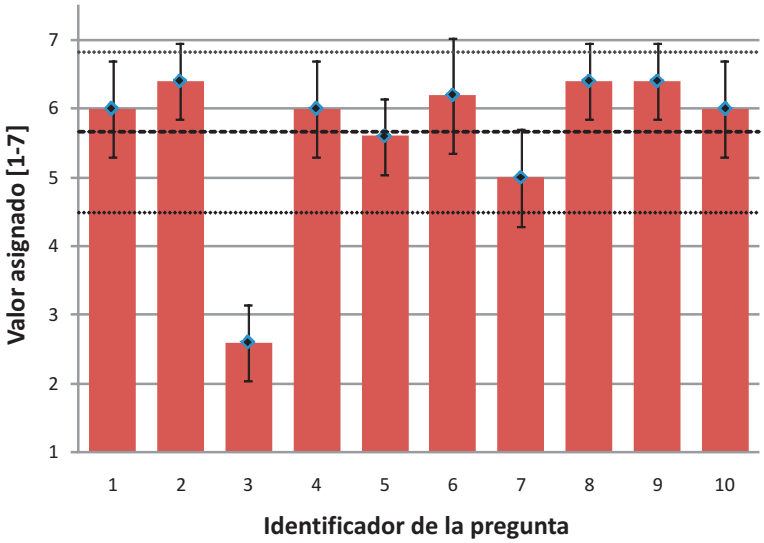

Figura 6.1: Resultados del cuestionario para desarrolladores de aplicaciones basadas en Fortunata.

Experimento de colaboración

Para el segundo experimento se seleccionaron seis estudiantes de un curso de master de "Tecnologías de la Web Semántica". Tres estudiantes fueron asignados al grupo "A" (desarrolladores tradicionales) y los otros tres fueron asignados al grupo "B" (desarrolladores de Fortunata). El cuestionario se comprendía se dos bloques principales de preguntas relativas a la complejidad, la colaboración, y la contribución. Algunas preguntas (Q3 a Q6) usaron el anterior rango de valores basado en escalas Likert, y el resto proporcionó un valor numérico continuo. Estas preguntas se muestran en la tabla 6.3.

6.1.2. Resultados experimentales

Experimento de usabilidad

Para el primer experimento propuesto, la figura 6.1 muestra los valores promedio asignados por los evaluadores a cada pregunta, así como sus desviaciones estándar. El valor medio de usabilidad (línea discontinua) fue 5.66 (con valores posibles en el rango [1, 7]), con una desviación estándar de 1.16 (líneas de puntos).

La pregunta Q3 ("Control del usuario y libertad") obtuvo valores bajos debido a la falta de funcionalidad para deshacer y rehacer acciones. Aunque el wiki-engine proporciona

Tabla 6.3: Cuestionario para desarrolladores de aplicaciones basadas en Fortunata.

ID	Objetivo de la evaluación	Pregunta
1	Complejidad	Horas dedicadas a crear la aplicación
2	Complejidad	Número de herramientas usadas para crear la aplicación
3	Complejidad	Nivel de dificultad para crear la aplicación
4	Colaboración	Nivel de dificultad para seguir las "ocho reglas de la usabilidad"
5	Colaboración	Nivel de dependencia del "código previo"
6	Colaboración	Nivel de dificultad para compartir el código fuente de su aplicación

mecanismos para deshacer, por medio de un control de versiones capaz de volver cualquier página wiki a un estado anterior, la funcionalidad implementada por los F-plugins debería ser capaz de dar soporte a esta capacidad. Las versiones actuales de OMEMO y VPOET no tienen esta capacidad.

Muchas preguntas (Q2, Q3, Q5, Q8, Q9) tuvieron valores altos de consenso (valores bajos de desviación estándar)) por parte de los evaluadores (desv. est. = 0.55). El menor consenso fue para Q6 (desv. est. = 0.84).

Las recomendaciones (cualitativas) de los evaluadores y su relación con las "ocho reglas de oro" se resumen en la tabla 6.4. Algunas recomendaciones fueron añadidas a la librería de Fortunata (e.g. Rec1), otras recomendaciones fueron guías genéricas (e.g. Rec2 and Rec3) que no han tenido una implementación concreta en la librería de Fortunata. Finalmente, hubo un grupo de recomendaciones que se pueden conseguir "trucando" JSPWiki (e.g. Rec4 and Rec5), pero otras recomendaciones no se pueden seguir porque la versión actual del wiki (JSPWiki version 2.4) no puede soportarlas (e.g. Rec6 and Rec7) o porque no se han implementado aún (e.g. Rec8). La "guía de usabilidad para dearrolladores basados en Fortunata", en la que se reúnen estas recomendaciones, se puede encontrar en `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=UsabilityRecomendations4Fortunata`.

Experimento de colaboración

Del cuestionario usado en el segundo experimento (véase tabla 6.3), se pueden considerar dos tipos de resultados,cuantitativos (preguntas Q1 y Q2), y cualitativos (Q3 a Q6). Las preguntas cuantitativas fueron diseñadas para medir el esfuerzo de desarrollo de las

Tabla 6.4: Agregación de recomendaciones de usabilidad de los evaluadores.

ID	Recomendación	R.	Solución proporcionada por el API de Fortunata
Rec1	Los mensajes de ejecución deben tener fijada una localización, unos colores y unos tamaños de letra	#1, #2	El API de Fortunata proporciona un método unificado para mostrar los mensajes de ejecución, con tres niveles distintos de avisos (“realizado con éxito”, “aviso” y “error”), visualmente diferenciados.
Rec2	Uso de enlaces a páginas de datos o ayuda para evitar tener que recordar códigos o identificadores.	#8, #7, #3	Recomendación general sin efectos en el API de Fortunata
Rec3	Los formularios deben caber en una sola página (para evitar scrolls de página). Se recomienda el uso de pestañas (tabs) en formularios grandes	#5, #7, #8	Recomendación general sin efectos en el API de Fortunata
Rec4	Redireccionamiento por medio de enlaces	#6	No hay redireccionamiento dinámico de páginas. Por tanto, la ejecución de un plugin no puede cambiar la página wiki. La solución es poner un enlace a la página destino en el texto del mensaje de ejecución del plugin.
Rec5	Mejorar las capacidades de los formularios	#4	JSPWiki impone algunas restricciones a los formularios: (1) sólo los botones pueden disparar los plugins, (2) no se permite Javascript, (3) No hay listas. Los evaluadores no encontraron en estas limitaciones problemas severos de usabilidad.
Rec6	Cambios dinámicos de skin	#2	La versión actual de JSPWiki soporta skins, pero no pueden cambiarse dinámicamente. La mayor parte de los skins soporta cambios de tamaño de letra.
Rec7	Capacidades Deshacer/Rehacer	#6	El wiki-engine proporciona un control de versiones de las páginas wiki, permitiendo deshacer/rehacer los contenidos de cada página wiki. Sin embargo, en lo relativo a la funcionalidad de los F-plugins, el deshacer/rehacer debe ser implementado por el desarrollador del plugin.
Rec8	Editores avanzados (e.g. código fuente coloreado, campos de texto con auto-completado)	#5, #4	Esta funcionalidad Esta funcionalidad no ha sido implementada aún en JSPWiki por ningún contribuidor. Estas capacidades servirían para mejorar, por ejemplo, el editor de plantillas de VPOET.

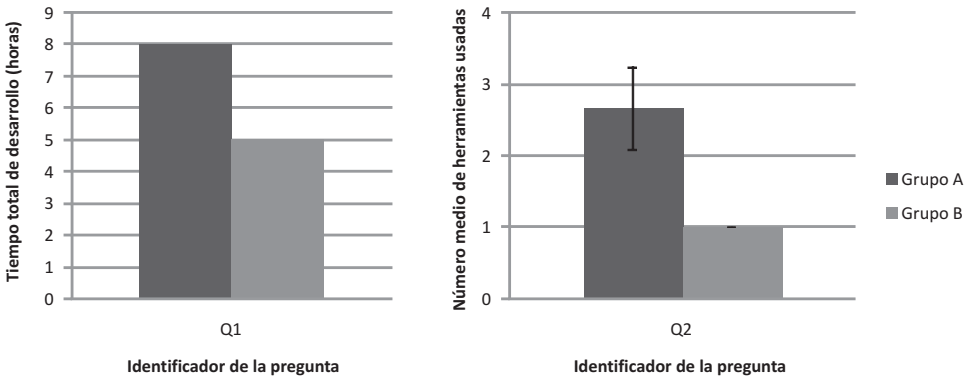

Figura 6.2: Dos primeras preguntas del cuestionario para desarrolladores. Tiempo total de desarrollo (Q1) y número medio de herramientas usadas (Q2) para dos tipos de desarrolladores: grupo de control (A) y grupo de Fortunata (B).

aplicaciones basadas en Fortunata (véase figura 6.2). Los resultados de Q1 muestran que el número de horas de desarrollo (suma de los tres pasos contributivos) se reduce alrededor de un 40 % si se usa la plataforma Fortunata. Los resultados obtenidos para Q2 muestran que el número de herramientas de desarrollo se reduce alrededor de un 60 %. Nótese que los tres desarrolladores del grupo B usaron únicamente la herramienta Fortunata, por lo que desv. est. = 0.0.

La figura 6.3 muestra los resultados del resto de las preguntas. En todas ellas la reducción por usar Fortunata está entre el 10 % (Q4) y el 60 % (Q6). El grupo A tuvo un consenso uniforme para todas las preguntas (desv. est. = 0.58) excepto para Q4 (desv. est. = 1.0). El grupo B tuvo también un consenso uniforme para todas las preguntas (desv. est. = 0.58) excepto para Q2 en el que el acuerdo fue completo (desv. est. = 0.0).

6.1.3. Conclusiones de la evaluación de Fortunata

Los experimentos realizados han permitió evaluar dos aspectos de Fortunata:(1) la calidad de las aplicaciones que se pueden generar con Fortunata, y (2) las ventajas que aporta a los desarrolladores de aplicaciones web semánticas. La calidad se ha medido por medio de la usabilidad de las aplicaciones en estados iniciales de su desarrollo (prototipos) mediante evaluaciones estándar de usabilidad. Estas medidas se ven confirmadas y ampliadas en estudios de usabilidad y satisfacción de usuario llevados a cabo en la evaluación de las versiones finales de las aplicaciones OMEMO y VPOET (véase sección 6.2). Las

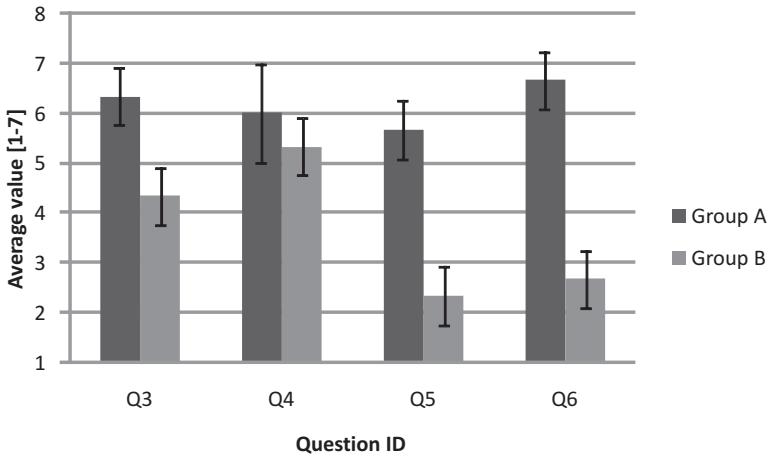

Figura 6.3: Resultados de las preguntas Likert del cuestionario para desarrolladores.

ventajas obtenidas por los desarrolladores que utilizan Fortunata son: (1) reducción de los tiempos de desarrollo, y (2)reducción de las competencias requeridas para desarrollar estas aplicaciones.

La principal limitación de esta evaluación es el reducido número de desarrolladores participantes en las pruebas, un total de seis. Sin embargo, los participantes mostraron bastante acuerdo en sus evaluaciones y un elevado nivel de satisfacción.

6.2. Evaluación de las herramientas OMEMO y VPOET

La sección anterior mostró que Fortunata facilita a los desarrolladores la creación de aplicaciones web semánticas. Los desarrolladores ven simplificada la generación de la interfaz de usuario, pero a cambio ven restringido el tipo de interfaces de sus aplicaciones por el wiki-engine usado, en el caso de Fortunata por JSPWiki. Por tanto, hay que comprobar que las aplicaciones creadas tienen unos valores razonables de calidad. Es decir, que las ventajas que obtienen los desarrolladores no son a cambio de una baja calidad de los interfaces de usuario. En concreto, la calidad de las aplicaciones se ha medido evaluando la usabilidad y el grado de satisfacción del usuario con respecto a la interfaz de usuario.

6.2.1. Descripción del experimento

Se seleccionaron quince personas post-graduadas de nuestra institución académica. Todos ellos tenían una base tecnológica y al menos conocimientos básicos de lenguajes web cliente tales como HTML, CSS o Javascript. Asistieron a una charla de 20 minutos en la que se les informó de las actividades que debían realizar en el rol de Diseñador Web. Esta charla introductoria les proporcionó una visión general de OMEMO y VPOET, siguiendo un tutorial on-line. Tras esta presentación, se les fijó un objetivo: crear una visualización para la clase `Person` definida en la ontología FOAF versión 20050403, utilizando el sistema propuesto. Se eligió esta clase por ser una de las más usada en la Web Semántica ($11 \cdot 10^6$ instancias en marzo de 2006) [Finin y Ding, 2006]. No se asignaron tiempos máximos o mínimos para realizar esta tarea. Se proporcionaron algunas fuentes de datos semánticos con instancias de la dicha clase para que los diseñadores web pudieran probar sus diseños.

Conviene recordar que los usuarios de VPOET pueden visualizar sus diseños utilizando fuentes de datos accesibles a través de la web, con el objetivo de detectar fallos en los diseños. Ejemplos típicos de estos errores son: mostrar de forma errónea propiedades multivaluadas, como `Person.knows`; o visualizaciones erróneas de propiedades sin valor, como `Person.title`. Una vez que el usuario ha terminado y probado su diseño, debe rellenar un cuestionario detallado que será mostrado en la siguiente sección.

6.2.2. Descripción del cuestionario

El objetivo de este cuestionario es medir la usabilidad de OMEMO y VPOET, así como el nivel de satisfacción del usuario con respecto a estas herramientas basadas en Fortunata. El análisis detallado de las respuestas nos permite obtener información detallada acerca de qué aspectos concretos son más difíciles de manejar o ser entendidos por los estos usuarios; y, por tanto, son susceptibles de ser mejorados. Como estas dos herramientas están orientadas a usuarios finales con experiencia en lenguajes web cliente, se consideró que las respuestas al cuestionario podían dependen del nivel de competencia de los usuarios con respecto a estas tecnologías. Por tanto, el cuestionario no comprende sólo preguntas relativas a las usabilidad o la satisfacción, sino que tiene partes adicionales relacionadas con la evaluación de los conocimientos del usuario y preguntas abiertas, lo que dio resultado a un detallado cuestionario (52 preguntas) divididos en 4 secciones:

1. Evaluación de los conocimientos de lenguajes cliente

Tabla 6.5: Estructura del cuestionario presentado a los participantes.

Partes	**Preguntas**	
	#	**IDs**
Parte 1: Competencias del usuario		
Experiencia	1	Ex-Q1
Competencias específicas	8	Sk-Q1 - Sk-Q8
Parte 2: Usabilidad		
Bloque 1: Aprendizaje	4	U-Q1 - U-Q4
Bloque 2: Realimentación y errores	4	U-Q5 - U-Q8
Bloque 3: Adaptación al usuario	5	U-Q9 - U-Q13
Bloque 4: Aspectos más positivos y menos	2	U-Q14 - U-Q15
Parte 3: Satisfacción		
Bloque 1: Reacción general al software	6	S-Q1 - S-Q6
Bloque 2: Pantalla	3	S-Q7 - S-Q9
Bloque 3: Terminología e Información del sistema	5	S-Q10 - S-Q14
Bloque 4: Aprendizaje	6	S-Q15 - S-Q20
Bloque 5: Capacidades del sistema	5	S-Q21 - U-Q25
Parte 4: Preguntas abiertas	3	O-Q1 - O-Q4
Total	**52**	

2. Evaluación de la usabilidad de las herramientas OMEMO y VPOET

3. Evaluación del grado de satisfacción del usuario con respecto a la interfaz de usuario

4. Preguntas abiertas

Las partes 2 y 3 son cuestionarios estándar [2], descritos en profundidad en la sección 6.2.4, divididos por bloques de preguntas. La tabla 6.5 muestra estas partes y bloques, el numero de cuestiones de que se compone cada uno, y el identificador de cada pregunta. Cada parte se detalla en las siguientes subsecciones.

6.2.3. Evaluación de las competencias de cada participante

El nivel de competencia de una usuario fue medido utilizando dos variables diferentes. La primera de ellas, *experiencia* (años de experiencia trabajando con estas tecnologías)

[2]Véase `http://oldwww.acm.org/perlman/question.html`

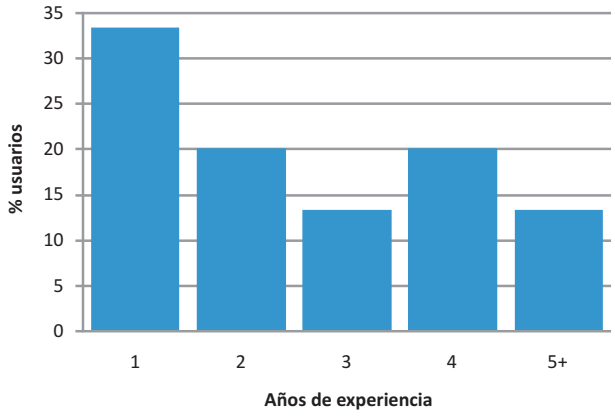

Figura 6.4: Experiencia en tecnologías cliente (en años) de los participanes.

proporciona una medida sencilla, de grano grueso, de las competencias de los usuarios en lenguajes web cliente.

La segunda, *competencias*, proporciona una medida de grano fino del nivel de competencia en cada una de las mismas. Los participantes de esta prueba iban desde 1 año de experiencia y conocimientos tan básicos con HTML básico, hasta usuarios que utilizan las últimas tendencias y tecnologías, como los mashups.

La variable experiencia fue medida en años, con 5 valores diferentes: 1, 2, 3, 4, 5+, este último para usuarios con 5 o más años. Por tanto, esta variable sólo puede tomar valores positivos.

En la fig 6.4 se muestra el histograma de la experiencia de los participantes. En esta figura se puede ver que más del 30 % de los participantes tiene sólo un año de experiencia en tecnologías web cliente, y que sólo el 13 % de los participantes tiene más de 5 años de experiencia en el uso de estas tecnologías. La hipótesis de este trabajo es que las últimas tecnologías cliente, conocidas como Web 2.0, no se utilizan tanto como cabría esperar debido a que están orientadas más a programadores que a diseñadores gráficos.

A fin de determinar el grado de adopción de las tecnologías anteriormente citadas por parte de nuestra muestra de usuarios, se usó una segunda variable, llamada *competencia*. Esta variable fue medida con una auto-evaluación sobre ocho temas distintos, y fue representada por valores enteros en un rango desde 1 (básico) a 5 (avanzado), incluyendo también como opción “N.A” como una posible respuesta (“No Aplicable”). La tabla 6.6 muestra los temas que se usaron para evaluar las competencias de los participantes.

Tabla 6.6: Preguntas utilizadas para medir las competencias de los participantes en tecnologías cliente.

Parte 1. Conocimientos en tecnologías cliente
Sk-Q1. HTML
Sk-Q2. CSS
Sk-Q3. DHTML(Javascript + DOM)
Sk-Q4. AJAX básico (Javascript + invocación asíncrona + DOM)
Sk-Q5. AJAX avanzado nivel 1 (AJAX básico + uso de XML/JSON)
Sk-Q6. AJAX avanzado nivel 2 (AJAX básico + uso de API's = mashups)
Sk-Q7. Flash básico
Sk-Q8. Flash avanzado (intercambio de datos XML y/o comunicación con Java/otros)

Se ha calculado el valor de *competencia* como la suma de los valores numéricos libremente auto-asignados por cada usuario conforme a su nivel de competencia en cara uno de los temas de la tabla 6.6. Puesto que el máximo valor de cada tema es 5, el mayor valor de competencia es 40. Aunque tanto *experiencia* como *competencia* son variables discretas, la segunda tiene un rango de valores mucho mayor. La figura 6.5 muestra las competencias de los participantes, con los temas organizados por columnas. Por ejemplo, en la columna 'HTML' se puede observar que alrededor del 40 % de los participantes declara conocimientos avanzados en esta tecnología (rectángulo punteado). Hay que destacar que este es el único tema en el que alguno de los participantes manifiesta conocimientos avanzados. También es destacable que hasta el 80 % de los participantes manifestaron no tener conocimientos (rectángulo negro) en tecnologías de última generación (AJAX o Flash).

Estos resultados avalan la hipótesis de que las últimas tecnologías cliente no se utilizan tanto como cabría esperar, al menos para el conjunto de usuarios utilizado en este experimento.

Para saber la relación entre los años de experiencia y los conocimientos, se han creado unos gráficos que muestran *competencia vs. experiencia*. Estos gráficos se muestran en las figuras 6.6 y 6.7. Los participantes con menos experiencia en tecnologías cliente muestran los valores más bajos de competencia, como parece logico; sin embargo, es destacable que la competencia alcanza un máximo entre 3-5 años de experiencia y después decrece. Por tanto, los usuarios con más años de experiencia no son los que tienen el mayor nivel de competencia.

Esto podría ser razonable para usuarios principiantes, ya que dominar las últimas tecnologías requiere dominar las tecnologías anteriores, y esto requiere tiempo, y por tanto

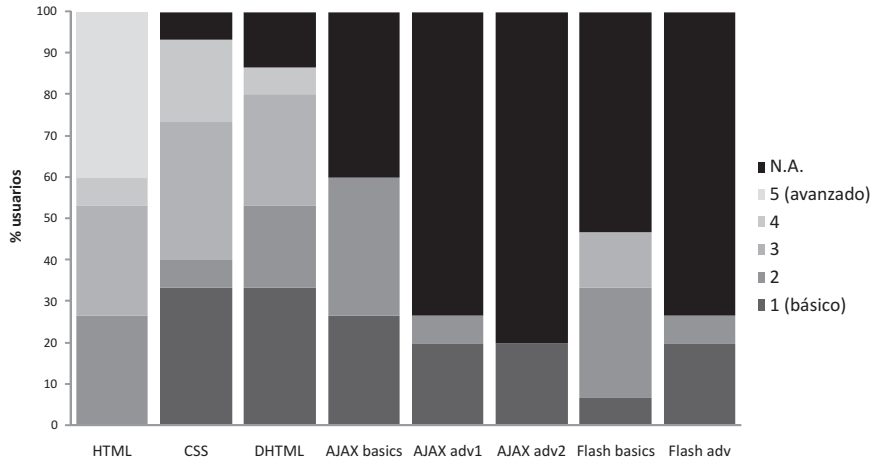

Figura 6.5: Nivel de competencia en tecnologías cliente de los participantes.

experiencia. Sin embargo, en el caso de usuarios experimentados, los resultados obtenidos sugieren que estos usuarios no utilizan las nuevas tecnologías por alguna razón.

Se puede explicar el máximo alrededor de 3-5 años con el argumento de que aunque las tecnologías AJAX se encuentran disponibles en los navegadores más usados desde 2002[3], la popularización de AJAX viene de la primera "Web 2.0 Conference" que tuvo lugar en 2004. Nuestra hipótesis es que las últimas tecnologías están orientadas a programadores, rompiendo con la tendencia de orientación a usuarios finales de las tecnologías cliente previas. Esta nueva tendencia resulta compleja para los usuarios veteranos, y en 2004 se produjo una fractura que hizo que únicamente los usuarios principiantes adoptasen estas tecnologías. Hoy, esos usuarios tienen entre 3 y 5 años de experiencia.

Esta hipótesis es consistente con este trabajo de tesis en el sentido de que las plataformas basadas en wikis son capaces de simplificar el uso de las últimas tecnologías, evitando tener que programar, y por tanto son útiles para la mayoría de los usuarios.

[3]AJAX existe desde que el IE5.0 implementó XMLHTTPRequest (marzo de 1999). Mozilla lo implementó en su versión 1.0 (2002). Aunque hay un Working Draft del W3C, aún no hay una especificación estándar.

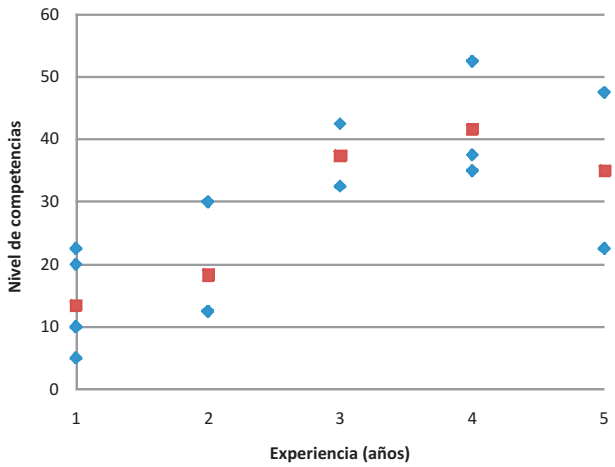

Figura 6.6: Nivel de competencias frente a años de experiencia. Los cuadrados muestran el valor medio.

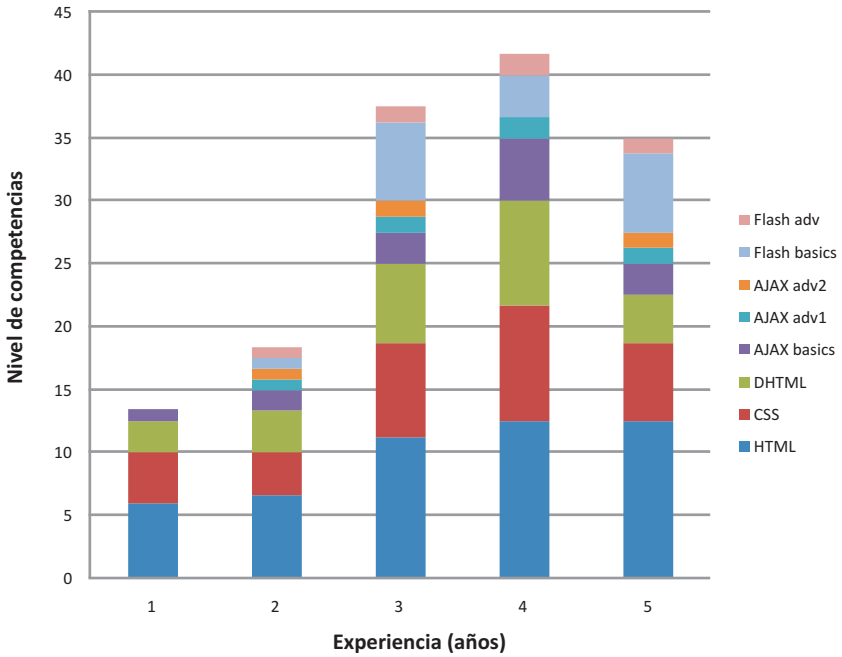

Figura 6.7: Detalle del nivel de competencias frente a la experiencia (años).

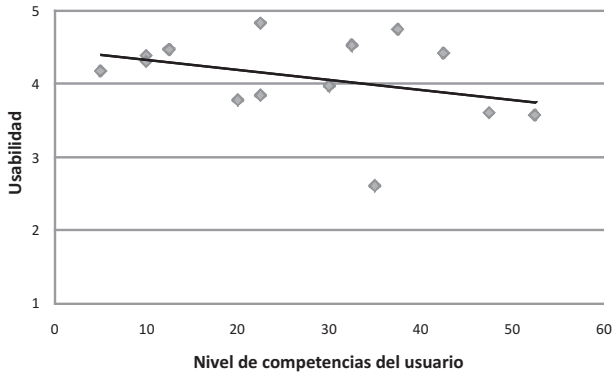

Figura 6.8: Usabilidad en función del nivel de competencias del usuario.

6.2.4. Evaluando la usabilidad en las herramientas basadas en Fortunata

La usabilidad puede ser considerada uno de los parámetros más importantes de calidad en la producción de software hoy en día. El éxito de un sistema está muy relacionado con lo que los usuarios dicen que les gusta [DeLone y McLean, 1992; Delone y McLean, 2003]. Para obtener una medida de la usabilidad del sistema propuesto se usó un test estándar: "Practical Heuristics for Usability Evaluation" [Perlman, 1997]. Como se mostró en la tabla 6.5, este test se compone de 4 bloques de preguntas (en total 13 preguntas) acerca de distintos aspectos del uso de la aplicación, con valores que van desde 1 (malo) a 5 (bueno), proporcionando una medida efectiva de la usabilidad del sistema percibida por el usuario.

Estas preguntas se agruparon en cuatro grupos: "Aprendizaje" (4 preguntas), "Realimentación y errores" (4 preguntas), "Adaptación al usuario" (5 preguntas), y dos preguntas abiertas (U-Q14 y U-Q15). La primera pregunta solicita una lista de los aspectos más positivos, y la segunda solicita una lista de los aspectos más negativos de la herramienta. Como ejemplos de aspectos positivos apuntados por los usuarios podemos mencionar "la simplicidad del sistema", o "el Google Gadget de VPOET es muy útil". Los aspectos negativos más demandados son "facilidades gráficas para insertar las macros" o "más ejemplos en el tutorial".

El análisis de la usabilidad se basó en el valor promedio asignado en el test mostrado anteriormente, que resulta en un variable continua denotada *usabilidad*. Los valores obtenidos se muestran en la figura 6.8. El valor promedio obtenido es 4.1 (un buen valor considerando que los valores posibles se mueven en el rango [1-5]), con una desviación

estándar de 0.595. Los usuarios con mayores competencias tienen una ligera tendencia a asignar un valor menor de usabilidad. Una posible explicación es que este tipo de usuarios son más exigentes, lo que resulta en asignaciones más bajas a la usabilidad. La correlación lineal (mostrado como R^2) con *competencias* es muy baja: 0.121. La correlación lineal entre *usabilidad* y *experiencia* es incluso menor (R^2=0.014). Una posible interpretación es que, aunque es una tendencia ligera, la usabilidad está más correlaciona con las competencias que con la experiencia porque *competencias* es una mejor medida del nivel de conocimiento que *experiencia*. Con el fin de encontrar qué factores determinan mejor la usabilidad, se llevó a cabo un análisis en mayor profundidad .

Las preguntas relativas a la usabilidad están agrupadas en tres bloques principales (bloques 1-3), como se mostraba en la tabla 6.5. El bloque 4 fue rechazado por comprender las anteriormente citadas "Preguntas abiertas" y no poder extraer de manera sencilla un valor numérico de las respuestas. El valor promedio de las respuestas de estos tres bloques proporcionó el valor de *usabilidad* de cada participante.

Para medir el "peso" de cada bloque en el valor asignado de *usabilidad*, se calcularon las correlaciones lineales entre *usabilidad* y la media de cada uno de los bloques. Se encontró una correlación muy alta con el bloque 3 ("Adaptación al usuario", $R^2 = 0.930$) y con el bloque 2 ("Realimentación y errores", $R^2 = 0.891$). Este análisis nos permite concluir que las preguntas pertenecientes a estos bloques no son relevantes para los participantes a la hora de evaluar la usabilidad del sistema propuesto. La tabla 6.7 muestra las preguntas comprendidas en estos bloques relevantes.

Evaluación de la satisfacción del usuario con la interfaz de usuario

Para evaluar la satisfacción del usuario en lo relativo a la interfaz de usuario de las herramientas basadas en Fortunata se usó una variante ligeramente modificada del test estándar "User Interface Satisfaction" [Chin *et al.*, 1988]. En su versión estándar incluye 27 preguntas, pero fue reducido a 25 debido a solapamientos con las preguntas del test de usabilidad descrito en la sección anterior. Como muestra la tabla 6.5, estas 25 preguntas se organizan en 5 bloques conforme al test estándar: "Impresión general de la aplicación" (5 preguntas), "Presentación en pantalla" (3 preguntas), "Información de la terminología del sistema" (5 preguntas), "Aprendizaje" (6 preguntas), y "Capacidad del sistema" (5 preguntas). Las respuestas a estas preguntas deben ser valores enteros en el rango de 0 (valor mínimo) a 7 (valor máximo).

Tabla 6.7: Preguntas más relevantes en relación a la usabilidad del sistema.

Parte 2. Bloque 3. Realimentación y errores.
U-Q5. Proporciona mapas y un rastro. Proporciona una manera de ver adonde ir, qué sucederá.
U-Q6. Muestra al usuario qué es (o no es) posible.
U-Q7 Mapeos intuitivos. Los elementos visuales se corresponden con la acción que realizan.
U-Q8. Reducción al mínimo de la carga de memoria. Evita la necesidad de recordar. Facilidad para comparar de manera sencilla.
U-Q9. Consistencia y uso de estándares. El mismo término/acción se usa de forma consistente. Si no hay nada mejor, que sea conforme al estándar
Parte 2. Bloque 2. Adaptación al usuario
U-Q10. Realimentación
U-Q11. Prevención de errores. (Hace que sea difícil cometer errores)
U-Q12. Mensajes de error. Diagnostica correctamente la fuente y la causa de un problema y sugiere una solución.
U-Q13. Indicaciones claras de salidas y recuperación de errores. El usuario puede salir fácilmente de un estado indeseado. La aplicación asume que el usuario cometerá errores y necesitará recuperar estados anteriores

Para lograr una medida de la satisfacción del usuario, no se consideró el promedio de todas las respuestas, sino que se consideró que el primer bloque "Impresión general de la aplicación" podía ser considerado por sí mismo una medida de grano grueso de la satisfacción del usuario. De la media de este bloque resultó una variable *reacciónGlobal*. Una medida de grano fino, denotada *satisfacciónPromedio*, se obtuvo calculando la media del resto de bloques. Una vez calculadas esta variables se calculó la correlación entre ellas, encontrando un valor $R^2 = 0.97$. Este valor no se consideró lo suficientemente grande como para elegir una única variable para medir la satisfacción, por lo que ambas variables fueron consideradas en el análisis posterior. El valor promedio de *satisfacciónPromedio* y *reacciónGlobal* fue 5.8 (std. desv. = 0.74) y 6.06 (std. desv. = 0.53) respectivamente.

La figura 6.9 muestra la relación entre la satisfacción del usuario y sus competencias. Las dos variables elegidas como medida de la satisfacción muestran comportamientos parecidos. Es destacable que el comportamiento de la satisfacción frente a la competencias del usuario es muy similar al mostrado por la usabilidad, esto es, los usuarios con mayores niveles de competencias asignan valores ligeramente menores a la satisfacción. Esta tendencia es similar para ambas medidas de la satisfacción, con valores de correlación similares.

La tabla 4 muestra las preguntas con las que se calculó la variable *reacciónGlobal*

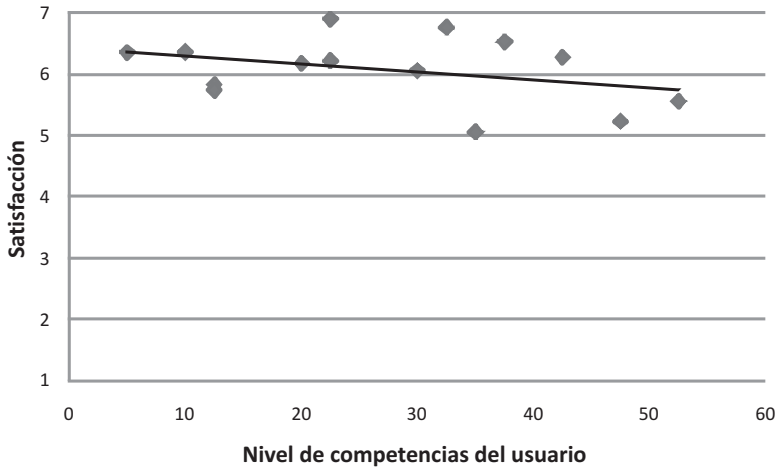

Figura 6.9: Satisfacción en función del nivel de competencias del usuario.

Tabla 6.8: Preguntas usadas para medir *reacciónGlobal*.

Parte 3. Bloque 1. Impresión general de la aplicación
S-Q1. terrible - maravilloso
S-Q2. difícil - fácil
S-Q3. frustrante - satisfactorio
S-Q4. potencia inadecuada - potencia adecuada
S-Q5. poco estimulante - muy estimulante
S-Q6. rígido - flexible

(bloque 1, parte 3 del cuestionario). En realidad no son preguntas, sino nivel de acuerdo entre dos extremos, con valores en el rango [1,7].

Para evaluar las dependencias de *satisfacciónPromedio* con respecto a sus 4 bloques de preguntas, se calcularon las correlaciones lineales con las medias de cada bloque. Los resultados, ordenados por R^2, fueron: bloque 4 (“Aprendizaje”, 0.900), bloque 5 (“Capacidad del sistema”, 0.854), bloque 2 (“Presentación en pantalla”, 0.700), y bloque 3 (“Información de la terminología y del sistema”, 0.167). Por tanto, los bloques 4 (“Aprendizaje”) y 5 (“Capacidad del sistema”) son los más relevantes, mientras que el bloque 3 (“Información de la terminología y del sistema”) no es muy relevante, y el bloque 2 puede ser considerado irrelevante. Aplicando este método a *reacciónGlobal*, y ordenando por R^2, se obtiene el mismo orden de bloques pero con “pesos” bastante diferentes: bloque 4 (“Aprendizaje”, 0.870), bloque 5 (“Capacidad del sistema”, 0.0.647), bloque 2 (“Presentación en pantalla”,

Tabla 6.9: Preguntas más relevantes en relación a la satisfacción del usuario con el interfaz gráfico del sistema.

Parte 3. Bloque 4. Aprendizaje
S-Q15. Aprendizaje del funcionamiento del sistema (difícil-fácil)
S-Q16. Exploración de nueva funcionalidad por prueba y error (difícil-fácil)
S-Q17. Recuerdo de nombres y uso de comandos (difícil-fácil)
S-Q18. La realización de tareas es directa(nunca-siempre)
S-Q19. Mensajes de ayuda (inútiles-provechosos)
S-Q20. Material de referencia suplementario (confuso-claro)
Parte 3. Bloque 5. Capacidad del sistema
S-Q21. Velocidad del sistema (muy lento - rápido)
S-Q22. Confiabilidad del sistema (No fiable - fiable)
S-Q23. El sistema tiende a ser. . . (ruidoso - estable)
S-Q24. Corrección de errores del usuario (difícil - fácil)
S-Q25. Diseñado para todos los niveles de usuarios (nunca - siempre)

0.0.362), y bloque 3 (“Información de la terminología y del sistema”, $5 \cdot 10^{-5}$). De este análisis concluimos que los bloques 4 y 5 son los más relevantes para la usabilidad del sistema. La tabla 6.9 muestra las preguntas de estos bloques.

Preguntas abiertas

La última parte del cuestionario, denotada como parte 4 en la tabla 6.5, comprende tres preguntas abiertas:

1. ¿Para qué cree que puede servir esta herramienta?

2. ¿Cómo cree que esta herramienta puede ayudarle en su trabajo habitual?

3. Comentarios adicionales

6.2.5. Análisis posterior

Se han realzado análisis experimentales adicionales con el fin de clarificar algunas cuestiones surgidas tras los resultados preliminares. Estas cuestiones se detallan en las siguientes subsecciones.

Tabla 6.10: Preguntas sobre tecnologías web cliente para diseñadores web profesionales.

Grado de acuerdo con las siguientes sentencias
Q1. Integrar diseño y funcionalidad (desarrollada por los programadores) es cada vez más complejo
Q2. Los diseños web multi-navegador son cada vez más complejos
Q3. Los diseños web multi-dispositivo son muy complejos
Q4. Con las tecnologías AJAX se ha perdido calidad estética en las aplicaciones web
Q5. Las tecnologías AJAX son para programadores, no para diseñadores web
Q6. Las tecnologías AJAX han complicado mucho la realización de diseños para las aplicaciones web
Q7. Hay que aprender demasiados lenguajes de programación
Q8. La creación de diseños para la Web Semántica es una tarea compleja

Nivel de competencia para usuarios académicos y profesionales

A pesar de que los participantes descritos es la sección anterior cubren un amplio rango en experiencia y competencias en el área de las tecnologías web cliente, todos ellos provienen del entorno académico. Con el objetivo de comprobar si la máxima competencia se alcanza alrededor de los 3-4 años de experiencia para los usuarios profesionales del diseño web, se creó un cuestionario más ligero. Este cuestionario comprende la parte 1 del test inicial así como una nueva parte en la que se solicita el nivel de acuerdo con algunas consultas predefinidas. Estas consultas se pueden ver en la tabla 6.10.

Este cuestionario requería diseñadores web profesionales con al menos 3 años de experiencia en tecnologías web cliente, y fue respondido por 10 personas de distintas compañías (asistentes al congreso MadInSpain [4], referente internacional para los expertos en diseño gráfico).

Se encontró el mismo resultado, esto es, las competencias alcanzan un máximo alrededor de los 4 años de experiencia, decayendo para experiencias mayores, como puede verse en la figura 6.10.

El valor promedio asignado a cada una de las preguntas, y su desviación estándar, se puede ver en la figura 6.11. Del análisis de las respuestas se pueden destacar algunos resultados. La pregunta en la que más participantes están de acuerdo es Q7 (“Hay que aprender demasiados lenguajes de programación”), con el máximo valor de acuerdo (4.5) y el máximo consenso (la mas baja desviación estándar, con valor 0.7). Este resultado es consistente

[4] Véase `http://www.madinspain.com/`

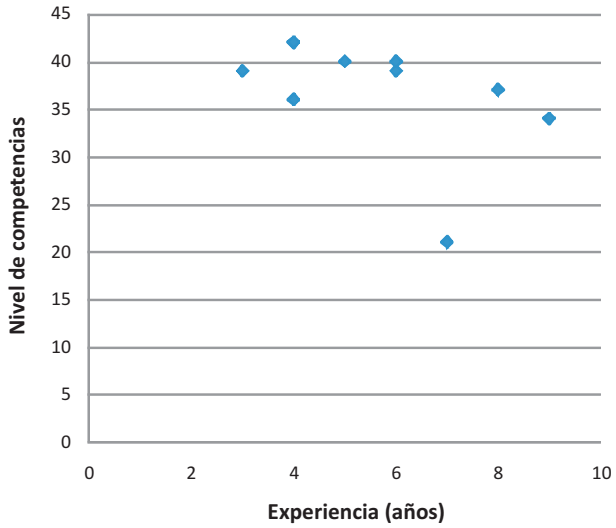

Figura 6.10: Usabilidad en función de las competencias del usuario.

con la hipótesis de trabajo.

Sin embargo, el menor acuerdo se da para Q4 (“Con las tecnologías AJAX se ha perdido calidad estética en las aplicaciones web”), con valor 2.1 y desviación 0.99. Este resultado muestra que a pesar de que AJAX ha añadido un mayor nivel de complejidad a las tecnologías web cliente, los diseñadores aún pueden proporcionar diseños con estéticas de alta calidad.

Independencia de la satisfacción del usuario con la interfaz y la usabilidad del sistema

Este estudio consideró que la usabilidad y la satisfacción son variables independientes. Esto provocó que existiesen las partes 2 y 3 del cuestionario. Esta suposición pudo ser confirmada una vez que las variables fueron correlacionadas. La correlación entre satisfacción y usabilidad para los participantes académicos se muestra en la figura 6.12. Como medida de la satisfacción del usuario se usaron las variables *reacciónGlobal* y *satisfacciónPromedio*. Se observa que la variable *reacciónGlobal* está ligeramente más relacionada con *usabilidad* ($R^2 = 0.788$) que *satisfacciónPromedio* ($R^2 = 0.632$). Sin embargo, ninguno de estos valores es lo bastante alto como para que podamos deducir el valor de satisfacción a partir del valor de usabilidad, confirmando nuestra hipótesis de que estas variables deben ser evaluadas por separado. En nuestro caso, mediante formularios distintos.

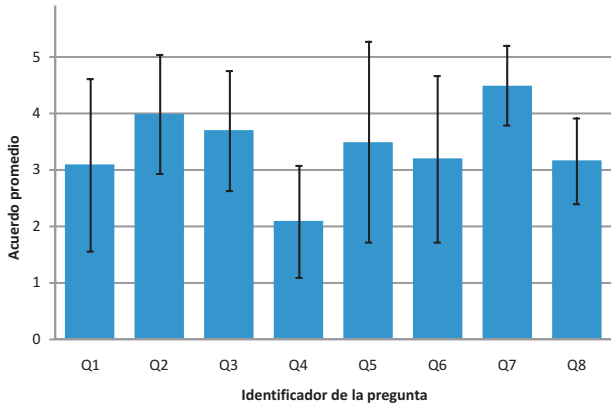

Figura 6.11: Grado de acuerdo de los diseñadores profesionales con las preguntas.

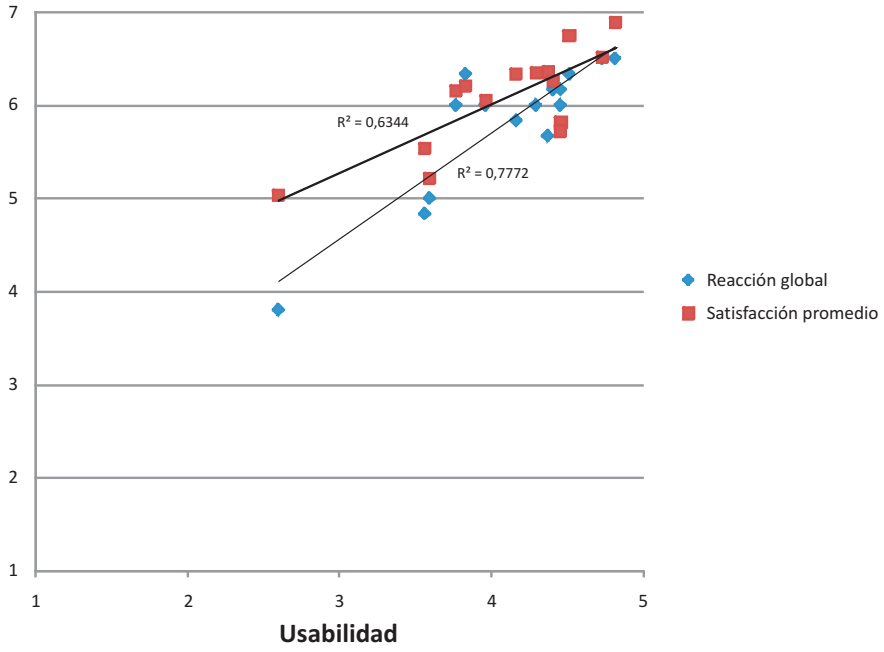

Figura 6.12: Correlación entre satisfacción con la interfaz de usuario y la usabilidad.

Búsqueda de dependencias no lineales

Una objeción al análisis de las dependencias es que sólo se consideran correlaciones lineales. Para asegurar que no hay correlaciones no lineales se usaron las técnicas estadísticas descritas a continuación.

Para obtener una medida de la dependencia no-lineal de *reacciónGlobal* respecto de los cuatro bloques que la componen, se usó un perceptrón multicapa. La capa de salida predice *reacciónGlobal* basado en las otras 4 variables (cada variable es el promedio de las respuestas del bloque correspondiente), que alimentan la capa de entrada. El trabajo de [Hornik, 1991] estableció que cualquier sistema no lineal puede ser modelado por un perceptrón multicapa utilizando una única capa oculta. Se probó este clasificador, para diferentes números de unidades en la capa oculta, utilizando un método de validación cruzada denominado "leave-one-out" para predecir los valores de salida. Este método entrena el perceptrón con todas las muestras excepto una. Esta es luego usada para probar el perceptrón, obteniendo un error en la predicción. Aplicando este procedimiento a todas las muestras, se obtiene el error cuadrático medio mostrado en la figura 6.13. En esta figura se puede observar que el hecho de incrementar el numero de unidades ocultas resulta en valores de error mayores. La mejor capacidad de predicción se obtiene para 0 unidades, esto es, mediante una aproximación lineal.

En la figura se pueden ver diversas arquitecturas de capas ocultas con las que se experimentó. Por ejemplo, la arquitectura [1,1,X] corresponde a tres capas, la primera con una unidad, la segunda con una unidad, y la tercera con un número variable de unidades (entre 1 y 10) en la capa oculta.

Se siguió este procedimiento para otras posibles variables dependientes, como *satisfacciónPromedio* (dependencias de los mismos cuatro bloques) o *usabilidad* (dependencia de sus tres bloques constituyentes), encontrando resultados similares, y concluyendo que no existen relaciones no-lineales entre las variables consideradas.

Tipos de usuarios

Otro de los objetivos ha sido descubrir si hay diferentes "tipos de usuarios" en nuestra muestra que corroboren los resultados obtenidos en la sección 6.2.3 en los que se detectó un máximo en el nivel de competencias alrededor de los 4 años de experiencia (véase figura 6.6). Para explorar esta característica se utilizó un método de clustering denominado X-Means [Pelleg y Moore, 2000]. Este algoritmo es una versión extendida y mejorada del

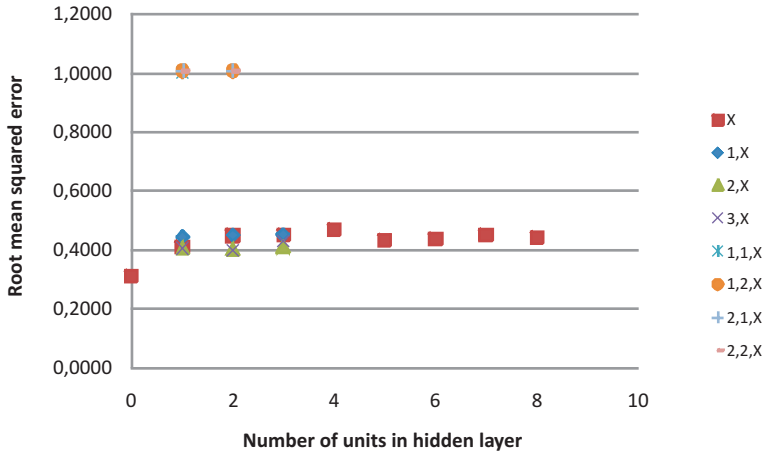

Figura 6.13: Perceptrón multicapa usado para detectar no linearidades

método K-Means [Duda *et al.*, 1973; Bishop, 1995]. Se aplicó este método para encontrar agregaciones en un espacio 8-dimensional utilizando estas variables: *experiencia*, *competencias*, *usabilidad*, *reacciónGlobal*, y los cuatro bloques que comprende la satisfacción del usuario (*presentación*, *terminología*, *aprendizaje*, y *capacidadSist*).

Se encontraron dos clusters con tamaños parecidos (8 y 7 usuarios). Para descubrir qué variables proporcionan una buena visualización en dos dimensiones de estos clusters, se realizó una búsqueda sistemática utilizando la herramienta de minería de datos Weka [5].

Se encontró que *experiencia* es la variable clave que separa la muestra, encontrando una separación óptima en combinación con la variable *reacciónGlobal*. La proyección de este espacio 8-dimensional sobre estas dos variables da lugar a la figura 6.14.

La frontera entre estos dos clusters se encuentra alrededor de los 2.5 años, dividiendo a los usuarios en dos grupos: usuarios con menos de 2.5 años (que serán denominados "principiantes"), y usuarios son más experiencia ("experimentados"). Los usuarios de cada cluster comparten características comunes: por ejemplo, los usuarios "principiantes" asignan un valor de satisfacción similar, mientras que los usuarios "experimentados" asignan un valor de satisfacción en un rango mucho más amplio y en media asignan valores ligeramente menores. Los centros de cada cluster pueden ser considerados como el usuario prototipo de cada cluster. El centro de los usuarios "principiantes" se encuentra en 1.3 años de experiencia (asignando un valor de *reacciónGlobal* de 6.0), mientras que el prototipo de usuario "experimentados" tiene alrededor de 4.0 años de experiencia (asignando una valor

[5] Véase `http://www.cs.waikato.ac.nz/ml/weka/`

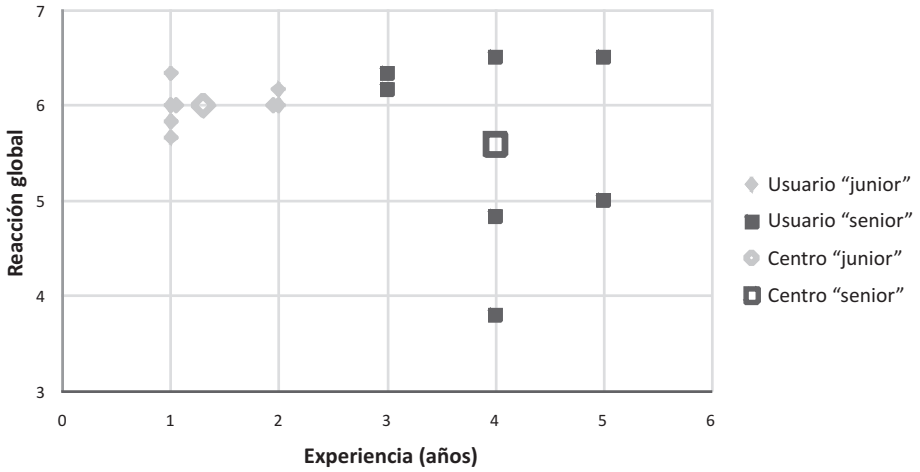

Figura 6.14: Dos tipos de usuarios: principiantes (junior) y experimentados (senior)

de *reacciónGlobal* de 5.5).

Como se mostró en la sección 6.2.3, los usuarios alcanzan un máximo de competencias alrededor de los 3-5 años de experiencia. Por tanto, los usuarios más competentes se encuentran dentro del cluster de usuarios “experimentados”, compartiendo con ellos las características del cluster, como los patrones de asignación de valores para usabilidad y satisfacción.

6.2.6. Conclusiones de la evaluación de OMEMO y VPOET

En lo relativo a los usuarios, hay que destacar que la mayor parte de ellos tienen conocimientos básicos de las últimas tendencias en tecnologías web cliente. Este hecho prueba la hipótesis de que la creación de diseños gráficos para web es una tarea compleja, lo que nos anima a seguir investigando en nuevas formas de simplificar esta tarea.

En lo que respecta al experimento llevada a cabo utilizando OMEMO y VPOET, se ha encontrado un valor razonable para la usabilidad así como para el grado de satisfacción con la interfaz de usuario. En un rango [0-10], estas herramientas han obtenido valores bastante buenos de usabilidad (8.2) y satisfacción de la interfaz (8.7). El coste de desarrollo de estas herramientas es menor que el coste de desarrollar aplicaciones desde cero, en términos de grado de conocimientos de programación requeridos y tecnologías web cliente. Esto nos anima a desarrollar más herramientas basadas en Fortunata ya que proporciona un buen balance entre coste y satisfacción/usabilidad del usuario.

La usabilidad de estas herramientas es ligeramente mayor para usuarios con menores competencias, y depende especialmente de los bloques de preguntas relativas a “Adaptación al usuario” y “Realimentación y errores”. La satisfacción del usuario sigue una tendencia similar: mejores valores para los usuarios con menores competencias. La satisfacción depende especialmente de los bloques de preguntas “Aprendizaje” y “Capacidades del sistema”.

Se ha podido clasificar a los participantes del experimento en dos perfiles de usuario básicos: aquellos con menos de 2.5 años de experiencia, y los que tienen mayor experiencia. Hay que destacar que más experiencia no implica mayores niveles de competencia, ya que los usuarios con más competencias se encuentran alrededor de los 3-5 años de experiencia. Este último dato ha demostrado ser el mismo, tanto para los participantes académicos, como para los profesionales del diseño gráfico de las distintas empresas consideradas.

Capítulo 7

Conclusiones y líneas futuras

7.1. Conclusiones

Como resultado de las investigaciones realizadas en el marco de esta tesis doctoral se ha desarrollado una plataforma software y diversas aplicaciones, con el objetivo de reducir la brecha tecnológica que separa a los expertos de la Web Semántica de los desarrolladores de aplicaciones web. La estrategia empleada se centra en lograr que los expertos de la Web Semántica puedan crear aplicaciones web con facilidad, y en lograr que los desarrolladores y usuarios puedan manejar datos de la Web Semántica.

Para el primer objetivo se ha propuesto una plataforma llamada Fortunata, dirigida a desarrolladores con conocimientos de tecnologías de la Web Semántica. Para el segundo objetivo se ha repartido la tarea entre especialistas en tecnologías web cliente (diseñadores web) y desarrolladores de la parte servidora de aplicaciones web. Para los diseñadores web se han creado dos aplicaciones web, llamadas OMEMO y VPOET, con las que pueden aplicar sus competencias especializadas para crear unos interfaces web atractivos y reutilizables capaces de manejar datos semánticos. Los desarrolladores pueden aprovechar los diseños de los diseñadores web para crear fácilmente aplicaciones web capaces de gestionar datos semánticos. Como ejemplo del tipo de aplicaciones que pueden crear los desarrolladores se ha creado un Google Gadget que permite que los usuarios finales pueden usar datos semánticos en sus propias páginas web.

Las soluciones aportadas han tenido por líneas maestras la reducción de competencias o el nivel de las mismas, así como la reutilización. La creación de aplicaciones web semánticas basadas en Fortunata se apoya en el concepto de colaboración por contribución, una extensión de la contribución por contenidos en la que la colaboración se basa en aportar funcionalidad en lugar de contenidos.

La evaluación experimental de la plataforma Fortunata ha constado de dos fases. En la primera, especialistas en usabilidad han evaluado los primeros prototipos de las aplicaciones creadas con Fortunata para corregir las principales deficiencias y elaborar una guía de usabilidad para creadores de aplicaciones web basadas en Fortunata. En la segunda fase, y sobre las aplicaciones finales, se ha medido la usabilidad y la satisfacción del usuario, obtenido resultados favorables que permiten concluir que con Fortunata se pueden crear aplicaciones con unos valores elevados de calidad, tan sólo limitadas por algunas limitaciones del wiki-engine utilizado. Por ejemplo, los test de usuario mostraron que algunos participantes echaron de menos más facilidades gráficas para insertar las macros o para colorear automáticamente el código fuente. Aunque la comunidad JSPWiki es muy activa,

este tipo de funcionalidad puede tardar en ser proporcionada.

7.2. Líneas de investigación futuras

7.2.1. Perfiles semánticos de usuario

La definición semántica de perfiles de usuario, iniciada con MIG, puede permitir la generación de interfaces adaptadas a las características del usuario, desde sus limitaciones interactivas (e.g. daltonismo o reducida agudeza visual), el dispositivo utilizado, o sus preferencias estéticas. Para avanzar en esta línea hay que resolver algunas limitaciones actuales, como son la composición de plantillas,la interacción entre plantillas, o algoritmos que permitan elegir de manera adecuada la plantilla más adecuada a un perfil dado.

7.2.2. Utilización de plantillas VPOET

Las plantillas de VPOET pueden ser utilizadas por usuarios comunes para mostrar de manera atractiva información semántica en entornos como páginas personales (por medio de GG-VPOET), pero se pueden implementar adaptaciones para otros entornos como wikis o portales semánticos. También se podrían adaptar a los Semantic Pipes para proporcionar una interfaz de usuario más amigable que la interfaz actual basada en tablas de tripletas (para salida) o campos de entrada sencillos (para entrada).

La integración de los templates de VPOET en un entorno wiki (por ejmplo, JSPWiki) estaría condicionada a que el wiki soporte la incorporación de nuevas etiquetas a la sintaxis del wiki (como es el caso de JSPWiki). Para generar el mensaje HTTP necesario para explotar las plantillas de VPOET, en el caso de JSPWiki se podría crear un plugin, que sería invocado por los usuarios en cualquier página wiki mediante un texto como este:

```
[{VPOETPlugin action='renderOutput' designID='X' provider='Y'
 source='Z' width='400' height='300'}]
```

7.2.3. My Data Are Mine (MDAM)

Es una iniciativa que aspira a ampliar los derechos que tienen los usuarios sobre sus datos. Por ejemplo, la legislación española actual permite que un usuario de una compañía solicite el borrado de todos los datos que dispone la compañía acerca del usuario. Pero

muy pocas compañías permiten que el usuario obtenga sus datos en formatos electrónicos estandarizados (como es el caso de algunos bancos que permiten, por ejemplo, descargar los movimientos de una cuenta en formato Excel).

Si el usuario dispusiera de sus datos en formatos como RDF, podría extenderse el uso de la reutilización de información. El usuario no tendría que proporcionar una y mil veces los mismos datos al registrarse en sitios web o para realizar transacciones electrónicas. Con indicar su repositorio de datos sería suficiente.

Evidentemente, habría que establecer un protocolo de actuación que permitiera al usuario saber a qué tipo de datos va acceder la compañía. En caso de que el usuario diese su visto bueno, la compañía accedería a los datos solicitados y quedaría suscrita a posibles cambios en la fuente original. Este mecanismo de suscripción evitaría, por ejemplo, la situación que se produce cuando un usuario cambia de domicilio. En lugar de tener que informar a todas y cada una de las compañías que envían su correo postal, serían estas las que serían notificadas del cambio de domicilio cuando el usuario cambiase la información de su fuente de datos.

Cuantos más datos pone un usuario en la Web (siguiendo la filosofía Web 2.0) más apremiante se hace esta necesidad. Por ejemplo, si tras varios años de publicar en un blog el usuario decide cambiar de proveedor de blog, ¿dispone de herramientas para mover sus textos, fotos, vídeos, etc, al nuevo blog?. Aunque hay algunas iniciativas en esta dirección, el problema sigue siendo la multiplicación de la información. Esta necesidad de migración de datos también se hace evidente en redes sociales web, en las que las facilidades para la migración de datos son inexistentes.

Apéndices

Apéndice A

Cuestionario de evaluación de VPOET y OMEMO para usuarios no profesionales

Cuestionario para diseñadores gráficos especializados en web. Nivel básico.

Este cuestionario pretende medir diversos parámetros de la interacción entre usuarios finales y la herramienta VPOET. Estos parámetros permitirán medir la usabilidad y el nivel de satisfacción de los usuarios. Se asume que los usuarios finales tienen unos conocimientos mínimos de los lenguajes de creación de interfaces para Web. En concreto se asumen conocimientos elementales de HTML, CSS y, opcionalmente, javascript.

Caracterización del diseñador

Años de experiencia con los lenguajes HTML, CSS o javascript	<1	2	3	4	5+	N.A.

Conocimientos de tecnologías web

	básico	1	2	3	4	5	avanzado	N.A.
HTML								
CSS								
DHTML (Javascript + DOM)								
AJAX básico (Javascript + Invocación asíncrona + DOM)								
AJAX avanzado 1 (AJAX básico + tratamiento de XML/JSON)								
AJAX avanzado 2 (AJAX básico + uso de APIs = mashups)								
Flash básico								
Flash avanzado (intercambio de datos XML y/o comunicación con Java/Otros)								

Usabilidad de la herramienta VPOET (Based on: **Perlman, G. (1997) *Practical Usability Evaluation*.**)

	malo	1	2	3	4	5	bueno	N.A.
Ayuda y documentación								
Adaptación al punto de vista del usuario								
Diálogos simples y naturales								
Facilidades para usuarios avanzados (aceleradores, customización…)								
Proporciona mapas y un rastro (Proporciona una manera de ver adonde ir, qué sucederá)								
Muestra al usuario qué es (o no es) posible								
Mapeos intuitivos (los elementos visuales se corresponden con la acción que realizan)								
Reducción al mínimo de la carga de memoria (Evitar la necesidad de recordar. Facilidad para comparar de manera sencilla)								

Consistencia y uso de estándares (El mismo término/acción se usa de forma consistente. Si no hay nada mejor, que sea conforme al estándar)	☐ ☐ ☐ ☐ ☐ ☐
Realimentación (Proporciona una adecuada realimentación sobre los procesos/estado del sistema)	☐ ☐ ☐ ☐ ☐ ☐
Prevención de errores (Hace que sea difícil cometer errores)	☐ ☐ ☐ ☐ ☐ ☐
Mensajes de error (Diagnostica correctamente la fuente y la causa de un problema y sugiere una solución)	☐ ☐ ☐ ☐ ☐ ☐
Indicaciones claras de salidas y recuperación de errores (El usuario puede salir fácilmente de un estado indeseado. La aplicación asume que el usuario cometerá errores y necesitará recuperar estados anteriores)	☐ ☐ ☐ ☐ ☐ ☐
Enumere los aspectos más negativos	1.- 2.- 3.- 4.- 5.-
Enumere los aspectos más positivos	1.- 2.- 3.- 4.- 5.-

Satisfacción con el Interfaz de usuario (Based on: **Chin, J.P., Diehl, V.A., Norman, K.L. (1988)** ***Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface.*** ACM CHI'88 Proceedings, **213-218. ©1988 ACM.**)

Impresión general de la aplicación

	1	2	3	4	5	6	7		N.A.
terrible								maravilloso	
difícil								fácil	
frustrante								satisfactorio	
potencia inadecuada								potencia adecuada	
poco estimulante								muy estimulante	
rígido								flexible	

Presentación en pantalla

		1	2	3	4	5	6	7		N.A.
Texto legible	difícil								fácil	
Organización de la información	confuso								muy claro	
Secuencia de pantallas	confuso								muy claro	

Información de la terminología y del sistema

		1	2	3	4	5	6	7		N.A.
Uso de términos en el sistema	inconstante								constante	
Posición de los mensajes	inconstante								constante	
Solicitud de datos al usuario	inconstante								constante	
Información sobre el progreso de las acciones	nunca								siempre	
Mensajes de error	inútiles								provechosos	

Aprendizaje

		1	2	3	4	5	6	7		N.A.
Aprendizaje del funcionamiento del sistema	difícil								fácil	
Exploración de nueva funcionalidad por prueba y error	difícil								fácil	
Recuerdo de nombres y uso de comandos	difícil								fácil	
La realización de tareas es directa	nunca								siempre	
Mensajes de ayuda	inútiles								provechosos	
Material de referencia suplementario	confuso								claro	

Capacidad del sistema

		1	2	3	4	5	6	7		N.A.
Velocidad del sistema	Muy lento								Muy rápido	
Confiablidad del sistema	No fiable								Fiable	
El sistema tiende a ser…	ruidoso								estable	
Corrección de errores del usuario	difícil								fácil	
Diseñado para todos los niveles de usuarios	nunca								siempre	

Preguntas abiertas

¿Para qué cree que puede servir esta herramienta?	
¿Cómo cree que esta herramienta puede ayudarle en su trabajo habitual?	
Comentarios adicionales	

Agradecimiento

Muchas gracias por el tiempo dedicado a rellenar este cuestionario. La información que ha aportado servirá para mejorar la integración entre estética y funcionalidad en aplicaciones web de nueva generación.

Atentamente,

-Mariano Rico

Apéndice B

Cuestionario de tecnologías web para profesionales

Apéndice C

Publicaciones derivadas de la tesis

Revistas internacionales

1. Mariano Rico, David Camacho, y Óscar Corcho. A Contribution-based Framework for the Creation of Semantically-enabled Web Applications. Aceptado con revisiones en *Information Sciences*, 2009.
   JCR (2007): índice de impacto= 2.15; posición (*Computer Science - Information Systems*)= 10/92.

2. Mariano Rico, José Antonio Macías, David Camacho, y Óscar Corcho. Enabling Web Designers to Handle Semantic Data. Enviado a *Journal of Web Semantics*, 2009.
   JCR (2007): índice de impacto= 3.41; posición (*Computer Science-Software Engineering*)= 2/84

3. Mariano Rico, José Antonio Macías y David Camacho. A User Study to Evaluate Quality Features in Wiki-Based Semantic Web Applications. Enviado a *Journal of Human-Computer Interaction*, 2009.
   JCR (2007): índice de impacto= 2.48; posición (*Computer Science-Theory & Methods*)= 4/79

4. Mariano Rico, Francisco García Sánchez, Juan M. Gómez, Rafael Valencia García y Jesualdo T. Fernández Breis. Enabling Intelligent Service Discovery with GGODO. Aceptado con revisiones en *Journal of Information Science and Engineering*, 2009.
   JCR (2007): índice de impacto= 0.20; posición (*Computer Science-Information Systems*)= 89/92.

Conferencias Internacionales

1. Mariano Rico, David Camacho, Óscar Corcho. Personalized Handling of Semantic Data with MIG. 8th International Workshop on Web Semantics (WebS'2009) at the 20th edition of DEXA, 2009. Enviado.

2. Mariano Rico, David Camacho, y Óscar Corcho. VPOET Templates to Handle the Presentation of Semantic Data Sources in Wikis. Fourth Workshop on Semantic Wikis: The Semantic Wiki Web (SemWiki2009) at the 6th Annual European Semantic Web Conference (ESWC), 2009.

3. Mariano Rico, David Camacho, y Óscar Corcho. Macros vs. scripting in VPOET. 5th Workshop on Scripting and Development for the Semantic Web (SFSW2009) at the 6th Annual European Semantic Web Conference (ESWC), 2009.

4. Mariano Rico, David Camacho, y Óscar Corcho. VPOET: Using a Distributed Collaborative Platform for Semantic Web Applications. Proc. 2nd International Symposium on Intelligent Distributed Computing (IDC'2008), Springer, Studies in Computational Intelligence (SCI), 162, páginas 167–176 , 2008.

5. Mariano Rico, José Antonio Macías. Tecnologías de la Web Semántica aplicadas a la creación y personalización de interfaces de usuario. Proc. VIII Congreso Internacional de Interacción Persona Ordenador. (Interacción 2007), Thomson, páginas 291–300, 2008.

6. Pablo Castells, Borja Foncillas, Rubén Lara, Mariano Rico, y Juan Luis Alonso. Semantic Web Technologies for Economic and Financial Information Management. Proc. *1st European Semantic Web Symposium (ESWS)*, Springer LNCS 3053, páginas 473–487, 2004

7. P. Castells, F. Perdrix, E. Pulido, M. Rico, R. Benjamins, J. Contreras, y J. Lorés. Neptuno: Semantic Web Technologies for a Digital Newspaper Archive. Proc. *1st European Semantic Web Symposium (ESWS)*. Springer LNCS 3053, páginas 445–458, 2004.

8. Mariano Rico y Pablo Castells. JACINTA: a Mediator Agent for Human-Agent Interaction in Semantic Web Services. Selected posters in *International Semantic Web Conference (ISWC)*, 2004. `http://iswc2004.semanticweb.org/posters/PID-IHZUQYLZ-1090177058.pdf`

Congresos nacionales

1. Mariano Rico y Pablo Castells. Jacinta, Primeros Pasos Hacia la Interacción Entre Personas y Servicios Web Semánticos. Libro de actas de *VI Congreso Internacional de Interacción Persona-Ordenador*. Ed. Thomson, 2005.

2. P. Castells, E. Pulido, C. Carranza, M. Rico, F. Perdrix, E. Piqué, J. Cal, R. Benjamins, J. Contreras, J. Lorés, y T. Granollers. Neptuno: Tecnologías de la Web Semántica para una Hemeroteca Digital. Libro de actas del *V Congreso en Interacción Persona-Ordenador*, páginas 306–313, Ed. Thomson, 2004.

Apéndice D

Glosario de términos

A lo largo de este documento se mencionan términos que son detallados en este apéndice.

D.1. Desarrollo por el usuario final

El Desarrollo por Usuarios Finales se define como "*un conjunto de métodos, técnicas, y herramientas, que permiten a los usuarios de sistemas software, quieres actúan como desarrolladores de software no profesionales, que de alguna manera creen, modifiquen, o extiendan un artefacto software*" [Lieberman *et al.*, 2006]. Esta área de investigación integra temas de IPO, Trabajo Cooperativo Soportado por Ordenadores (CSCW), Ingeniería del Software e Inteligencia Artificial. EUD se centra es los usuarios que actúan como desarrolladores, uniendo los perfiles de usuarios finales y desarrolladores. Para llevar a cabo esta tarea, los desarrolladores son responsables de diseñar aplicaciones de uso sencillo centradas en el usuario que ayuden a los usuarios finales a resolver tareas engorrosas de forma automática. JotSpot[1] es un buen ejemplo de esta tendencia aplicada a los wikis. Sin embargo, aunque JotSpot tiene capacidades muy interesantes, tales como formularios o almacenamiento XML de las páginas wiki, no está diseñado para manejar semántica. El número de aproximaciones semánticas basadas en la web relacionadas con este tema es muy bajo, ya que apenas ha sido tratado aún.

El trabajo de esta tesis se une a esta tendencia, pero con un objetivo menos ambicioso que proporcionar una infraestructura que permita crear aplicaciones web semánticas a los usuarios finales. Se sigue una aproximación orientada a usuarios finales, proporcionando a los diseñadores web (usuarios especializados, que se encuentran entre los usuarios finales y los expertos en tecnologías de la web semántica) un entorno amigable (macros y ciertas características de los wikis) para aplicar el poder creativo de estos usuarios a potenciar la Web Semántica.

D.2. Interfaces de Usuario basadas en modelos

Otro tema relacionado con nuestro trabajo son los Interfaces de Usuario basados en Modelos (MBUI), en los que se soportan distintos niveles de abstracción para modelar interfaces de usuario desde cero. Estos niveles conceptuales proporcionan descripciones del

[1] Véase `http://www.jot.com`

interfaz y permiten dividir responsabilidades entre programadores, diseñadores, y proveedores de contenido. Se han desarrollado muchas herramientas siguiendo este paradigma (e.g., MECANO [Puerta, 1996], MOBI-D [Puerta, 1997], TERESA [Mori *et al.*, 2004]) y también se han creado algunos lenguajes de especificación para modelar la descripción lógica de una interfaz de usuario (véase CTT [Paternò y Giammarino, 2006] o UsiXML [Limbourg *et al.*, 2004]). Para una buena revisión de este paradigma, ver [Florins, 2006].

Sin embargo, en la mayor parte de las aproximaciones MBUI existentes, apenas se usan las técnicas de la Web Semántica. Tan solo unas pocas, como PEGASUS [Castells y Macías, 2002] [Macias *et al.*, 2006], proporcionan características semánticas. PEGASUS es uno de los pocos sistemas que permiten una especificación MBUI usando un lenguaje semántico as hoc basado en RDF. Otra aproximación interesante dirigida a la creación de aplicaciones web usando las tecnologías de la Web Semántica es Semantic Hypermedia Design Method (SHDM) [Lima y Schwabe, 2003a]. En general, la mayor parte de estas aplicaciones están dirigidas a usuarios con un entrenamiento avanzado en el uso de estas herramientas. En cuando a las posibilidades de extensión de estas herramientas, requieren unos desarrolladores con unos conocimientos avanzados del diseño del sistema, por lo que la ampliación de estos sistemas es compleja.

D.3. Adaptación al usuario

Existen algunos estándares para especificar las características de la interacción a través de dispositivos. El W3C recomienda CCPP [Klyne *et al.*, 2004], una ontología RDFS sencilla. Usando esta especificación, se puede mejorar el paradigma MBUI usando nuestra aproximación semántica para tratar el problema de mapeo que pretende enlazar las presentaciones abstractas y finales de los elementos. Teniendo este objetivo en mente, este trabajo propone una ontología básica (utilizada por VPOET) a fin de proveer la información semántica necesaria para obtener la plantilla de visualización más adecuada para una situación dada.

Las preferencias del usuario, aplicadas a Servicios Web Semánticos se pueden encontrar en [Khushraj y Lassila, 2005], en el que se consideran aspectos como el contexto actual del usuario, el historial (uso y contexto), o datos corporativos, con el fin de crear interfaces de usuario para teléfonos móviles. La aplicación de las preferencias del usuario final para la creación de ontologías se trata en [Thomopoulos, 2005], donde se define el concepto de

viewpoint. El tema de la personalización puede ser revisado en [Blom y Monk, 2003].

Las interfaces adaptativas se caracterizan por su capacidad de adaptación al usuario considerando sus características físicas (e.g., daltonismo, o reducida capacidad visual). En [Florins, 2006] se revisa el estado del arte. No hay implementaciones que usen la técnicas de la Web Semántica, aunque este trabajo sigue estas ideas.

Digital Item Adaptation (DIA)[Vetro y Timmerer, 2005] es la parte-7 del estándar MPEG-21, y relaciona el contenido multimedia con el entorno de uso. Se proporcionan descripciones de los contenidos y del entorno de uso en XML. Estas descripciones están modeladas conforme a unos esquemas XML que consideran cuatro aspectos: capacidades de los terminales, características del usuario, red, y entorno natural. Aspectos de accesibilidad, como deficiencias visuales, son considerados en gran detalle (e.g. "LowVisionSymptoms" se compone de "LoosOfFineDetail", "LackOfContrast", "LightSensitivity", "NeedOfLight", "CenterVisionLoss", "PeripherialVisionLoss"), todos ellos con un valor numérico que indica el grado de deficiencia. Todos estos estándares y aproximaciones han sido usados como fuente de inspiración para crear perfiles de usuario.

D.4. Nuevas tendencias en interfaces web (Web 2.0)

El término Web 2.0 no se basa en nuevas tecnologías, sino en un cambio en la manera en la que las aplicaciones web manejan las las contribuciones de los usuarios. Herramientas tales como blogs, wikis, o podcasts, permiten que que los usuarios creen datos públicos de manera sencilla. La frase "es una actitud, no una tecnología" [2] define el espíritu de la Web 2.0 en términos de usuarios proporcionando información. A diferencia de la web tradicional, centrada en la publicación de información sin estructura, la Web2.0 se centra en la publicación de datos estructurados no semánticos, entre los que destacan XML y JSON.

Aunque se ha hecho hincapié en que no hay un fundamento tecnológico sobre el que descanse la Web 2.0, la cara visible de esta filosofía aprovechó una tecnología emergente: AJAX[3], acrónimo de "Asynchronous Javascript And XML". Estas mejoras dieron lugar a aplicaciones web más rápidas y amigables. Las ventajas más destacables de AJAX son las llamadas HTTP asíncronas y los modelos de documentos XML. Aunque la mayor parte

[2] Véase `http://iandavis.com/blog/2005/07/talis-web-20-and-all-that?year=2005&monthnum=07&name=talis-web-20-and-all-that`

[3] Véase `http://www.adaptivepath.com/publications/essays/archives/000385.php`

de estas nuevas capacidades aparecieron en 2000, no se extendió a los navegadores más comunes hasta 2002.

El proceso de popularización de AJAX empezó con la primera "Web 2.0 Conference" que tuvo lugar en octubre de 2004, y alcanzó su máximo siendo "Persona del año" y portada de la revista Time el 25 de septiembre de 2006, con el titular "*Persona del año: tú. Sí, tú. Tú controlas la era de la información. Bienvenido a tu mundo*".

Algunos toolkits relacionados con este trabajo han aparecido bajo esta nueva tendencia. Estos toolkits permiten a los desarrolladores la creación de widgets usando lenguajes de script (típicamente javascript) y proporcionan librerías compatibles con los navegadores más usados. Estos widgets pueden ser usados por los usuarios finales en sus páginas web o en los escritorios de sus ordenadores personales. Ejemplos de estos toolkits son Google Gadgets [4], Konfabulator [5] para MacOSX, o Yahoo Widgets [6]. Todos comparten una estrategia doble: (1) los desarrolladores consiguen simplificar el proceso de creación de nueva funcionalidad, y (2) los usuarios finales mejoran la forma en la que pueden usar esta nueva funcionalidad para crear nuevos contenidos. La aproximación de este trabajo sigue esta estrategia, encontrando un paralelismo entre widgets y plugins. Sin embargo, la aproximación seguida por este trabajo aprovecha las ventajas de la tecnologías de la Web Semántica para crear contenidos semánticos.

[4] Véase `http://www.google.com/apis/gadgets/index.html`
[5] Véase `http://www.widgetgallery.com/`
[6] Véase `http://widgets.yahoo.com/`

Referencias

Bibliografía

[Arpírez *et al.*, 2001] Julio C. Arpírez, Oscar Corcho, Mariano Fernández-López, y Asunción Gómez-Pérez. WebODE: a Scalable Workbench for Ontological Engineering. En *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, páginas 6–13, New York, NY, USA, 2001. ACM.

[Baker *et al.*, 2001] K. Baker, S. Greenberg, y C. Gutwin. Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration. *LECTURE NOTES IN COMPUTER SCIENCE*, 2254:123–140, 2001.

[Bechhofer *et al.*, 2004] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, y L.A. Stein. OWL Web Ontology Language Reference. technical report `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, 2004.

[Berners-Lee *et al.*, 2001] T. Berners-Lee, J. Hendler, y O. Lassila. The semantic web. *Scientific American*, 284(5):28–37, 2001.

[Berners-Lee *et al.*, 2006] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, y D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. En http://swui.semanticweb.org/swui06/program.html, editor, *Proceedings of The 3rd International Semantic Web User Interaction Workshop (SWUI). Athens, Georgia, USA*. collocated with ISWC 2006, November 2006.

[Bishop, 1995] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1995.

[Bizer *et al.*, 2006] Christian Bizer, Ryan Lee, y Emmanuel Pietriga. Fresnel: A Browser Independent Presentation Vocabulary for RDF. *Proceedings of the Second International Workshop on Interaction Design and the Semantic Web*, páginas 158–171, 2006.

[Blom y Monk, 2003] J. O. Blom y A. F. Monk. Theory of personalization of appearance: Why users personalize their pcs and mobile phones. *Human-Computer Interaction*, 18(3):193–228, 2003.

[Brickley y Guha, 2004] Dan Brickley y R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. technical report `http://www.w3.org/TR/rdf-schema/`, 2004.

[Brook, 1996] John Brook. *Usability Evaluation in Industry*, chapter SUS - A quick and dirty usability scale, páginas 189–194. Taylor & Francis; 1 edition (11 Jun 1996), 1996.

[Castells *et al.*, 2007] P. Castells, M. Fernández, y D. Vallet. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):261–272, 2007.

[Castells y Macías, 2002] Pablo Castells y José A. Macías. Context-sensitive user interface support for ontology-based web applications. En *Poster Session of the 1st. International Semantic Web Conference (ISWC). Sardinia, Italy.*, 2002.

[Cegarra Sánchez, 2004] José Cegarra Sánchez. *Metodología de la investigación científica y tecnológica*. 2004. ISBN: 9788479786243.

[Chin *et al.*, 1988] John P. Chin, Virginia A. Diehl, y Kent L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. En Elliot Soloway, Douglas Frye, y Sylvia B. Sheppard, editors, *Interface Evaluations. Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems*, páginas 213–218, 1988. June 15-19, 1988. Washington, DC, USA.

[Corcho *et al.*, 2006a] O. Corcho, A. López-Cima, y A. Gómez-Pérez. A platform for the development of semantic web portals. En *Proceedings of the 6th international conference on Web engineering (ICWE2006)*, páginas 145–152, New York, NY, USA, 2006. ACM Press.

[Corcho *et al.*, 2006b] Oscar Corcho, Angel López-Cima, y Asunción Gómez-Pérez. The odesew 2.0 semantic web application framework. En *WWW '06: Proceedings of the 15th international conference on World Wide Web*, páginas 1049–1050, New York, NY, USA, 2006. ACM.

[Cárdenas, 2005] José Luis Cárdenas. La investigación científica y el problema de su justificación en la discusión Boyle/Spinoza. *Ideas y Valores*, 54(128):33–60, Aug 2005.

[d'Aquin *et al.*, 2007] Mathieu d'Aquin, Claudio Baldassarre, Laurian Gridinoc, Sofia Angeletou, Marta Sabou, y Enrico Motta. Characterizing Knowledge on the Semantic Web with Watson. En *Proceedings of the 5th International Workshop on Evaluation of Ontologies and Ontology-Based Tools (EON2007), ISWC/ASWC*, volumen 329, páginas 1–10. CEUR Workshop Proceedings, 2007.

[DeLone y McLean, 1992] WH DeLone y ER McLean. Information systems success: The quest for the dependent variable. *Information & Management*, 3(1):60–95, 1992.

[Delone y McLean, 2003] W.H. Delone y E.R. McLean. The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems*, 19(4):9–30, 2003.

[Ding *et al.*, 2004] L. Ding, T. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, y J. Sachs. Swoogle: a search and metadata engine for the semantic web. *Proceedings of the Thirteenth ACM conference on Information and knowledge management*, páginas 652–659, 2004.

[Domingue, 1998] John Domingue. Tadzebao and webonto: Discussing, browsing, editing ontologies on the web. Knowledge Adquisition for Knowledge-Based Systems (KAW) Workshop. Banff, Canada., 1998.

[Duda *et al.*, 1973] R.O. Duda, P.E. Hart, et al. *Pattern classification and scene analysis*. Wiley New York, 1973.

[Fialho y Schwabe, 2007] André T. S. Fialho y Daniel Schwabe. Enriching hypermedia application interfaces. En *Proc. 7th International Conference, ICWE*, volumen 4607 de *LNCS*, páginas 188–193. Springer, 2007. 2007 Como, Italy, July 16-20, 2007.

[Finin y Ding, 2006] Tim Finin y Li Ding. Search engines for semantic web knowledge. En *Proceedings of XTech 2006: Building Web 2.0*, 2006. Amsterdam, 16-19 May 2006.

[Florins, 2006] Murielle Florins. *Graceful Degradation: a Method for Designing Multiplatform Graphical User Interfaces*. PhD thesis, Université catholique de Louvain, 2006.

[Fluit *et al.*, 2002] C. Fluit, F. van Harmelen, y M. Sabou. *Ontology-based information visualisation*, chapter 3, páginas 36–48. Springer Verlag, 2002.

[García *et al.*, 2008a] R. García, J.M. Gimeno, F. Perdrix, R. Gil, y M. Oliva. A Platform for Object-Action Semantic Web Interaction. En *Proceedings of the 16th international conference on Knowledge Engineering: Practice and Patterns*, páginas 404–418. Springer, 2008.

[García *et al.*, 2008b] R. García, F. Perdrix, R. Gil, y M. Oliva. The Semantic Web as a Newspaper Media Convergence Facilitator. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(2):151–161, 2008.

[García-Castro *et al.*, 2008] Raúl García-Castro, Óscar Muñoz-García, M. Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Stefania Costache, Diana Maynard, Stamatia Dasiopoulou, Raúl Palma, Vit Novacek, Freddy Lécué, Ying Ding, Monika Kaczmarek, Ruzica Piskac, Dominik Zyskowski, Jérôme Euzenat, Martin Dzbor, Lyndon Nixon, Alain Léger, Tomas Vitvar, Michal Zaremba, y Jens Hartmann. Architecture of the semantic web framework v2. Technical report, EU-IST Network of Excellence (NoE) FP6-507482 KWEB. Deliverable D1.2.5 (WP1.2), 2008.

[García y Gil, 2006] R. García y R. Gil. Improving Human–Semantic Web Interaction: The Rhizomer Experience. En *3rd Italian Semantic Web Workshop (SWAP'06)*, volumen 201, páginas 57–64. CEUR-WS, 2006.

[Gilb y Finzi, 1988] T. Gilb y S. Finzi. *Principles of software engineering management*. Addison-Wesley, 1988.

[Guha *et al.*, 2003] R. Guha, Rob McCool, y Eric Miller. Semantic Search. En *WWW '03: Proceedings of the 12th international conference on World Wide Web*, páginas 700–709, New York, NY, USA, 2003. ACM.

[Gulli y Signorini, 2005] A. Gulli y A. Signorini. The indexable web is more than 11.5 billion pages. En *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, páginas 902–903, New York, NY, USA, 2005. ACM Press.

[Heath *et al.*, 2006] T. Heath, J. Domingue, y P. Shabajee. User Interaction and Uptake Challenges to Successfully Deploying Semantic Web Technologies. En *Proceedings of*

the 3rd International Semantic Web User Interaction Workshop (SWUI2006), 5th International Semantic Web Conference (ISWC2006), 2006.

[Heflin y Hendler, 2001] Jeff Heflin y James Hendler. A portrait of the semantic web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[Holzinger, 2005] Andreas Holzinger. Usability engineering methods for software developers. *Communications of the ACM*, 48:71–74, 2005.

[Hornik, 1991] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[Horridge *et al.*, 2007] M. Horridge, S. Bechhofer, y O.Ñoppens. Igniting the OWL 1.1 Touch Paper: The OWL API. En *OWL: Experiences and Directions. Third International Workshop. Innsbruck, Austria*, volumen 258, June 2007.

[Houben *et al.*, 2004] Geert-Jan Houben, Flavius Frasincar, Peter Barna, y Richard Vdovjak. Modeling user input and hypermedia dynamics in hera. En Nora Koch, Piero Fraternali, y MartinWirsing, editors, *Proc. 4th International Conference, ICWE*, volumen 3140, páginas 60–73. Springer, 2004. 2004, Munich, Germany, July 26-30, 2004.

[Jabornig, 2006] Johannes Jabornig. Delivery context. a comparison and mapping model. Master's thesis, Universit¨at Klagenfurt. Fakult¨at f¨ur Wirtschaftswissenschaften und Informatik, Dic 2006.

[Kalyanpur *et al.*, 2004] A. Kalyanpur, D. Pastor, S. Battle, y J. Padget. Automatic mapping of OWL ontologies into Java. En *Proceedings of the International Conference on Software Engineering & Knowledge Engineering (SEKE)*, 2004.

[Karim y Tjoa, 2006] S. Karim y A. M. Tjoa. Towards the use of ontologies for improving user interaction for people with special needs. *LNCS. Computers Helping People With Special Needs, Proceedings*, 4061:77–84, 2006.

[Keller *et al.*, 2004] R. M. Keller, D. C. Berrios, R. E. Carvalho, D. R. Hall, S. J. Rich, I. B. Sturken, K. J. Swanson, y S. R. Wolfe. SemanticOrganizer: A customizable semantic repository for distributed NASA project teams. *International Semantic Web Conference (ISWC), LNCS Proceedings*, 3298:767–781, 2004.

[Khushraj y Lassila, 2005] D. Khushraj y O. Lassila. Ontological approach to generating personalized user interfaces for web services. *International Semantic Web Conference (ISWC), Proceedings*, 3729:916–927, 2005.

[Kiryakov *et al.*, 2004] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, y Damyan Ognyanoff. Semantic Annotation, Indexing, and Retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49 – 79, 2004.

[Klyne *et al.*, 2004] Graham Klyne, Franklin Reynolds, Chris Woodrow, Hidetaka Ohto, Johan Hjelm, Mark H. Butler, y Luu Tran. Composite capability/preference profiles (cc/pp): Structure and vocabularies 1.0. Technical report, W3C, 2004.

[Klyne y Carroll, 2004] G. Klyne y J.J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`. Technical report, W3C, 2004.

[Koechley, 2006] Nate Koechley. Graded Browser Support. http://developer.yahoo.com/yui/articles/gbs/gbs.html. Technical report, Yahoo! Inc., 2006. Senior Front-End Engineer.

[Lausen *et al.*, 2005a] Holger Lausen, Ying Ding, Michael Stollberg, Dieter Fensel, Ruben Lara, y Sung-Kook Han. Semantic web portals: state-of-the-art survey. *Journal of Knowledge Management*, 9(5):40–49, May 2005.

[Lausen *et al.*, 2005b] Holger Lausen, Axel Polleres, y Dumitru Roman. Web service modeling ontology (wsmo). Technical report, W3C Member Submission 3 June, 2005.

[Lieberman *et al.*, 2006] Henry Lieberman, Fabio Paternò, Volker Wulf, Volker Wulf, Alan F. Backwell, John F. Pane, y Brad A. Myers. *End-User Development (Human-Computer Interaction Series)*. Springer, 2006.

[Lima y Schwabe, 2003a] F. Lima y D. Schwabe. Modeling applications for the semantic web. En J.M. Cueva Lovelle, B.M. González Rodríguez, y L. Joyanes Aguilar, editors, *Proc. International Conference , ICWE 2003,*, volumen 2722 de *LNCS*, páginas 417–426. Springer, 2003. ICWE 2003, Oviedo, Spain, July 14-18, 2003.

[Lima y Schwabe, 2003b] Fernanda Lima y Daniel Schwabe. Application modeling for the semantic web. En *LA-Web 2003. Proceedings of the First Conference on Latin American Web Congress*, volumen 0, página 93, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[Limbourg *et al.*, 2004] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, y V. Lopez-Jaquero. UsiXML: a Language Supporting Multi-Path Development of User Interfaces. *LNCS. Proceedings of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS*, 3425/2005:200–220, 2004.

[Macias *et al.*, 2006] A. Macias, A. R. Puerta, y P. Castells. Model-based user interface reengineering. *HCI Related Papers of Interaction 2004*, páginas 155–162, 2006.

[Martin *et al.*, 2005] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan11, y Katia Sycara1. Bringing Semantics to Web Services: The OWL-S Approach. *LNCS. Semantic Web Services and Web Process Composition. Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, 3387/2005(3387/2005):26–42, 2005. J. Cardoso and A. Sheth (Eds.).

[Morbidoni *et al.*, 2008] Christian Morbidoni, Danh Le Phuoc, Axel Polleres, y Giovanni Tummarello. Previewing semantic web pipes. En Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, y Manolis Koubarakis, editors, *Proceedings of the 5th European Semantic Web Conference (ESWC2008)*, number 5021 in LNCS, páginas 843–848. Springer, 2008. Tenerife, Canary Islands, Spain, June 1-5, 2008.

[Mori *et al.*, 2004] G. Mori, F Paterno, y C. Santoro. Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. *IEEE Trans. Software Eng*, 30(8):507–520, 2004.

[Motta y Sabou, 2006] Enrico Motta y Marta Sabou. Next generation semantic web applications. En Riichiro Mizoguchi, Zhongzhi Shi, y Fausto Giunchiglia, editors, *Proceedings of the First Asian Semantic Web Conference*, volumen 4185 de *LNCS*, páginas 24–29. Springer, 2006. Beijing, China, September 3-7, 2006.

[Muñoz *et al.*, 2007] Sergio Muñoz, Jorge Pérez, y Claudio Gutierrez. Minimal deductive systems for rdf. En *The Semantic Web: Research and Applications. 4th European Semantic Web Conference, ESWC 2007.*, volumen 4519, páginas 53–67. Springer, 2007.

[Nardi, 1993] B.A. Nardi. *A Small Matter of Programming: Perspectives on End User Computing*. MIT Press, 1993.

[Nielsen y Mack, 1994] Jakob Nielsen y Robert L. Mack. *Usability Inspection Methods*, chapter 2: Heuristic Evaluation. Wiley & Sons, Inc., 1994.

[Noy *et al.*, 2001] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, y M. A. Musen. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems & Their Applications*, 16(2):60–71, 2001.

[Nunes y Schwabe, 2006] Demetrius A. Nunes y Daniel Schwabe. Rapid prototyping of web applications combining domain specific languages and model driven design. En *ICWE '06: Proceedings of the 6th international conference on Web engineering*, páginas 153–160, New York, NY, USA, 2006. ACM.

[Oren *et al.*, 2006] E. Oren, R. Delbru, K. Moller, M. Volkel, y S. Handschuh. Annotation and navigation in semantic wikis. En *ESWC Workshop on Semantic Wikis*, 2006.

[Oren *et al.*, 2007a] E. Oren, A. Haller, C. Mesnage, M. Hauswirth, B. Heitmann, y S. Decker. A Flexible Integration Framework for Semantic Web 2.0 Applications. *IEEE Software*, 24(5):64–71, Sept-Oct 2007.

[Oren *et al.*, 2007b] Eyal Oren, Renaud Delbru, Sebastian Gerke, Armin Haller, y Stefan Decker. ActiveRDF: Object-Oriented Semantic Web Programming. En *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, páginas 817–824, New York, NY, USA, 2007. ACM.

[Parsia, 2008] Bijan Parsia. Literate, active owl ontologies. En *Proc. International Workshop OWLED*, 2008. October 26-27, 2008.

[Paternò y Giammarino, 2006] F. Paternò y F. Giammarino. Authoring Interfaces with Combined Use of Graphics and Voice for Both Stationary and Mobile Devices. *Proceedings of the Working Conference on Advanced Visual Interfaces*, páginas 329–335, 2006.

[Pelleg y Moore, 2000] D. Pelleg y A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. *Proceedings of the 17th International Conf. on Machine Learning*, páginas 727–734, 2000.

[Perlman, 1997] Gary Perlman. Practical usability evaluation. En *CHI '97: CHI '97 extended abstracts on Human factors in computing systems*, páginas 168–169, New York, NY, USA, 1997. ACM. Los Angeles, USA. April 18-23.

[Pietriga *et al.*, 2006] Emmanuel Pietriga, Christian Bizer, David Karger, y Ryan Lee. Fresnel: A Browser-Independent Presentation Vocabulary for RDF. En *Proc. ISWC*, volumen 4273 de *LNCS*, páginas 158–171. Springer, 2006.

[Pietriga, 2001] Emmanuel Pietriga. Isaviz: A visual authoring tool for rdf. see http://www.w3.org/2001/11/isaviz/, 2001.

[Prud'hommeaux y Seaborne, 2008] Eric Prud'hommeaux y Andy Seaborne. Sparql query language for rdf. Technical report, W3C Recommendation, 2008.

[Puerta, 1996] A.R. Puerta. The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development. *Proc. of CADUI96: Computer-Aided Design of User Interfaces*, 1996.

[Puerta, 1997] Angel R. Puerta. A model-based interface development environment. *IEEE Software*, 14(4):40–47, 1997.

[Ricci y Schwabe, 2006] Luiz A. Ricci y Daniel Schwabe. An authoring environment for model-driven web applications. En *WebMedia '06: Proceedings of the 12th Brazilian symposium on Multimedia and the web*, páginas 11–19, New York, NY, USA, 2006. ACM.

[Rico *et al.*, 2009] Mariano Rico, Francisco García-Sánchez, Juan Miguel Gómez, Rafael Valencia-García, y Rodrigo Martínez-Béjar. Enabling Intelligent Service Discovery with GGODO. *Journal of Database Management. Special Issue on Service-oriented Computing. Accepted with revisions.*, 2009.

[Rochen *et al.*, 2006] R. Rochen, M.B. Rosson, y M.A. Pérez. *End user Development of Web Applications*, chapter 8, páginas 161–182. Springer, 2006.

[Schraefel *et al.*, 2008] M.C. Schraefel, Jennifer Golbeck, Duane Degler, Abraham Bernstein, y Lloyd Rutledge. Semantic Web User Interactions: Exploring HCI Challenges. En *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, páginas 3929–3932, New York, NY, USA, 2008. ACM. Florence, Italy April 05 - 10, 2008.

[Shneiderman y Plaisant, 2005] Ben Shneiderman y Catherine Plaisant. *Designing the user interface:strategies for effective Human-Computer interaction*. Adison-Wesley, 2005.

[Staab *et al.*, 2000] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H. P. Schnurr, R. Studer, y Y. Sure. Semantic community web portals. *Computer Networks*, 33(1-6):473 – 491, 2000.

[Studer *et al.*, 1998] R. Studer, V.R. Benjamins, y D. Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161–197, 1998.

[Sure *et al.*, 2002] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, y D. Wenke. Ontoedit: Collaborative ontology development for the semantic web. *LNCS. Proceedings of International Semantic Web Conference (ISWC)*, 2342:221–235, 2002.

[SVG, 2003] SVG. Scalable vector graphics. w3c recommendation. http://www.w3.org/graphics/svg/, 2003.

[Thomopoulos, 2005] R. Thomopoulos. Expressing preferences in a viewpoint ontology. *On the Move to Meaningful Internet Systems: COOPIS, DOA, and ODBASE, PT 2. Proceedings*, 3761:1596–1604, 2005.

[Vdovjak *et al.*, 2003] R. Vdovjak, F. Frasincar, G.J. Houben, y P. Barna. Engineering Semantic Web Information Systems in Hera. *Journal of Web Engineering*, 2:3–26, 2003.

[Vetro y Timmerer, 2005] A. Vetro y C. Timmerer. Digital item adaptation: Overview of standardization and research activities. *IEEE Transactions On Multimedia*, 7(3):418–426, junio 2005.

[Völkel, 2006] Max Völkel. Rdfreactor – from ontologies to programatic data access. En *In Proc. of the Jena User Conference*, 2006. HP Bristol, May 2006.

[Walker y Walker, 1963] M.J. Walker y M. Walker. *The nature of scientific thought*. Prentice-Hall, 1963.

[Warren y Baker, 2005] Robert H. Warren y Christopher O. J. Baker. Ontologies: Where are we at? En *Knowledge-Based Bioinformatics Workshop*, Montreal, Quebec, Canada, September 2005. Poster.

[Wooldridge y Jennings, 1995] M Wooldridge y N.R. Jennings. Intelligent agents theory and practice. *Knowledge Engineering Review*, 10(2):115–152, junio 1995.