Aplicaciones dinámicas de Internet

Un nuevo enfoque para su desarrollo en educación.

Autor: Francisco A. Lizarralde

Director: Prof. Lic. Luis Mariano Bibbó.

TRABAJO FINAL INTEGRADOR PARA OPTAR POR EL TÍTULO DE
ESPECIALISTA EN TECNOLOGÍA INFORMÁTICA APLICADA EN EDUCACIÓN

FACULTAD DE INFORMÁTICA

UNIVERSIDAD NACIONAL DE LA PLATA

Diciembre de 2008

Objetivo

Esta monografía analiza las características de las aplicaciones dinámicas de Internet y presenta un nuevo enfoque para su desarrollo. Este tipo de aplicaciones, más conocidas como *RIA*, por sus siglas en inglés *(Rich Internet Applications)* presenta un mayor nivel de expresividad y de interactividad con los contenidos, lo que las hace particularmente atractivas para su aplicación en ámbitos educativos.

Este nuevo enfoque está basado en *Seaside*, un entorno de desarrollo para construir aplicaciones dinámicas de Internet, que posean una lógica interna relativamente compleja. *Seaside* provee una arquitectura basada en componentes, cuya característica distintiva es, que permite modelar múltiples flujos de información independientes, lo que simplifica notablemente la implementación de modelos complejos en Internet.

Prefacio

El desarrollo tecnológico producido en los últimos años en las aplicaciones de Internet, ha ampliado la variedad de contenidos e incorporado nuevos medios para su difusión. Todos estos avances, ampliamente utilizados en aplicaciones comerciales, recién están siendo incorporados al ámbito educativo. Sin embargo, la incorporación de multimedia, simulaciones, exploraciones, juegos, etc., en aplicaciones educativas de Internet, requiere de un modelo capaz de resolver situaciones más complejas, que las presentes en el modelo de interacción web tradicional.

Seaside se propone como un nuevo paradigma para el desarrollo de aplicaciones complejas de Internet, ya que presenta características que lo diferencian notablemente del modelo tradicional. En el modelo tradicional, el servidor web responde al cliente, con una nueva página a cada petición realizada. Mientras que Seaside, permite el manejo simultáneo de múltiples flujos de información dentro de la misma página, y solo recurre al servidor cuando la información requerida no se encuentra presente en el cliente. Este intercambio simultáneo de información, permite crear aplicaciones mucho más interactivas, ya que es posible actualizar solo parte de la información contenida en la página web y muchas veces, en forma totalmente local.

La organización de esta presentación es la siguiente:

- En el capítulo 1 se describirá el modelo clásico de interacción de las aplicaciones de Internet. Se presentarán las caracteríticas de las aplicaciones dinámicas de Internet *(RIA)*, se mostrará su ubicación dentro de un esquema de desarrollo tecnológico y se describirán sus principales ventajas.

- En el capítulo 2 se presentarán las principales tecnologías utilizadas para construir aplicaciones dinámicas de Internet. Y se describirá el modelo *AJAX*, que posibilita una mayor interactividad del usuario con las aplicaciones web.

- En el capítulo 3 se presentarán las características de los distintos tipos de aplicaciones educativas, en referencia a los diferentes modelos instructivos y teorías del aprendizaje. Describiéndose además, las ventajas de las aplicaciones de Internet, en el ámbito educativo.

- En el capítulo 4 se hará un recorrido sobre los orígenes y evolución de *Squeak*, el lenguaje con el que se construyó *Seaside*. Se presentarán además, los conceptos principales en los que se basó su predecesor, Smalltalk-80, un lenguaje desarrollado especialmente con fines educativos.

- En el capítulo 5 se hará una breve reseña del método *DistSem* [30], para el análisis de estimaciones de proximidad semántica entre conceptos. Precisamente, la implementación de una aplicación para la captura de datos correspondiente al método *DistSem* será presentada como un ejemplo de utilización de *Seaside* para el desarrollo de aplicaciones web con características altamente interactivas.

- En el capítulo 6 se utilizará el ejemplo antes mencionado para analizar con mayor detalle, las características principales de *Seaside*. Estas características, incluyen la posibilidad de integrar recursos tales como hojas de estilo (*CSS*) y controles interactivos (*Scriptaculous*). Y se hará especial mención acerca de una de las principales ventajas de *Seaside* sobre otros entornos de desarrollo, es decir, la posibilidad de modificar la aplicación misma, en plena ejecución y en forma totalmente remota.

- En el capítulo 7 se presentarán las conclusiones y los planes de trabajo a seguir.

Índice general

Capítulo 1

Aplicaciones de Internet.

Mucho se ha evolucionado desde los comienzos de la World Wide Web, cuando la mayoría de la información estaba contenida en páginas web interconectadas por vínculos estáticos. [3] Hasta hace poco más de una década, la posibilidad de interacción con la información contenida en una página web, estaba limitada a su lectura y, en algunos casos, al acceso a cierta información complementaria, por medio de un enlace a una dirección URL *(Uniform Resource Locator)*.

Poco tiempo después, comenzaron a utilizarse pequeños programas instalados en el servidor web, denominados CGI-Scripts *(Common Gateway Interface-Scripts)* [20]. A partir de ese momento, se logró que los usuarios mantuvieran cierta interacción con una página web, como por ejemplo, el ingreso de datos en un formulario y su posterior envío al servidor web para el procesamiento de los mismos. A este tipo de aplicaciones de Internet, las denominaremos, aplicaciones clásicas de Internet.

1.1. Aplicaciones clásicas de Internet.

Inicialmente, las aplicaciones de Internet estaban constituidas básicamente por páginas web escritas en *HTML (HyperText Markup Language)*. Este lenguaje, utilizado originalmente para escribir páginas web estáticas, utiliza ciertas marcas *(tags)*, para describir la estructura y su contenido en forma textual, e incluso complementar el texto con contenidos diversos, como imágenes, sonidos, etc.

Estas páginas web se relacionan a su vez con otras, por medio de enlaces. Creando así, las bases de una estructura hipertextual de navegación. Es decir, conforman un conjunto de información textual, gráfica y sonora, enlazada entre sí, o con otras informaciones relacionadas. Estos enlaces, per-

miten un recorrido de la información en una forma no siempre prefijada ni secuencial. Este modelo representa una arquitectura relativamente simple, con un conjunto limitado de opciones de desarrollo, lo que facilita su diseño y mantenimiento.

En este modelo, cada interacción con la interfaz del usuario es transformada en una petición *HTTP (HyperText Transfer Protocol)*, la cual es enviada al servidor web. De esta forma, el servidor web, no sólo debe atender a los requerimientos relacionados con información proveniente de sus bases de datos, sino que también debe hacerse cargo de la actualización de la página misma, enviando al navegador el código *HTTP* y *CSS (Cascade Style Sheets)* necesarios para dicha tarea.

En la figura 1.1 se aprecia el modelo arquitectónico de una aplicación de Internet clásica.[10]

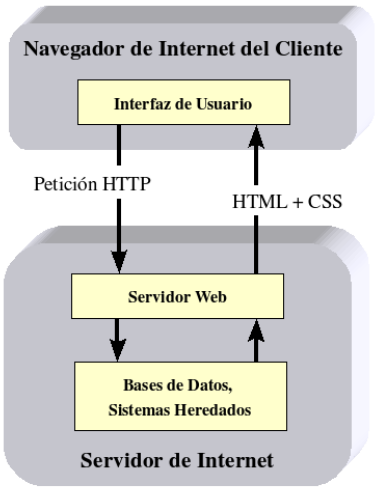

Figura 1.1: Modelo de una aplicación Web clásica.

En la misma, se aprecia como el servidor es quien debe atender en forma remota a cada requerimiento de los usuarios, ya sea por solicitud de información ó por actualización de la interfaz. Esto produce una sobrecarga de tareas en el servidor, lo que reduce el nivel de interactividad de las aplicaciones. Como veremos a continuación, las aplicaciones dinámicas de Internet modifican este sencillo modelo para incrementar el grado de interacción con el usuario.

Sin embargo, independientemente del nivel de interactividad logrado, una de las mayores ventajas de las aplicaciones web, comparadas con las aplicaciones *cliente-servidor* convencionales, es que el usuario no necesita instalar

un programa especial para cada aplicación que desee utilizar. Solo necesita tener instalado un navegador y una conexión a Internet, lo que simplifica notablemente su distribución y actualización, ya que al conectarse al servidor siempre contará con la última versión disponible, instalada y funcionando.

1.2. Aplicaciones dinámicas de Internet. *(RIA)*

Las aplicaciones dinámicas de Internet [1], *(RIA)* surgen como una respuesta a los nuevos requerimientos de los usuarios de la web. Intentar definirlas puede resultar tan ambiguo como el término **Web 2.0** acuñado por *Tim O'Reilly* [22], para describir la nueva generación de recursos tecnológicos disponibles en Internet y su impacto social. Por este motivo, nos concentraremos en describir sus características y las tecnologías asociadas a las mismas, mencionando como ejemplo, algunas de las implementaciones más conocidas.

En general, puede decirse que las aplicaciones dinámicas de Internet comparten una gran cantidad de características con las aplicaciones de escritorio, en cuanto a recursos disponibles y nivel de interactividad. Principalmente esto es lo que percibe el usuario cuando las compara con los sitios web tradicionales. Para dar una idea de la diferencia, consideremos la comparación entre un sitio web que permite acceder a un conjunto de figuras con mapas estáticos, con una aplicación como *Google Maps* que le permite a los usuarios interactuar directamente con el mapa, desplazándolo, agrandando una zona ó buscando una localización en particular, en tiempo real.

1.2.1. Categorización de las RIA

Las aplicaciones dinámicas de Internet (*RIA*) cubren un amplio espectro, y por lo general se diferencian por la magnitud de la actividad delegada, ya sea del lado del servidor como del lado del cliente. Es decir, se podría considerar que, en un extremo se hallan las aplicaciones exclusivamente de escritorio, en las cuales todo el procesamiento se realiza en forma local y, en el otro extremo, se encuentran las aplicaciones web tradicionales, donde todo el procesamiento se realiza en el servidor del sitio web.

Si ubicamos en el eje horizontal a los diferentes tipos de aplicaciones, desde las aplicaciones de escritorio hasta las aplicaciones web tradicionales y en el eje vertical representamos el nivel de procesamiento, ya sea del lado del

[1] Este tipo de aplicaciones son internacionalmente conocidas como RIA, por sus siglas en inglés, *(Rich Internet Applications)*, por lo que en el presente trabajo, utilizaremos frecuentemente esta denominación.

cliente como del servidor, vemos claramente cuál es el lugar que ocupan las aplicaciones dinámicas de Internet.

En la figura 1.2, se observa un área sombreada, cuyo extremo superior izquierdo está representado las aplicaciones con gran nivel de procesamiento del lado del cliente (Cliente Pesado) y en el extremo inferior derecho se hallan las aplicaciones con gran nivel de procesamiento del lado del servidor (Cliente Delgado). Estos puntos permiten determinar un área rectangular en la que se encuentran enmarcados los diferentes tipos de aplicaciones dinámicas de Internet (*RIA*). [21]

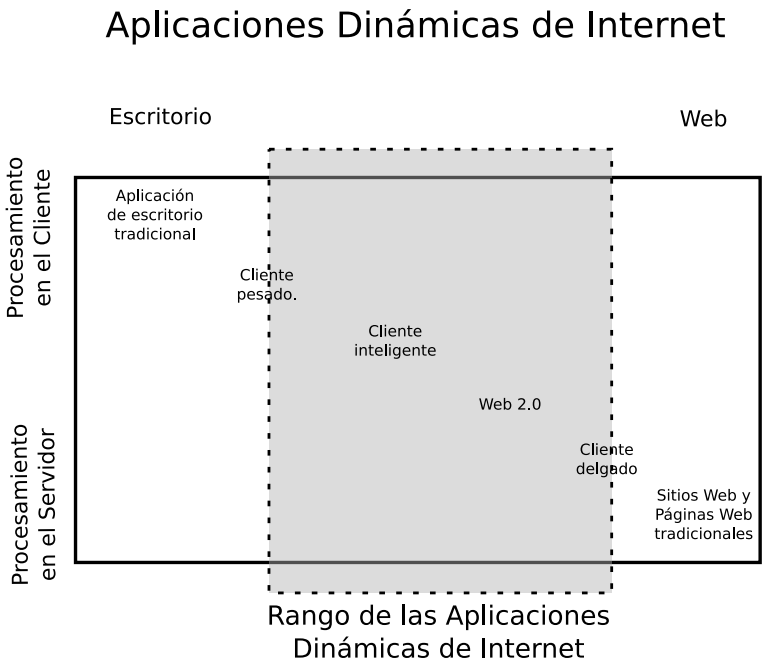

Figura 1.2: Categorización de las RIA

Sin embargo, a pesar de las diferencias de nivel de procesamiento, ya sea en el cliente ó en el servidor, la mayoría de las aplicaciones dinámicas de Internet comparte una característica principal, la complejidad del modelo subyacente es superior al de las aplicaciones tradicionales de Internet.

1.3. Características principales.

En un comienzo, el modelo utilizado por la mayoría de las aplicaciones de Internet era muy similar al de las instalaciones centralizadas. En las cuales un computador central *(Mainframe)* era el responsable de todo el procesamiento

y un conjunto de terminales conectadas al mismo, permitían a los usuarios interactuar con las aplicaciones.

La principal desventaja de este modelo es que toda la interacción con la aplicación, es procesada íntegramente en el servidor. Esto significa que, cada interacción del usuario con la aplicación se traduce en una petición al servidor, la cual es procesada por el mismo y posteriormente, la respuesta es enviada nuevamente al cliente, lo que generalmente produce la actualización de toda la página web, por parte del navegador de Internet.

Una de las principales características de las *(RIA)*, es la delegación de una parte del procesamiento en el cliente, sobre todo, lo referente a la interacción directa con el usuario. Esta distribución de las responsabilidades entre cliente y servidor presenta las siguientes ventajas:

- **Mayor interactividad.**
  Al delegarse parte del procesamiento en el cliente, el usuario suele percibir una respuesta más inmediata al interactuar con la aplicación, que la obtenida con las aplicaciones web tradicionales, que realizan todo el procesamiento en el servidor.

- **Actualización permanente.**
  Parte del código de la aplicación se transfiere directamente al navegador al arrancar la misma y suele actualizarse frecuentemente a medida que la misma se está ejecutando, por esta razón no es necesaria ninguna actualización de versiones.

- **Ejecución sin instalación previa.**
  El código de la aplicación se va descargando de Internet mientras la misma se ejecuta, por lo tanto, no es necesaria una instalación previa de la aplicación en la computadora del usuario.

- **Distribución y ejecución multiplataforma.**
  Lo único que necesita el usuario para utilizarlas es un navegador de Internet. El código de las *RIA* es interpretado por el propio navegador, lo que lo hace independiente del sistema operativo que utilice el usuario. Por lo tanto, no es necesario contar con diferentes versiones para cada sistema operativo.

Por supuesto que estas ventajas tienen un costo asociado. Para que las aplicaciones dinámicas de Internet no necesiten instalación previa, el código de las mismas debe transferirse en algún momento desde el servidor hasta el navegador del usuario. Y si bien es posible utilizar técnicas de compresión, de transferencia incremental y de almacenamiento intermedio para minimizar el

tiempo de transferencia, puede existir cierta latencia que dependerá diversos factores, como la velocidad de la conexión y la disponibilidad de atención que presente el servidor.

Por otra parte, para independizar a las aplicaciones del sistema operativo del usuario, se delega en el navegador la ejecución de código interpretado, usualmente *JavaScript*[2]. La utilización de código interpretado brinda una gran flexibilidad, permitiendo incluso su modificación en tiempo de ejecución, como veremos más adelante. Si bien estos lenguajes interpretados pueden presentar una velocidad de procesamiento inferior a la de los lenguajes compilados, esto es cada vez menos perceptible por el usuario, ya que las velocidades de comunicación y procesamiento son grandes y continúan en franco crecimiento.

[2]JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web.

Capítulo 2

Tecnologías Aplicadas

Actualmente existe un conjunto de tecnologías utilizadas para construir aplicaciones dinámicas de Internet. Una de las más populares es *AJAX (Asynchronous JavaScript And XML)*, la cual está presente en *Google Maps*, *Google Docs* y *GMail*, y en aplicaciones de Internet como *Flickr*[1] ó *Mint*[2]. Otra tecnología muy difundida es *Flash*® de Macromedia, que aporta detalles estéticos de gran calidad, como puede apreciarse particularmente en las gráficas de cotización de acciones presentes en *Yahoo*, ó en las utilizadas para las elecciones de Estados Unidos en 2008 por la *CNN*. A éstas se les suman, productos comerciales para desarrollar *RIAs* como *Flex*® y *Air*® de Adobe, *Silverlight*® de Microsoft y alternativas de código abierto como *OpenLaszlo*^{TM} de Lazlo Systems y *JavaFX*^{TM} de Sun MicroSystems. .

2.1. Incluidas en el Navegador

Inicialmente las páginas web sólo incluían información sobre sus contenidos de texto, imágenes y enlaces a otras páginas relacionadas. Sin embargo, al desarrollarse el estándar HTML (*HyperText Markup Language*), se incluyó además, información sobre la presentación de los contenidos, dentro de la misma página. Posteriormente, se incorporaron más funcionalidades a las páginas web, lo que hizo necesario que los navegadores fueran capaces de interpretar, además del código HTML, la información contenida en otros formatos, como CSS, XML, SVG, PDF, etc., que forman parte de los estándares abiertos establecidos por el W3C (*World Wide Web Consortium*). [3]

[1] Almacenamiento de Imágenes y Videos

[2] Análisis estadístico de sitios de Internet

[3] El World Wide Web Consortium es la principal organización que establece los estándares de la World Wide Web.

El W3C fue fundado por *Sir Tim Berners Lee* [4], luego de abandonar su trabajo en el CERN [5] en Octubre de 1994. Y su principal tarea es asegurar la compatibilidad y el acuerdo de la industria en la adopción de nuevos estándares. Antes de su creación, existían distintas versiones de HTML creadas por diferentes empresas, lo que generaba inconsistencias en el desarrollo de sitios web. Esto creaba innumerables problemas a los usuarios, quienes percibían que la apariencia y el funcionamiento de las páginas de un sitio web era diferente, y que este comportamiento, muchas veces dependía del navegador utilizado.

2.2. Módulos adicionales

Otro grupo de tecnologías que aportan interactividad y recursos a las aplicaciones de Internet son los módulos adicionales *(Plug-in)*[6]. Si un usuario intenta abrir un archivo y su navegador carece del correspondiente plugin, el mismo navegador se encargará de avisarle y le brindará un enlace para obtenerlo. Estos plug-ins se integran al navegador permitiéndole acceder a cierto tipo especial de contenidos, como es el caso de los plug-ins de *Real Player*® ó *QuickTime*®, que permiten acceder a contenidos de audio y video. Los plug-ins típicos tienen la función de reproducir determinados formatos de gráficos, visualizar datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos, etc.

Otro tipo de plug-ins permiten la ejecución de animaciones como es el caso de *Flash*® (Adobe) ó de pequeñas aplicaciones denominadas *Java Applets*® (Sun Microsystems). Por medio de estas mini-aplicaciones, es posible realizar cálculos, simulaciones, tutoriales interactivos, etc. Aunque algunas de estas alternativas presentan ciertas limitaciones, como el acceso a ciertos recursos del usuario, por razones de seguridad y la dificultad que encuentran los buscadores automáticos de Internet (webBots), para incluir a estos sitios en sus bases de datos.

[4] Es el creador del primer navegador de Internet y el autor principal de las especificaciones originales de URL, HTTP y HTML, las cuales conforman las bases de la World Wide Web.

[5] Consejo Europeo para la Investigación Nuclear (Conseil Européen pour la Recherche Nucléaire)

[6] En inglés, *plug-in* significa enchufar y en informática este término está referido a un tipo de aplicación que interactúa con otra para aportarle nueva funcionalidad, por lo general, muy específica.

2.3. Tecnologías asincrónicas

El término AJAX *(Asynchronous JavaScript And XML)*, fue introducido por primera vez por Jesse James Garrett en Febrero de 2005 [10], para denominar a un conjunto de técnicas inter-relacionadas, tales como XHTML, CSS, DOM, XML, HTTP y JavaScript, comunmente utilizadas para crear aplicaciones web interactivas.

La utilización de AJAX para la creación de aplicaciones web, permite lograr un mayor grado de interactividad, ya que disminuye la necesidad de "refrescar" toda la información contenida en la página web. Esto reduce notablemente la cantidad de información transmitida desde y hacia el servidor, con el consiguiente incremento de velocidad en las respuestas.

En la figura 2.1 se puede apreciar el modelo que representa la interacción asincrónica del cliente con el servidor, por medio de AJAX. A diferencia del modelo clásico, vemos que aparece una nueva capa, representada por el *motor AJAX*, que permite realizar una parte del procesamiento directamente del lado del cliente.

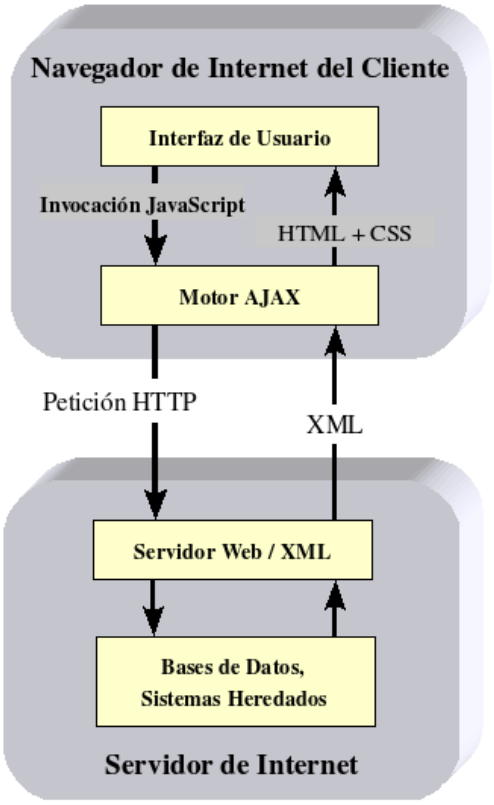

Figura 2.1: Modelo de una aplicación Web AJAX.

Por lo general, este procesamiento suele estar relacionado con la interfaz de usuario. Es decir, todas aquellas peticiones que no requieren del acceso a datos almacenados en el servidor, son resueltas directamente por el motor AJAX.

Un mecanismo de interacción similar fue propuesto por Netscape en 2003. En un documento llamado "Inner Browsing"[9], planteó la posibilidad de realizar toda la navegación del sitio en una sola página web. Esta página se actualizaría por partes, según las necesidades del usuario, quebrando así, con el paradigma de navegación página por página de los sitios web tradicionales. Este cambio creó un nuevo paradigma, el de las aplicaciones dinámicas de Internet. En este tipo de aplicaciones, la interfaz de usuario, que puede presentar el mismo aspecto que una aplicación de escritorio, no se actualiza por completo, sino sólo en aquellas partes que han modificado su contenido.

Actualmente, este conjunto de tecnologías agrupadas bajo el nombre de AJAX, permite que las aplicaciones web posean recursos, que hasta hace poco tiempo, sólo estaban disponibles en las aplicaciones de escritorio.

A continuación se presentan las principales características y lenguajes presentes en AJAX:

- **Asincrónico.** Los datos adicionales solicitados al servidor, se transfieren por medio de procesos que se ejecutan en segundo plano (background), para no interferir con la visualización de la página ni con el comportamiento de la aplicación.
- **JavaScript.** Es un lenguaje interpretado que se ejecuta en el navegador. Gran parte de los recursos, hoy disponibles, en las aplicaciones dinámicas de Internet, suelen estar programados en este lenguaje.
- **XML.** Sus siglas significan lenguaje de marcas extensible *(**Ex**tensible **M**arkup **L**anguage)*. Este lenguaje se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Y su utilización no se halla restringida a Internet, sino que es ampliamente utilizado en bases de datos, editores de texto, hojas de cálculo, etc.

Por otra parte **AJAX** utiliza normas abiertas como **DOM** *(Document Object Model)*, lo que permite que funcione en diferentes sistemas operativos, diferentes arquitecturas de computadoras y diferentes navegadores.

En resúmen, en las aplicaciones de Internet tradicionales, la interacción con los contenidos no es contínua, ya que presenta los lógicos saltos entre páginas. Esto se debe a que, las acciones del usuario se transforman en peticiones al servidor y es necesario esperar a que éste las responda, para continuar.

En cambio en las aplicaciones dinámicas de Internet, el navegador descarga inicialmente el motor AJAX, el cual posibilita cierta interacción con el usuario en forma local. De esta forma, gran parte de las modificaciones de la interfaz, debidas a la interacción con el usuario, son resueltas directamente por el motor AJAX. Sólo aquellas peticiones que no pueden resolverse localmente son enviadas al servidor. Estos procesos trabajan en forma independiente y en paralelo, lo que redunda en una mejora del tiempo total de respuesta. De esta forma, el usuario percibe sólo la interacción con la aplicación, mientras la carga de datos desde el servidor se realiza por medio de un proceso que corre simultáneamente, en segundo plano.

Capítulo 3

Aplicaciones Educativas.

La utilización de la informática en el ámbito educativo no es un asunto nuevo, sin embargo, aún se está lejos de aprovechar todo su potencial. Es mucho lo que se ha debatido acerca de las nuevas tecnologías y de la integración de las computadoras en los procesos de aprendizaje. Sin embargo, su presencia en el aula aún es poco frecuente, ya que por lo general, aún están reservadas para un espacio diferente, la "sala de computadoras".

En su obra "El ordenador invisible", Gros Salvat [1] sostiene la idea que la computadora pasará a integrarse completamente a los procesos educativos cuando ya no se hable de ella, cuando sea algo invisible, cuando al entrar en el aula observemos con la misma naturalidad a las computadoras, que a los lápices, los libros o los cuadernos. [13]

Mientras las computadoras continúen utilizándose en lugares especiales, con acceso y horarios restringidos y con una escasa relación de equipos por alumno, será bastante difícil lograr una integración total de las mismas en el proceso de enseñanza-aprendizaje. Afortunadamente, esta integración puede facilitarse con la incorporación al ámbito educativo, de computadoras más accesibles. Accesibles, no sólo desde el punto de vista económico, sino también desde otros aspectos. Nos estamos refiriendo a computadoras más compactas, más robustas, con menos requerimientos de energía y fundamentalmente con gran capacidad de conectividad y fácil acceso a una amplia biblioteca de programas educativos.

En tal sentido, un proyecto del *MIT* [2], liderado por Nicholas Negroponte,

[1] Begoña Gros Salvat es Doctora en Pedagogía y profesora titular en la Universidad de Barcelona. Se especializa en el tema de la utilización de nuevas tecnologías en el ámbito educativo. Ha publicado varias obras en esta área y artículos en revistas especializadas. Participa en proyectos de investigación sobre el uso de entornos tecnológicos en la enseñanza y la formación.

[2] Massachusetts Institute of Technology

denominado **OLPC** *(One Laptop Per Child)*, ha desarrollado una pequeña computadora de muy bajo costo, especialmente diseñada para niños. Este proyecto, sin fines de lucro, se encuentra actualmente en una fase inicial de producción y ya se están realizando pruebas piloto en comunidades educativas de varios países de la región.

En la figura 3.1, puede observarse a un grupo de niños en Brasil disfrutando de sus OLPCs fuera del aula, ya que las mismas están diseñadas para ofrecer autonomía y conectividad inalámbrica en cualquier espacio.

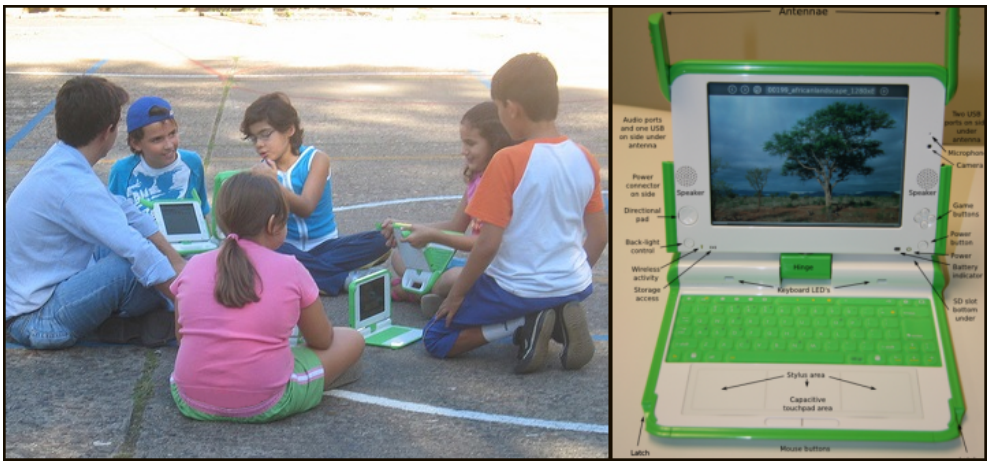

Figura 3.1: OLPCs en Brasil

Este tipo de equipamiento[3], especialmente diseñado para que lo utilicen los niños, tanto en la escuela como en su hogar, posee una gran robustez, escasos requerimientos de energía, comunicación inalámbrica y dispositivos de audio y video. Es decir, técnicamente poseen todo lo necesario para ser potencialmente una excelente herramienta educativa.

Sin embargo, contar con el equipamiento físico (Hardware) no es suficiente, también es necesario contar con aplicaciones adecuadas (Software).

[3]El proyecto OLPC, desarrolló una laptop cuyo costo apenas supera los 150 dólares (USD). Cuenta con un S.O. basado en Linux y posee una pantalla de modalidad dual, con ambos modos a todo color, modo de transmisión DVD, y una opción de pantalla secundaria reflectiva en blanco y negro, legible a la luz del sol a una resolución 3x. Además tiene un procesador de 500MHz y 128MB de memoria DRAM, con 500MB de memoria Flash (similar a la que utilizan los pen drives); no tiene unidad de disco duro, pero tiene cuatro puertos USB. Las OLPC tienen conexión de red inalámbrica que, entre otras cosas, les permite conectarse entre sí (mesh network); cada laptop puede comunicarse con su vecino más cercano, creando una red ad hoc, o red de area local. Estas computadoras de bajo costo usan fuentes de energía innovadoras (incluyendo la posibilidad de carga manual) y pueden hacer la mayoría de las tareas que realiza una computadora convencional con la excepción de guardar grandes cantidades de información.

Y para diseñar estas aplicaciones, es necesario tener en cuenta los requerimientos propios de los procesos de enseñanza-aprendizaje. Para ello, es muy importante conocer cuáles son los componentes procedimentales y los requisitos previos, con el objetivo de que sirvan de apoyo al nuevo aprendizaje. Ya que de nada sirve, una aplicación técnicamente impecable, si no está debidamente sustentada por un adecuado marco pedagógico.

3.1. Objetivos pedagógicos

Es posible diferenciar cuatro tipos de aplicaciones educativas, en función del objetivo pedagógico buscado. Los cuales podemos denominar como:

- **Tutorial:** Sirve para enseñar un determinado contenido. En general, se denomina tutoriales a cierto tipo de aplicaciones que desarrollan contenidos muy específicos, detallando claramente los pasos a seguir e incorporando ejemplos claros, que pueden reproducirse fácilmente. No debe confundirse este tipo de aplicaciones con los denominados Sistemas Tutoriales Inteligentes (STI), los cuales suelen realizar un seguimiento automático de la evolución del alumno, para adecuar el nivel de las explicaciones y del contenido, a su desempeño.

- **Práctica y ejercitación:** Son aplicaciones pensadas para desarrollar destreza en la utilización de conocimientos ya impartidos, lo que ayuda a asimilarlos. Por lo general, presentan actividades de identificación, comprensión y familiarización.

- **Simulación:** Las aplicaciones de simulación pueden ser de muy diverso tipo. Se pueden recrear situaciones reales para que los alumnos se familiaricen con ciertas tareas, como podría ser la utilización de una cuenta bancaria o la administración de una empresa. O se pueden simular procesos físicos, químicos ó nucleares, que por razones de costo o peligrosidad, no es posible realizar en forma real.

- **Hipertexto e hipermedia:** Su principal característica reside en la naturaleza no lineal de la navegación por el contenido. Permiten integrar diferentes estímulos, ya sea por medio de imágenes, sonidos, videos, etc. Se utilizan especialmente cuando se desea que el aprendizaje se produzca por medio del descubrimiento y la asociación.

La evolución de las aplicaciones educativas ha estado siempre fuertemente inluenciada por las diferentes teorías de aprendizaje y modelos instructivos.

Por lo tanto, es posible categorizar a las mismas, de acuerdo a las diferentes teorías de aprendizaje a las que adhieren. Los primeros diseños instruccionales estaban basados en un enfoque conductista, siendo uno de sus principios fundamentales, la descomposición de los contenidos en pequeñas unidades. El resultado, es un diseño de actividades en donde el estudiante debe dar respuestas y se utilizan refuerzos, positivos ó negativos, según corresponda. Un ejemplo de este tipo de aplicaciones, son los primeros programas de enseñanza asistida por computadora, en los cuales, la conducta del estudiante es modelada, no guiada y la actividad principal del alumno consiste en seguir las instrucciones y responder a las preguntas que se le presentan.

Posteriormente, aparecieron los diseños instruccionales basados en un enfoque cognitivista, por cuanto desarrollan prescripciones explícitas de las acciones instruccionales, que enfatizan la comprensión de los procesos de aprendizaje. En este caso, las estrategias utilizadas son heurísticas, los contenidos se plantean como tácitos y los conocimientos pueden ser de tipo conceptual, factual y procedimental, basados en la práctica y resolución de problemas.

En este tipo de aplicaciones se reconocen los siguientes elementos:

- Mayor interactividad orientada a la aplicación de simulaciones.
- Énfasis en el estudio de los niveles mentales de los alumnos y de la estructura cognitiva.
- Utilización de tecnologías multimedia para maximizar el aprendizaje.
- Objetivos instruccionales más integrales.

Posteriormente, aparecen diseños basados en el modelo constructivista, el cual privilegia la habilidad del alumno para crear interpretaciones por sí mismo y manipular las situaciones hasta que las asuma como proceso de aprendizaje. Este modelo pone especial énfasis en el valor del descubrimiento para futuros aprendizajes. Un ejemplo de este tipo de aplicaciones es el lenguaje LOGO, creado por Seymour Papert. [23]

Las Tecnologías de la Información y la Comunicación (TICs) han tenido una gran incidencia en la redefinición de los modelos de diseño instruccional, al hacerlos pasar de modelos centrados en la enseñanza a modelos centrados en el alumno. Estos últimos describen y promueven actividades que fortalecen la capacidad de un aprendizaje duradero, transferible y auto-regulable por parte del alumno, ya que concibe al sujeto como un ser que percibe, codifica, elabora, transforma la información en conocimientos, y la utiliza para la superación de problemas y la generación de nuevos conocimientos.

Internet, potencia la formulación de nuevos entornos de aprendizaje, ya que incluye procesos que son dirigidos a satisfacer los intereses, intenciones y objetivos del estudiante, y le proporciona estrategias y medios que le permiten abordar y comprender lo que es primordial para él. Por otra parte, las posibilidades de interconexión de Internet, permiten la utilización de entornos de trabajo compartido y/o colaborativo. [25]

TIPOS DE PROGRAMAS	**TEORÍAS DEL APRENDIZAJE**	**MODELOS INSTRUCCIONALES**
Enseñanza asistida por computadora. (primeros programas)	CONDUCTISMO	Aprendizaje basado en la enseñanza programada.
Programas multimedia de enseñanza, simulaciones, Hipertextos.	COGNITIVISMO	Aprendizaje basado en el almacenamiento y la representación de la información.
LOGO, micromundos.	CONSTRUCTIVISMO	Aprendizaje basado en el descubrimiento.
Programas de comunicación.	TEORÍAS SOCIALES DEL APRENDIZAJE.	Aprendizaje colaborativo.

Cuadro 3.1: Teorías de aprendizaje y Modelos instruccionales.

En el cuadro anterior, se resumen los diferentes tipos de aplicaciones informáticas, agrupados de acuerdo a las diferentes teorías de aprendizaje y modelos instructivos. Esta categorización se refiere a las aplicaciones educativas en general, ya sea que se trate de aplicaciones de escritorio ó de aplicaciones que corren en forma remota, por medio de un navegador. Sin embargo, para los propósitos del presente trabajo, nos concentraremos en las características distintivas de estas últimas.

3.2. Internet y Educación

Las aplicaciones de Internet, cuando se utilizan con propósitos educativos, presentan en general, algunas limitaciones importantes. Estas son básicamente tres, a saber: la desorientación que se produce en algunos estudiantes al recorrer sitios que contienen gran cantidad de información sobre los temas ofrecidos; la fatiga cognitiva que experimentan algunos usuarios al obtener

contenidos en diferentes formatos y de manera simultánea y, por último, la utilización de una única estrategia instruccional, asumiendo que los estudiantes están dotados de las mismas capacidades, conocimientos, experiencias y estilos para procesar y percibir la información que se les proporciona.[8]

Una alternativa para disminuir el impacto de las limitaciones anteriormente descriptas es adaptar la presentación de contenidos, los formatos de información y las opciones de navegación a las características de sus potenciales usuarios, esto con el fin de proporcionar la información más relevante y en los formatos más adecuados para cada uno de ellos.

En algunos sistemas la adaptación se materializa exclusivamente en la presentación de los contenidos, en otros, se incorporan además sugerencias para las opciones de navegación. Distintas y variadas estrategias instruccionales, que incluyen distintos formatos de información, organización de contenidos y secuencia de actividades, son utilizadas para ajustar los sistemas a las necesidades de los estudiantes-usuarios en función de sus respectivos estilos de aprendizaje [26].

Para lograr este objetivo en forma eficiente, es necesario contar con tecnologías que permitan la creación de aplicaciones de cierta complejidad. Que además, puedan ser adaptadas a cada estilo en una forma sencilla y rápida, aún a pesar de las limitaciones propias de la comunicación por HTTP (*Hyper-Text Transfer Protocol*). Nos estamos refiriendo, a las aplicaciones dinámicas de Internet.

3.3. Ventajas de las RIA en Educación

La utilización de las aplicaciones dinámicas de Internet *(RIA)* en el ámbito educativo presenta diversas ventajas. Desde el punto de vista estrictamente técnico, si las comparamos con las aplicaciones de escritorio, la ventaja fundamental radica en la uniformidad del entorno de ejecución. Es decir, las *RIA* corren dentro de un navegador web genérico, lo que evita los problemas de instalación de la aplicación y posibles incompatibilidades entre diferentes sistemas operativos. Esta ventaja no es tan apreciable cuando se trabaja en ámbitos controlados, como pueden ser las redes internas de las aulas-laboratorio, pero facilitan enormemente el acceso a los alumnos, cuando los mismos las utilizan desde sus propias computadoras, ó desde lugares de acceso público (ciber-cafés, locutorios, etc.).

Por otra parte, la sigla *RIA (Rich Internet Applications)*, hace referencia a un cierto enriquecimiento tecnológico y de utilización de los medios, en comparación con las aplicaciones de Internet tradicionales. Actualmente muchas *RIA* incorporan una gran cantidad de elementos multimedia dentro de

un entorno altamente interactivo. La utilización de estos recursos, posibilita la generación de contenidos con formatos más atractivos para los estudiantes, ya que se trata de presentarles entornos similares a los que ya están acostumbrados a utilizar en forma recreativa, en sitios tales como *Facebook*, *Youtube* o *MySpace*.

Uno de los principales beneficios que proveen las *RIA* en el aula es la posibilidad de diversificar las estrategias de enseñanza. Por ejemplo, en la Facultad de Wharton de la Universidad de Pensilvania, los profesores utilizan aplicaciones dinámicas de Internet como simuladores de negocios. Estos simuladores permiten a los estudiantes resolver problemas, estudiar casos y fundamentalmente visualizar los conceptos abstractos en un escenario similar a la realidad. [29]

Algunas productos comerciales como *Adobe Flex*, proveen de recursos para visualizar datos en forma de gráficas interactivas. En este contexto, los estudiantes pueden manipular directamente los datos en la simulación y a su vez visualizar las consecuencias de sus acciones. Esto refuerza la comprensión de los conceptos teóricos y a su vez le otorga al estudiante un mayor control sobre su aprendizaje.

Las *RIA* permiten a los estudiantes acceder a sus aplicaciones y a sus archivos desde cualquier sitio en el que dispongan de una conexión a Internet. Esto significa que pueden continuar con sus estudios, o el desarrollo de sus proyectos más allá del horario y del ámbito escolar. Si bien, el almacenamiento de archivos en un servidor central no es una idea nueva, el acceso a los mismos por medio de una aplicación específica que se descarga automáticamente y se ejecuta en un navegador de Internet, abre un nuevo espectro de posibilidades, entre ellas, la posibilidad de utilizar entornos de trabajo colaborativos.

Por último, no es menos importante mencionar el tema de las licencias de software. La mayoría de las aplicaciones comerciales, poseen licencias que prohiben tanto su copia, como su utilización a usuarios no autorizados. Actualmente existen aplicaciones dinámicas de Internet como *Google Docs* o *StarOffice*, que brindan la toda funcionalidad necesaria para crear y editar documentos de texto, planillas de cálculo, presentaciones, etc., con sólo contar con una conexión a Internet. Estas aplicaciones, no sólo son gratuitas y de acceso libre, sino que además, no precisan de instalación previa, simplemente se accede a las mismas a través de un navegador de Internet.

En los párrafos anteriores, hemos analizado las características de las aplicaciones dinámicas de Internet y las ventajas que presenta su utilización dentro del ámbito educativo. A continuación, analizaremos las características de un nuevo enfoque para construirlas, denominado *Seaside*.

Capítulo 4

Un nuevo enfoque.

Como se ha mencionado anteriormente, existen varias alternativas para desarrollar aplicaciones dinámicas de Internet, desde productos comerciales como Flex® o SilverLight® hasta proyectos gratuitos y de código abierto como OpenLazlo^{TM} y JavaFX^{TM} . La mayoría de ellas, permite la utilización de un conjunto de tecnologías conocidas como AJAX y dispone de diferentes herramientas de programación, dentro de un entorno de programación tradicional.

Sin embargo, el objetivo del presente trabajo es profundizar y exponer las características de un entorno muy diferente y casi desconocido para la mayoría de los desarrolladores. Se trata de un nuevo enfoque para implementar RIAs, denominado *Seaside*, que se diferencia notablemente de las alternativas tradicionales, por las características únicas de su entorno de desarrollo y por las posibilidades que brinda para la creación de aplicaciones Web. Aunque muchas de sus características distintivas provienen del hecho que *Seaside* está programado en *Smalltalk* [1].

Seaside, cuya traducción literal del inglés significa *costa*, proviene del acrónimo ***S****queak* ***E****nterprise* ***A****ubergines* ***S****erver* ***I****ntegrated* ***D****evelopment* ***E****nvironment* [2].

Desde un punto de vista estrictamente técnico, *Seaside* es un *framework*[3] para crear aplicaciones web complejas. Sin embargo, desde un punto de vista más conceptual, puede decirse que *Seaside* rompe con prácticas ampliamente

[1] Smalltalk es un lenguaje orientado a objetos puro.

[2] Según Avi Bryant, su creador, el origen de este acrónimo proviene de un juego de palabras entre "Enterprise JavaBeans" y "Enterprise Objects" (una parte de WebObjects de Apple que fue la principal inspiración de Seaside). Por otra parte, Aubergine es una referencia humorística, similar a la presente en el proyecto CLEE (Common Lisp Enterprise Eggplants), ya que Aubergine y Eggplant son denominaciones del mismo vegetal, la berenjena.

[3] Un framework, es un diseño reutilizable de un sistema de software.

difundidas en el desarrollo de aplicaciones Web, razón por la cual, también es conocido como *"el framework hereje"*. [27]

En el presente capítulo haremos un recorrido histórico y conceptual acerca de los orígenes de Seaside y de Smalltalk, ya que no es posible hablar de Seaside sin mencionar a Smalltalk. Por otra parte, veremos que Smalltalk representa algo más que un lenguaje de programación, ya que surgió como resultado de un proyecto de investigación en el campo educativo y cuyos creadores estuvieron fuertemente inspirados por las ideas de Montesori, Papert y Piaget [28]. Por esta razón, si bien Seaside recién está naciendo, posee toda la potencialidad para transformarse, en el futuro, en una excelente plataforma de desarrollo de aplicaciones educativas.

4.1. Smalltalk, Squeak y Seaside.

Avi Bryant desarrolló Seaside en *Squeak*[4]. Para comprender cabalmente, no sólo algunas de las características distintivas de Seaside, sino también parte de la filosofía subyacente en su diseño, es necesario profundizar un poco en los orígenes y características particulares de Squeak y de Smalltalk-80, su predecesor.

Smalltalk es un lenguaje de programación orientada a objetos puro, desarrollado en la década del 70 en el *Learning Research Group* bajo la dirección del **Dr. Alan Kay**[5], en los laboratorios de XeroX[6]. Con fuertes influencias de Simula[7], Smalltalk fue concebido como un lenguaje de programación basado en una poderosa metáfora, que podría resumirse en dos frases, *"en Smalltalk todo es un objeto"* y *"los objetos sólo se comunican entre sí, mediante mensajes"*.[16]

El diseño de Smalltalk tenía como meta construir una herramienta para que los usuarios, especialmente los niños, pudieran describir sus propios modelos del mundo real como modelos de computadora, permitiéndoles experimentar con ellos y de esa forma lograr un aprendizaje más significativo de conceptos físicos, matemáticos, biológicos, económicos, etc. Es decir, no se trataba simplemente de utilizar un programa fijo y trabajar con diferentes juegos de datos, sino que el usuario (programador) tuviera la capacidad de modificar en forma rápida y sencilla, el modelo conceptual subyacente.

[4]Squeak, creado en 1996, es una versión aggiornada de Smalltalk-80, con nuevas características multimedia.

[5]Dr.Alan Kay fue premiado con el Turing Award de la Association of Computing Machinery en 2003.

[6]XeroX PARC - Palo Alto Research Center

[7]Simula-67, es un lenguaje para simulación creado por Kristen Nygaard y Ole-Johan Dahl.

Para lograr este cometido, los cambios deberían ser rápidos, seguros e implementables por alguien enfocado sólo en obtener una solución para su dominio, despreocupándose de cuestiones técnicas tales, como los tipos de datos, el almacenamiento de los mismos o la administración de la memoria. [11]

En la figura 4.1, se observa el aspecto de la interfaz gráfica de Smalltalk en una computadora Alto, en los laboratorios de Xerox. En la misma, podemos observar gráficos detallados, ventanas solapables, diferentes tipos y tamaños de letra, dispositivos de señalamiento (mouse), etc. Sin duda, todos estos elementos nos resultan muy familiares actualmente, pero la figura en cuestión data de 1979 y los conceptos desarrollados para Smalltalk en ese momento fueron fundamentales para el desarrollo de los sistemas gráficos e interactivos con que actualmente cuentan las computadoras personales.

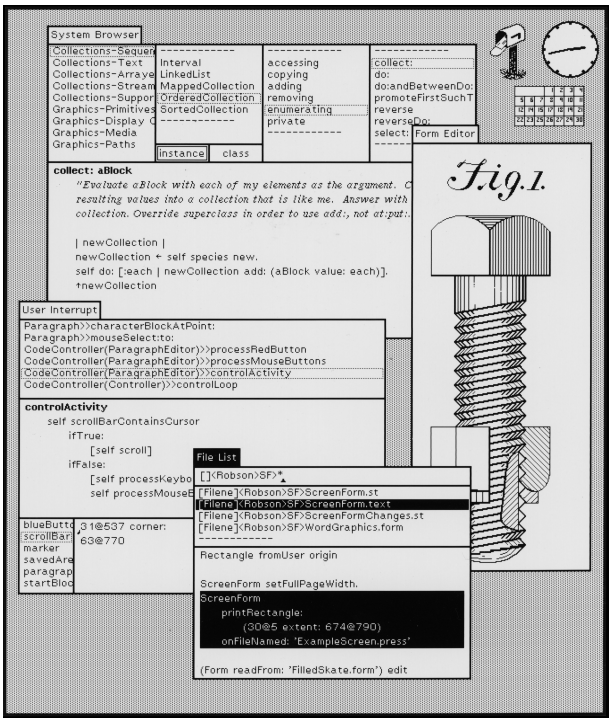

Figura 4.1: Interfaz gráfica de Smalltalk-80

Aunque es un hecho muy poco conocido, la historia de Smalltalk siempre estuvo fuertemente ligada a la educación y al estudio de los procesos de aprendizaje, sobre todo en los niños. El *Learning Research Group* de Xerox liderado por el Dr. Alan Kay, que le dió origen a fines de los 70, es ampliamente conocido por sus investigaciones sobre dos paradigmas dominantes de la computación: la interfaz gráfica de usuario y la programación orientada a

objetos. Sin embargo, dentro del amplio discurso de las nuevas tecnologías y el uso de Internet en educación, nunca se han reconocido adecuadamente los aportes realizados por Alan Kay en tal sentido. Un ejemplo de ello son las ideas expuestas en su tesis de maestría, "*The Dynabook*" - "*A Personal Computer For Children Of All Ages.*", en 1972. [17]

Aunque afortunadamente, muchas de esas ideas hoy se ven plasmadas en iniciativas educativas de alcance mundial, como el proyecto *OLPC*, desarrollado por el *MIT*[8], ya que varias de las aplicaciones incluidas en las OLPCs, están desarrolladas en Squeak[9], el actual sucesor de Smalltalk-80.

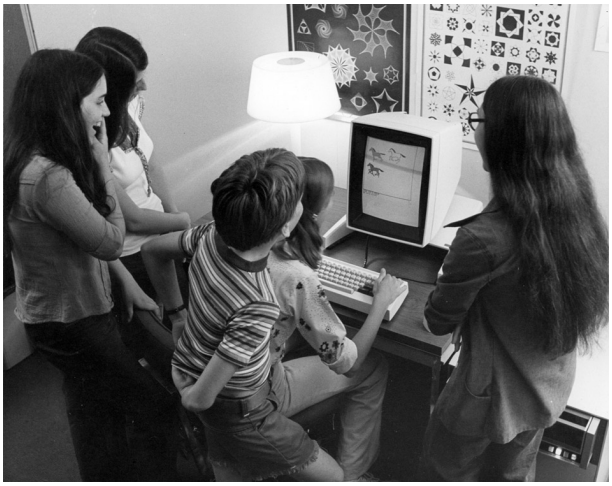

Figura 4.2: Niños experimentando en el Learning Research Group

La versión original de Smalltalk, implementada a comienzos de los 70 tenía una fuerte influencia del lenguaje LOGO de *Seymour Papert*[10], en términos tanto estéticos como sintácticos, aunque sus aspiraciones eran considerablemente mayores en muchos aspectos.[19] Como lo expresó Alan Knight *"En esencia, Smalltalk es un lenguaje de programación enfocado en los seres humanos, en lugar de las computadoras"*. [11]

[8]Massachusetts Institute of Technology.

[9]Etoys, un proyecto basado en Squeak, forma parte del software incluido en la OLPC (One Laptop Per Child)

[10]Destacado científico computacional, matemático y educador. Pionero de la inteligencia artificial, inventor del lenguaje de programación LOGO en 1968.

[11]Alan Knight es desarrollador Senior de Cincom Systems y es un destacado columnista y conferencista en temas relacionados con Smalltalk y programación orientada a objetos.

4.2. Squeak, el sucesor de Smalltalk-80

A pesar de las ideas fuertemente innovadoras contenidas en Smalltalk, o tal vez, precisamente por ello, Smalltalk nunca se transformó en un lenguaje masivo. Muy por el contrario, luego de una espectacular presentación en la revista Byte, en Agosto de 1981, seguida de una descripción detallada de sus características en libros como los de Adele Goldberg[12], Lalonde y Pugh [18] entre otros, su presencia en el mundo de la computación fue languideciendo con el correr del tiempo. Los integrantes del grupo original de XeroX Parc, Alan Kay, Dan Ingalls, Adele Goldberg y Ted Kaheler fueron desarrollando sus propios proyectos y por bastante tiempo, poco se supo de ellos. Apenas un par de empresas[12] desarrollaron versiones comerciales, pero los altos precios de las mismas, limitaron su penetración en el mercado y la consiguiente difusión del lenguaje.

Sin embargo, el 1ro. de Octubre de 1996, un escueto mensaje de correo electrónico enviado a la lista *comp.lang.smalltalk* por Dan Ingalls, volvió a instalarlo en escena. El título del mensaje era "*Squeak - A Usable Smalltalk written in itself*" y marcaba el nacimiento de una nueva etapa. [15]

Desde el título mismo del mensaje, Dan Ingalls ponía en evidencia una diferencia fundamental presente en Squeak con respecto a otros lenguajes. *Squeak es un Smalltalk programado en sí mismo.* Esta característica, junto a un tratamiento totalmente homogéneo de sus componentes internos, le confiere a Squeak características únicas de *"reflexión"*, es decir la capacidad de interactuar y analizar el funcionamiento mismo del sistema, mientras éste se está ejecutando.

Este nuevo proyecto fue inicialmente auspiciado por Apple Inc. y luego por Walt Disney Imagineering, quienes contrataron a varios miembros del grupo original de XeroX para el desarrollo de Squeak. A partir de ese momento, comenzaron a desarrollarse una gran cantidad de proyectos basados en Squeak, en áreas tan diversas como, nuevas interfaces de usuario (Morphic), gráficos 3D (Ballon3D), herramientas para la web (Celeste, Scamper, Swiki), servidores web (Comanche, Swazoo), etc.

Muchos de estos proyectos pusieron de manifiesto sus impresionantes capacidades gráficas e interactivas y dieron origen a aplicaciones educativas como *Plopp*, una herramienta de dibujo 3D para niños, *Scratch*, una herramienta de programación por medio de bloques, con la cual es posible construir programas en forma similar a las maquetas que se construyen con LEGO^{TM} o *Wonderland-Alice*, una herramienta de autor, con la que se pueden crear personajes tridimensionales y generar animaciones.

[12]ParcPlace y Digitalk, ambas establecidas en California.

La difusión gratuita de Squeak a través de la web y una licencia poco restrictiva, permitieron además, que los proyectos se multiplicaran y que la comunidad Squeak fuera creciendo. A partir de ese momento, la conectividad creciente entre computadoras y la proliferación de páginas y sitios web generó un nuevo contexto de desarrollo para Squeak, y fue precisamente en este contexto, que nació *Seaside*.

4.3. Origen y evolución de Seaside.

La mayoría de los proyectos surgidos a partir de Squeak estaban enfocados en la utilización de sus capacidades multimedia, las cuales brindaban una excelente plataforma para aplicaciones educativas, prueba de ello, son los desarrollos llevados a cabo por SqueakLand (www.squeakland.org) [1] y por Squeakpolis (squeak.educarex.es/Squeakpolis) [6], [5], esta última, una excelente iniciativa de la Junta de Extremadura (España).

Aunque las capacidades gráficas, de sonido y de simulación, formaban la base de la mayoría de los proyectos basados en Squeak, su desarrollo no podía permanecer ajeno al crecimiento de Internet. Así surgió *Seaside*, un proyecto diseñado para la creación de aplicaciones dinámicas de Internet. Actualmente, Seaside ocupa un lugar destacado y se ha transformado en uno de los principales medios de difusión de Squeak, en ámbitos no relacionados directamente con Smalltalk.

Seaside fue presentado públicamente en la lista de correo para programadores *squeak-dev*, el 21 de Febrero de 2002, por sus creadores Avi Bryant y Julian Fitzell, quienes lo diseñaron, principalmente para utilizarlo en su compañia de desarrollo web y consultoría. La primera versión de Seaside estaba fuertemente inspirada en una aplicación previa llamada *"Iowa"*, escrita también por Avi, en *Ruby*[13]. Esta versión ya proveía de llamadas a funciones (action callbacks) para los enlaces y formularios, manejo del estado de la sesión con soporte para evitar el problema del botón de retroceso y un sistema basado en componentes.

Casi inmediatamente después de su aparición, comenzaron a trabajar en las versiones 2.X, bajo el nombre clave *Borges*, en referencia al cuento corto *"El jardín de los senderos que se bifurcan."*, de Jorge Luis Borges[4], aludiendo al hecho que, Seaside mantiene el control sobre el estado de las diferentes sesiones que se bifurcan.

La principal novedad de estas versiones, fue la creación de una nueva arquitectura basada en capas:

[13]Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro Matsumoto.

- Una capa de núcleo (Kernel) que provee el soporte requerimiento-respuesta sobre HTTP. Esta capa es la que permite el control del estado de las diferentes sesiones.

- Una capa de visualización. Esta capa provee un conjunto de funciones para generar el código HTML necesario para visualizar los diferentes componentes de la aplicación.

- Una capa de componentes. Esta capa maneja la semántica necesaria para la interrelación de los distintos componentes y las herramientas de desarrollo.

Como Seaside es un proyecto libre y de código abierto, en la medida que sus autores originales fueron necesitando más tiempo para dedicarle a sus propios desarrollos comerciales, otros integrantes de la comunidad comenzaron a realizar valiosos aportes al proyecto. De esta forma, programadores de todo el mundo liderados por Lukas Renggli, Philippe Marschall y Michel Bany, tomaron a su cargo la tarea de limpiar, arreglar y simplificar el código original.

En Octubre de 2007, se lanzó la versión 2.8, con importantes mejoras de optimización y de portabilidad. Mucho más veloz y con menos requerimientos de memoria que las anteriores, hoy cuenta con versiones en la mayoría de los dialectos de Smalltalk.

4.4. Características de Seaside

A continuación presentaremos una breve descripción de las principales características técnicas de Seaside. Más adelante, en el capítulo 6, analizaremos con mayor detalle algunas de ellas, por medio de un ejemplo de aplicación.

- **Generación programática de XHTML.** Seaside provee una aproximación muy diferente para la generación de XHTML (*eXtended Hyper Text Markup Language*), en comparación con los sistemas basados en plantillas (templates). XHTML es un lenguaje que cumple con especificaciones más estrictas que el clásico HTML, al cual se supone sustituirá, en la creación de páginas web. Este tipo de lenguajes basados en marcas (tags) establecen una estructura de anidamiento, en la cual todos los atributos tienen una marca de comienzo y una de finalización. En los sistemas basados en plantillas (templates), es responsabilidad del programador controlar el correcto anidamiento de las marcas (tags) y de los atributos, para que la página web se genere correctamente.

En cambio en Seaside esto no es necesario, ya que cada componente genera automáticamente el código XHTML correcto, que describe su visualización en el navegador. Es decir, para construir una aplicación, sólo se necesita pensar en términos de los componentes necesarios para cumplir con la funcionalidad deseada, ya que los aspectos visuales son generados automáticamente por el componente mismo.

- **Manejo de peticiones basado en Callbacks.** En las páginas web tradicionales es necesario que exista un único nombre para cada enlace y formulario de entrada, para poder responder adecuadamente a las peticiones. Sin embargo, Seaside automatiza este proceso y permite la asociación de bloques (block clousures)[14] en lugar de nombres, con campos de entrada y enlaces. Es decir, Seaside enlaza las acciones a ejecutar (callback actions) directamente con cada componente. De esta forma, a cualquier componente se le puede asignar un bloque, para que en el momento adecuado ejecute las acciones programadas en el mismo. La principal ventaja, además de la increible flexibilidad que esto representa a la hora de programar, es que las aplicaciones se estructuran en términos de objetos y mensajes, en lugar de identificadores y cadenas de caracteres. Es decir, de la misma forma que cualquier aplicación hecha en Smalltalk.

- **Componentes embebidos.** En lugar de estructurar las aplicaciones como un conjunto de páginas independientes, Seaside permite la construcción de la interfaz de usuario como un árbol individual, con objetos componentes que poseen un estado y donde cada uno de ellos encapsula una pequeña parte de la página. De esta forma, las aplicaciones se programan de forma muy similar a cualquier aplicación de escritorio. Es decir, una vez decidida la funcionalidad deseada, se incorporan los componentes necesarios para satisfacerla. Cada componente de la página será responsable de su propia visualización, así como de la invocación de las funciones establecidas. De esta forma, las aplicaciones se crean relacionando a los diferentes componentes, con el resto de los objetos propios del dominio.

- **Manejo modal de sesiones.** A diferencia de los modelos basados en *servlets*[15] que requieren de un manejador separado para cada página

[14]Son funciones sin nombre, que pueden almacenarse en variables, enviarse como parámetro o ejecutarse en forma diferida.

[15]La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor.

o petición, Seaside modela una sesión de usuario completa como una pieza contínua de código. En Seaside, los componentes pueden llamarse entre sí, y retornar respuestas como si fueran subrutinas y es posible encadenar dichas llamadas en un método, como si se tratase de cajas de diálogo modales. Esto facilita enormemente la programación de aplicaciones, ya que cada componente se encarga de una tarea determinada, como por ejemplo, el ingreso y la validación de los datos, o la impresión de un reporte. Esto permite acometer una tarea compleja, dividiéndola en varias tareas más pequeñas y sencillas.

- **Control de flujo múltiple y simultáneo.** Cada componente además, puede definir su propio control de flujo de información, en forma totalmente independiente de cualquier otro componente presente en la misma página. Esto permite implementar la lógica de la aplicación, distribuida en múltiples páginas, como si se tratara de una única pieza de código. La característica que permite esto, está basada en el concepto de *"continuations"*, que aprovecha la habilidad de Smalltalk para acceder y manipular la pila *(stack)* de ejecución. Esto permite resolver, entre otras cosas, el problema del botón de retroceso. Ya que, aunque cada página se genera dinámicamente, Seaside tiene incorporada toda la información necesaria para reconstruir dicha página, cuando se retrocede a la página anterior. Sin embargo, es necesario tener en cuenta que, mantener toda esta información requiere de una apreciable cantidad de memoria.

Además de las características ya mencionadas, Seaside tiene un entorno de programación bastante diferente de los entornos tradicionales. Sus características dinámicas, heredadas de Squeak, son las que le permiten al programador, realizar modificaciones en la aplicación mientras la misma se está ejecutando. Siendo posible realizar esto, inclusive en forma remota, sin la necesidad de detener la aplicación o de reiniciar el servidor luego de cada modificación.

Capítulo 5

Un caso de aplicación de Seaside.

En los capítulos anteriores, se expusieron las características de las aplicaciones de Internet y se mencionaron algunas de las ventajas que presenta Seaside para generarlas, tanto desde el punto de vista del entorno de desarrollo, como de su posterior desempeño. En el presente capítulo, se hará una breve descripción de las características de un proyecto interdisciplinario, que reúne a grupos de investigación de la Facultad de Psicología y de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata, este proyecto permitió explorar y comprobar algunas de las ventajas presentes en Seaside.

Una parte de este proyecto, denominado *"Mapeo de redes semánticas aplicado a Educación y Neuropsicología. Evaluación de la calidad de la respuestas."*, consistió en el desarrollo de un conjunto de herramientas para evaluar, visualizar y comparar la constitución de las redes semánticas. Estas herramientas tienen por objetivo, la implementación informática del método *Distsem* (Vivas, 2004, 2007, Vivas et al., 2008)[31], el cual puede aplicarse al análisis de las respuestas, en grupos de individuos con características muy diferentes, tales como alumnos de nivel universitario, niños en edad pre-escolar o pacientes con enfermedad de Alzheimer.

Al comenzar la etapa de implementación, se decidió desarrollar estas aplicaciones en Smalltalk para aprovechar las características de simplicidad del lenguaje y la rapidez para la obtención de prototipos, contando así, con un modelo robusto pero flexible a la vez, que permitiera una rápida adaptación a diferentes contextos.

Una de las etapas del método Distsem consiste en la ponderación, por parte de los participantes del experimento, de las relaciones entre ciertos conceptos que se les presentan. Inicialmente, estos datos eran obtenidos por medio de una aplicación de escritorio instalada en varias computadoras de la Facultad de Ingeniería. Y posteriormente, se hizo una prueba piloto con un grupo de estudiantes, a quienes se les envió copias del programa por correo

electrónico para que lo instalaran en sus computadoras y luego retornaran sus respuestas por el mismo medio.

Inmediatamente fue evidente que, para poder acceder a un mayor número de participantes, distribuidos en un marco geográfico mucho más amplio, era necesario implementar este proceso como una aplicación de Internet. Sin embargo, esta aplicación debía ser capaz de manejar una lógica relativamente compleja, e integrar de la forma más sencilla posible, el modelo pre-existente.

Por otra parte, junto con la aparición de la versión 2.8 de Seaside, una de las más rápidas y estables hasta el momento, se publicó el primer libro [24] detallando sus características, ya que hasta ese momento la mayor parte de la información sobre Seaside estaba dispersa en correos electrónicos, blogs y en presentaciones a congresos. Por lo tanto, se tomó la decisión de utilizar Seaside como herramienta de desarrollo, ya que permitía incorporar el modelo ya desarrollado en Smalltalk practicamente sin cambios y colocarlo en la web.

La implementación de Infosem-Web, permitió además, explorar las posibilidades de Seaside para desarrollar futuros proyectos relacionados con el mejoramiento de la enseñanza en Ingeniería. Si bien, el objetivo principal de Infosem-Web no es estrictamente pedagógico, el mismo se encuentra enmarcado en un proyecto de investigación más amplio, para el mejoramiento de los procesos de enseñanza/aprendizaje.[14]

A continuación, se describirá brevemente, qué se entiende por red semántica y cómo el método Distsem procesa la información provista por los participantes del experimento, para evaluar y comparar sus respectivas redes semánticas.

5.1. Relaciones y distancias semánticas

En una red semántica clásica, dos conceptos se hallan semánticamente relacionados si se encuentran próximos en la red. Podemos medir la *"proximidad"* como la distancia literal entre ambos, esto es, la longitud del camino que ambos comparten. Cuando una persona estima la similitud semántica entre dos o más ideas puede establecer entre ellas diferentes tipos de relaciones semánticas. Su proximidad puede estar dada porque ambos conceptos presentan una relación lógica inferencial, pero también porque ambos conceptos pueden compartir numerosos atributos por medio de los cuales se establezcan relaciones no necesariamente lógicas. Las semejanzas en los atributos compartidos entre dos conceptos pueden promover el establecimiento de relaciones analógicas que se hallan facilitadas por la presencia de activación en las etiquetas respectivas. Algunos estudios sobre relaciones semánticas [2] sugieren que distintos procesos cognitivos permiten obtener relaciones parte-todo,

causa-propósito, contraste, etc. De hecho, estos estudios recuperan trabajos que proponen taxonomías de, al menos, trece sistemas de clasificación diferentes, que varían, en el marco de la teoría de la propagación de la activación, del control ejecutivo que la persona pueda ejercer cuando produce una activación de la red semántica. En la figura 5.1 se observa, una visualización de la red semántica de un experto en un dominio particular, en este caso, relacionado con conceptos de Análisis Numerico.

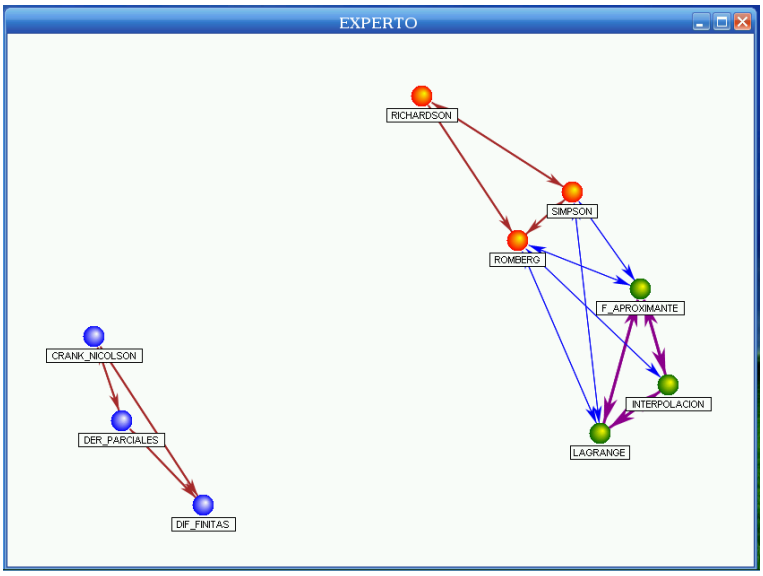

Figura 5.1: Red semántica del experto.

5.2. El método DistSem

El método Distsem, consiste en una utilización particular del Análisis de Redes Sociales [32], donde se ha substituido la canónica utilización de nodos-agentes sociales por nodos-conceptos y los vínculos tradicionalmente considerados como relaciones sociales por vínculos a partir de la distancia semántica entre conceptos.

Este método permite capturar las estimaciones de proximidad semántica brindada por los participantes y su inclusión en una matriz, sobre la cual es posible visualizar y comparar cuali y cuantitativamente, las redes semánticas de los individuos con un único nivel de restricción definido por el número limitado de conceptos previamente definidos.

El procedimiento se desarrolla según las siguientes etapas:

- **Confección de matrices y pares de conceptos:** Se seleccionan los conceptos cuya vinculación semántica se desea conocer. Se genera una matriz cuadrada de conceptos contra conceptos. De acuerdo a la naturaleza del problema, se decide la mejor consigna de escalamiento de pares de conceptos según su similitud / disimilitud y se generan pares de conceptos. La cantidad de pares resulta de aplicar n*(n-1)/2 para relaciones no direccionales. Se agregan cuatro pares repetidos con orden invertido para evaluar consistencia interna y se presentan en forma aleatoria a los participantes.

- **Información de los participantes:** Se solicita, tanto a los expertos en el dominio, como a los participantes, que estimen la similitud (proximidad) entre los pares de conceptos presentados. Estos pares de conceptos, que originalmente se administraban por medio de una única planilla impresa, actualmente se presentan de a uno, en un orden aleatorio distinto para cada participante y con diferentes alternativas para su ponderación.

- **Evaluación:** Como en la matriz resultante cada concepto queda definido por un vector constituido por los valores respecto a los otros n conceptos, estimados por cada sujeto, se aplica un procedimiento de escalamiento métrico multidimensional de objetos, en este caso conceptos, para generar un espacio semántico bidimensional para cada participante. Para conocer los agrupamientos semánticos producidos por cada participante, se aplica Análisis de Cluster Jerárquico a cada matriz en base al proceso propuesto por Johnson's (1967). De este modo se generan los agrupamientos por mayor cohesión (menor distancia) entre subgrupos y su relación con la totalidad. Se presenta en pantalla o se imprime, el gráfico de redes de la matriz seleccionada. Esto permite visualizar y analizar cualitativamente la configuración de la matriz de distancias para cada participante. Para comparar cuantitativamente la similitud entre las matrices producidas por los participantes entre sí o contra la matriz del experto, se aplica el método QAP (Quadratic Assignment Procedure) propuesto por Hubert y Schultz, (1976). Esta operación permite obtener el coeficiente de correlación de Pearson entre ambas matrices.

- **Análisis de los resultados:** El procedimiento descrito permite diferentes perspectivas y niveles de análisis según los intereses del investigador: Visualizar la red semántica que vincula los conceptos. Ver su distancia relativa en el plano. Apreciar la fortaleza de sus conexiones

en función del color y trazo de los lazos. Las relaciones que constituyen ideas asociadas se agrupan en nodos con igual color. *Evaluación cualitativa:* Se puede visualizar ausencias, excesos e impertinencias de enlaces entre los conceptos. *Evaluación cuantitativa:* Permite medir el nivel de similitud entre la red semántica de cada participante en diferentes momentos, su grupo y con una matriz considerada correcta.

5.3. Implementación informática de Distsem.

Las etapas descriptas de Distsem y todos los procesos requeridos se implementaron informáticamente en Infosem, para facilitar su aplicación a diferentes ámbitos e intereses de investigación. Es decir, procesar las estimaciones de proximidad semántica brindadas por los sujetos y posteriormente, analizar y visualizar su relación y distribución en dos dimensiones.

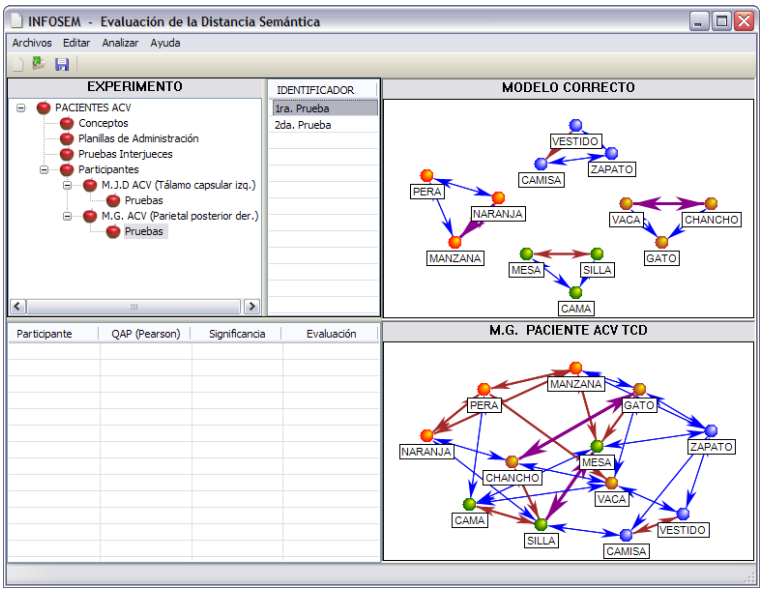

Figura 5.2: Análisis cualitativo en Infosem.

En la figura 5.2 se aprecian dos redes semánticas, la superior corresponde a una persona sana (modelo correcto) y la inferior a un paciente que sufrió un accidente cerebro-vascular. Como puede observarse, las relaciones establecidas entre conceptos referidos a frutas, animales, vestimenta y mobiliario, son completamente diferentes.

La implementación de Infosem en Smalltalk permitió una gran flexibilidad para experimentar cambios rápidos en el modelo, ya que éste se fue modificando y actualizando en sucesivas iteraciones.

En base a lo expuesto, Seaside apareció como la opción natural para brindar acceso a Internet a ciertos módulos de Infosem, ya que permitía:

- Integrar en forma directa y completa el modelo de objetos ya existente.
- Aprovechar desarrollos en tecnologías como AJAX, directamente desde Smalltalk.
- Mantener el flujo de información de la aplicación en forma transparente.
- Aprovechar las herramientas de desarrollo de Smalltalk (debuggers, inspectors, que permiten modificación de código y compilación en forma dinámica, etc.).

En resúmen, para la creación de la aplicación web no fue necesario realizar casi ningún cambio en el modelo existente. Se utlizaron recursos AJAX (Scriptaculous), sin necesidad de programar en javaScript. Se interactuó con el modelo existente en forma transparente por medio de componentes Seaside o de subclases de los mismos. Y por último y no menos importante, se aprovecharon las facilidades de las herramientas de depuración propias de Smalltalk, incluso en forma remota.

En el próximo capítulo, analizaremos en detalle cada una de las características más sobresalientes de Seaside, utilizando partes de Infosem-Web, como ejemplo de aplicación.

Capítulo 6

Seaside en profundidad

Inicialmente, el proceso de ponderación de las relaciones entre los diferentes conceptos, por parte de los participantes, se realizaba utilizando una planilla impresa, y en una etapa posterior, por medio de una aplicación de escritorio. Sin embargo, ambas opciones imponían ciertas limitaciones. Por lo tanto, para poder acceder a una mayor cantidad de participantes distribuidos en un marco geográfico más amplio, era necesario convertir este proceso en una aplicación de Internet.

Para ello se diseñó un módulo de ingreso de estimaciones, cuyos requerimientos eran los siguientes:

- Cada usuario, podrá acceder al módulo de ingreso de estimaciones por medio de una dirección de Internet *(URL)*.
- Una vez identificado en el sistema, podrá ingresar las respectivas ponderaciones.
- Las ponderaciones no se ingresarán directamente en forma numérica, sino por medio de una barra de desplazamiento *(slider)*, para lograr que esta valoración sea intuitiva y no esté condicionada por sus límites, ni por sus niveles de cuantificación.
- Los pares de conceptos se presentarán a cada usuario en un orden aleatorio y éste tendrá la posibilidad de avanzar o retroceder a voluntad, para modificar las estimaciones ya realizadas.

Este módulo se implementó y se utilizó recientemente para obtener datos de más de 200 estudiantes de la carrera de Psicología de la Universidad Nacional de Mar del Plata, a quienes se les solicitó, estimaran la proximidad entre pares de conceptos relacionados con una asignatura de su especialidad.

A continuación, analizaremos con mayor detalle algunas de las características presentes en Seaside, basándonos en la experiencia adquirida al desarrollar esta aplicación.

6.1. Los componentes de Seaside

En una aplicación desarrollada con Seaside, las principales entidades son ciertos objetos, denominados **componentes**, los cuales permiten agrupar diferentes objetos y sus contenidos, en una única entidad conceptual. Estos componentes son los responsables de definir tanto la interfaz de usuario, como el control del flujo de información de ciertas partes de la aplicación. La mayoría de estos componentes son instancias de clases definidas por el programador, que heredan de la clase *WAComponent*, o de alguna de sus subclases, y definen la apariencia y el comportamiento de una porción de la página, es decir, de la cara visible de la aplicación.

Una página Web sencilla suele contener información y alguna forma para acceder a otras partes de la misma, como por ejemplo un menú. Si construyéramos dicha página como una pieza monolítica, deberíamos incluir en la misma, no sólo la información propiamente dicha, sino todo el código necesario para navegar por ella, incluyendo además, su formato, tipografía, colores, etc. Es de hacer notar, que una página construida de esta forma, se vuelve extremadamente complicada y esto dificulta notablemente su mantenimiento.

Precisamente para evitar este problema, es que Seaside utiliza el concepto de componentes. Los componentes son una forma de dividir semánticamente la aplicación (página) en partes, lo que facilita su creación y mantenimiento. En el caso anterior, uno de sus componentes, evidentemente es el menú. Las instancias de este componente son las responsables de manejar la información acerca de su propio estado actual, como por ejemplo, cuáles de sus ítems están habilitados en determinado momento y cuáles no. Por otra parte, la página Web correspondiente al ítem seleccionado del menú también es un componente, que a su vez, puede contener a otros componentes. Estos componentes pueden ser componentes simples, como formularios, listas, etc. o componentes complejos, formados por un conjunto de componentes simples.

En nuestro caso de aplicación, los participantes de una prueba necesitan ser identificados por el sistema y si aún no están registrados en el mismo, deben tener la posibilidad de hacerlo. Para ello, debemos crear un componente específico *(SDWRegisterComponent)*, que será el encargado de registrar a los participantes en el sistema.

Dentro de este componente, hay actividades que pueden manejarse de manera conjunta, como el ingreso de los datos requeridos, y otras que deben

realizarse de manera independiente, como es el caso de los avisos de error, que pueden producirse cuando el participante no ingresa correctamente la información requerida. La utilización de otro componente *(SDWMessage-Component)* para esta sencilla tarea, muestra una diferencia fundamental con un diseño monolítico.

En este caso, el componente de mensajes, se encarga de todos los aspectos relacionados con el aviso de una condición anómala. Es decir, este componente sabe como mostrar un mensaje y sabe como diferenciar un mensaje de error, de una advertencia. Es decir, conoce que tipo, tamaño de letra y color de fondo utilizar en cada caso y se encarga de borrar el mismo cuando ya no es necesario. De esta forma, los demás componentes, sólo deben avisarle cuando alguna acción del usuario amerita una advertencia o un mensaje de error.

En la figura 6.1 se muestran los componentes presentes en el formulario de registro, entre ellos el componente que muestra las advertencias y los mensajes de error.

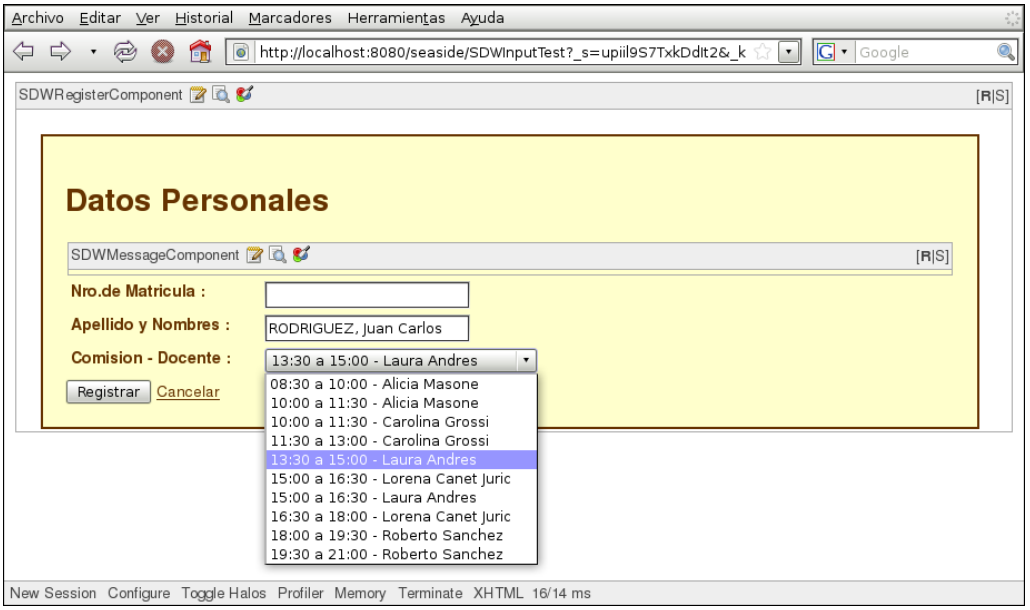

Figura 6.1: Componentes en Seaside

Así es, como los componentes pueden ser considerados como vistas y controladores de la tríada MVC[1]. Es de hacer notar que, a diferencia de los enfoques tradicionales, en este caso, las instancias de los componentes suelen

[1]Model-View- Controller es un framework, que permite separar el comportamiento del modelo, de la forma como se visualiza y de los medios para interactuar con el mismo.

existir durante todo el tiempo que permanece activa la sesión del usuario . De esta forma, la sesión delega en los componentes la atención de las peticiones del usuario, los cuales se encargan de su correspondiente representación visual *(rendering)* y del envío de mensajes a otros objetos *(callbacks)*.

Así es como, cada componente visible en una aplicación responde al mensaje *"renderContentOn:"*, que genera el código XHTML necesario para su visualización. De esta forma, cada componente sabe perfectamente como debe representarse en la ventana del navegador. Y si un componente está, a su vez compuesto por otros, cada subcomponente se hará responsable de su propia representación. Esto permite dividir las responsabilidades necesarias y encapsularlas, lo que da como resultado, un diseño mucho más comprensible y fácil de mantener y extender.

6.2. Integración de recursos

Para que las aplicaciones web puedan exhibir un comportamiento similar al de una aplicación de escritorio, las mismas deben ser capaces de aprovechar las capacidades que brindan actualmente los navegadores de Internet. Para ello, no basta con generar código XHTML que cumpla con las exigencias del W3C, también es necesario un cuidado diseño de la interfaz de usuario y contar con librerías de funciones que aporten nuevos recursos a los procesos interactivos.

6.2.1. Generación dinámica de código XHTML

Una de las primeras cosas que suele sorprender a quienes están acostumbrados a desarrollar aplicaciones web en forma tradicional, es que Seaside no utiliza plantillas *(templates)* para la creación de aplicaciones, sino que utiliza un enfoque completamente diferente.

El código XHTML de las aplicaciones web construidas con Seaside no proviene de plantillas, sino que es generado dinámicamente. Dentro de Seaside se ha construido un lenguaje específico que le permite escribir código XHTML, ya que Smalltalk es capaz de expresar el mismo árbol sintáctico abstracto de marcas *(tags)* y atributos que expresa XHTML. En la figura 6.2, se puede observar el código HTML generado automáticamente por Seaside para representar la página inicial de ingreso de datos de los participantes.

De esta forma, el código XHTML es generado dinámicamente a partir de Smalltalk, es decir, sin la necesidad de escribir XHTML en forma directa, ni utilizar estructuras prefabricadas *(plantillas)*. Pero hay algo más importante, como cada componente genera su propia visualización por medio de méto-

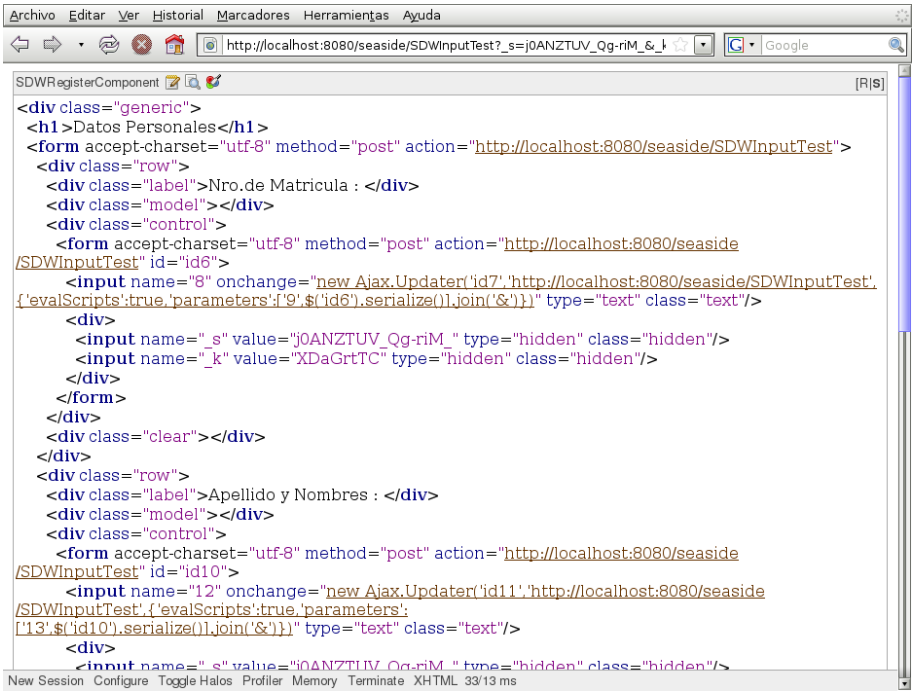

Figura 6.2: HTML generado dinámicamente.

dos que generan XHTML, es posible aplicar factorizaciones al código, que facilitan enormemente el mantenimiento de páginas de gran complejidad.

6.2.2. Aspectos estéticos

Hasta ahora nos hemos enfocado en los aspectos funcionales de las aplicaciones dinámicas de Internet. Sin embargo, un aspecto muy importante a tener en cuenta, sobre todo en aplicaciones educativas, es el aspecto estético. Para esta tarea, se suele recurrir a profesionales del diseño gráfico, quienes por lo general, prefieren mantenerse al margen de las tareas de programación. Por esta razón, es conveniente separar los aspectos funcionales de las consideraciones estéticas de la presentación.

La utilización de *CSS (Cascade Style Sheets)* permite separar los aspectos funcionales de los estéticos, de forma tal, que puedan ser manejados de manera totalmente independiente. En la figura 6.3 podemos apreciar, dos versiones de la misma página web. En la parte superior, la página se muestra en crudo, es decir, sin especificar ninguna caracterítica sobre su representación visual. En cambio, en la parte inferior, se observa la misma página pero con un aspecto estético totalmente diferente. Este cambio de aspecto no forma parte del código HTML de la página, sino que se encuentra en un archivo

Figura 6.3: Hojas de Estilo - CSS

totalmente separado, denominado *"Hoja de Estilo"*.

No sólo se pueden utilizar hojas de estilo en las aplicaciones desarrolladas con Seaside, sino que, por medio de un editor integrado, al que puede accederse mientras la aplicación se encuentra corriendo en el modo de desarrollo, es posible modificar el aspecto estético de los componentes.

Recientemente, un nuevo proyecto demoninado *SeaBreeze* [7], permite editar en forma gráfica el aspecto de las aplicaciones Web 2.0, desarrolladas con Seaside. *SeaBreeze* consiste en un conjunto de editores que se integra perfectamente con las herramientas existentes en Seaside. Se trata de una herramienta muy interesante, tanto para desarrolladores que recién se inician con Seaside, como para aquellos con gran experiencia en Smalltalk, pero

no demasiado acostumbrados a lidiar con *HTML* y *CSS*. De esta forma, es posible controlar los aspectos estéticos de la interfaz de usuario, en una forma más sencilla y amigable.

6.2.3. Interacción y Web 2.0

Actualmente existe una tendencia a crear aplicaciones Web fuertemente interactivas y con prestaciones similares a las de escritorio. Como mencionamos anteriormente, mediante la utilización de AJAX, es posible aumentar el nivel de interacción con el usuario, actualizando cada parte de la aplicación en forma independiente, en lugar de actualizar la página completa. Seaside posee una perfecta integración con uno de los frameworks de AJAX más populares, *Scriptaculous*.

Scriptaculous fue creado por Tomás Fuchs [2] y está basado en el framework *Prototype*. El intercambio de datos se realiza por medio de mensajes en XML, por lo que no es necesario utilizar un lenguaje especial de programación del lado del servidor. Lukas Renggli [3] integró *Scriptaculous* en Seaside, de forma tal, que no fuera necesario programar ni una línea en javaScript, ya que todo puede programarse directamente en Smalltalk.

También existen otras iniciativas que se complementan perfectamente con Seaside, como *SeaChart* (seachart.seasidehosting.st), que permite integrar gráficos de torta, de barras y otros efectos. Y *ShoreComponents* (shorecomponents.seasidehosting.st), que ha desarrollado una gran variedad de controles, estructuras de árbol y diseños de tablas para reportes, que facilitan la creación de visualizaciones de los resultados.

6.3. Control del flujo de información

Una de las principales dificultades al desarrollar aplicaciones web se debe a la asimetría de la relación cliente-servidor, en la cual el servidor sólo responde a las peticiones del cliente. Por otro lado, los navegadores web ofrecen al usuario facilidades de navegación que frecuentemente conducen a los servidores a no poder manejar el estado de los clientes. Esto último, se conoce como *"el problema del botón de retroceso"* (back button problem) y en

[2] Tomás Fuchs es un arquitecto de software austríaco. Desarrolla aplicaciones web desde 1996 y contribuyó al desarrollo de *Prototype*, un framework orientado a objetos contruido en AJAX/javaScript.

[3] Es desarrollador del núcleo de Seaside y es autor de varios frameworks en Smalltalk, tales como Magritte y Pier.

aquellas aplicaciones que generan su contenido en forma dinámica pueden dar lugar a respuestas como la que se observa en la figura 6.4

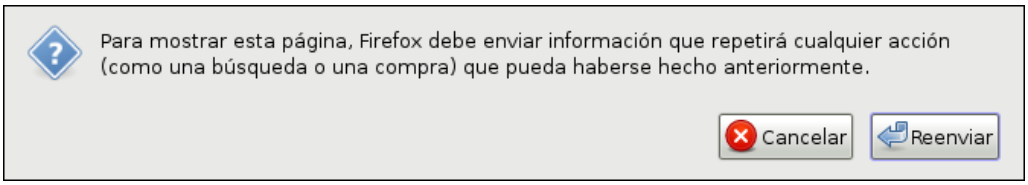

Figura 6.4: Problema del botón de retroceso

La mayoría de los frameworks de desarrollo de aplicaciones web no ofrecen una solución adecuada para modelar el flujo de información en un nivel elevado de abstracción. Por esta razón, los programadores se ven obligados a suplir en forma manual las carencias del protocolo *HTTP*. Y si bien, algunos entornos de desarrollo posibilitan la creación de aplicaciones a partir de componentes, no solucionan el problema del manejo del flujo de información a alto nivel.

Seaside introduce una capa de abstracción sobre el protocolo asincrónico de interacción para controlar el flujo de información, haciendo uso de las capacidades reflexivas de Smalltalk. Este mecanismo, basado en *"Continuations"* permite modelar el control de flujo de información entre varias páginas como una unidad e incluso permite manejar múltiples flujos de control activos dentro de la misma página, en forma simultánea.

Por ejemplo, en el proceso de captura de datos de Infosem Web, los participantes del experimento ingresan sus estimaciones sobre la distancia semántica entre pares de conceptos. Estos pares de conceptos, generados aleatoriamente, son presentados de a uno al participante, teniendo éste la posibilidad de modificar sus estimaciones anteriores, si así lo cree necesario.

En la figura 6.5 se observa que el participante cuenta con dos botones, uno de avance *(Siguiente)* y otro de retroceso *(Anterior)*, para situarse en un par de conceptos determinado y establecer el valor de la estimación por medio de la barra de desplazamiento (slider).

La utilización de una barra de desplazamiento (slider) permite al usuario expresar su estimación de proximidad entre conceptos de una forma más intuitiva. Ya que sólo necesita deslizar el cursor hacia la zona de mayor o menor proximidad, sin tener que preocuparse de su representación numérica.

En todo momento, Seaside mantiene el estado de la sesión del usuario, y cada acción del mismo forma parte de dicha sesión. Es decir, la aplicación conoce en cada momento, no sólo cuál es el par de conceptos actual, sino también cuál es el anterior y el posterior. De esta forma, no importa si el participante utiliza los botones de avance o retroceso de la aplicación para

Figura 6.5: Captura de estimaciones de proximidad

posicionarse en un par determinado, o si avanza o retrocede la página por medio de los controles del navegador. En todos los casos, la página generada presentará la información correcta. Es decir, no se producirá el *"back button problem"*.

6.4. Programación y depuración remota

La mayoría de los frameworks de desarrollo de aplicaciones dinámicas de Internet, proveen un escaso soporte para las tareas de detección de errores y depuración. La información proporcionada suele ser apenas un número de línea y los valores de la pila de ejecución desde el momento en que se produjo el error. Esta escasa información conduce a un proceso difícil y tedioso, que suele requerir a menudo, de la introducción de código adicional para localizar el origen del problema.

Seaside aprovecha las ventajas de la programación interactiva e incremental presentes en Smalltalk, ya que hereda la mayoría de las capacidades dinámicas y *"reflexivas"* presentes en su entorno. Esto le permite acceder a un conjunto de ventajas adicionales que no poseen otros frameworks de desarrollo de aplicaciones Web, como es la posibilidad de acceder a herramientas de programación y depuración en forma dinámica, es decir, con la aplicación funcionando.

Las aplicaciones construidas con Seaside poseen dos modos de ejecución. Uno denominado, de *"distribución"* (deployment), que es el utilizado cuando la aplicación ya se encuentra lo suficientemente probada y se halla corriendo prácticamente sin modificaciones. Y un modo denominado de *"desarrollo"*

(development), que es el utilizado en la fase de construcción y prueba del sistema. La principal ventaja de este modo, es que permite la realización de cambios y depuración de errores, en forma completamente remota.

En Smalltalk, cuando un objeto envía un mensaje a otro objeto, y éste no es capaz de responder adecuadamente, el objeto iniciador de la acción recibe como respuesta un mensaje *doesNotUnderstand* [4] y se abre una ventana del depurador (Debugger). En Seaside, sucede algo similar, ya que cuando un objeto no puede responder a un mensaje, se genera automáticamente una página como la que se aprecia en la figura 6.6, con toda la información necesaria para detectar el problema.

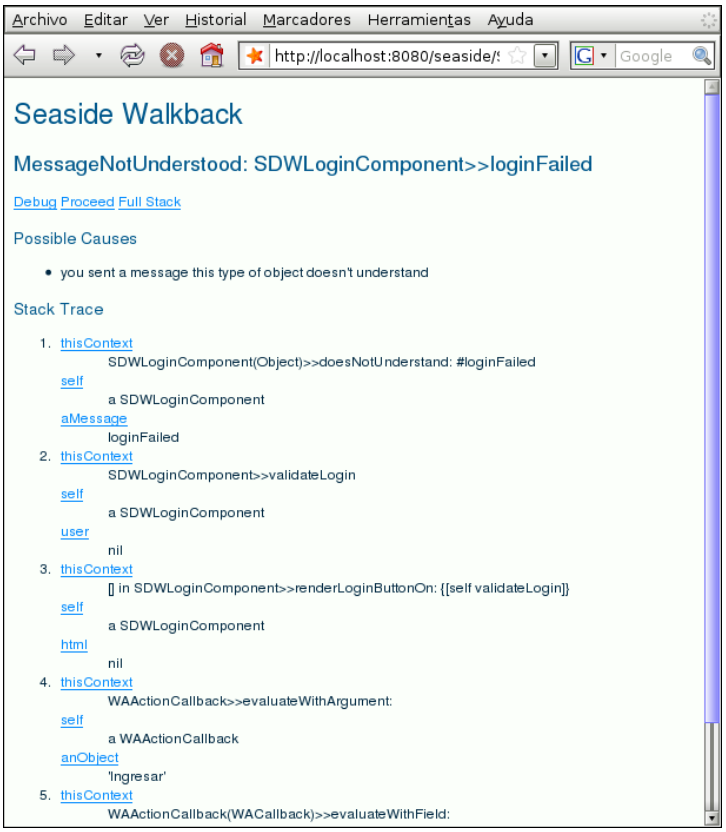

Figura 6.6: Acceso remoto a la ventana de depuración.

A modo de ejemplo, generaremos deliberadamente un error en nuestra aplicación, y analizaremos más detalladamente las ventajas de este mecanismo.

[4] "Does not understand" significa literalmente "No se entiende". Y es la forma en la que un objeto Smalltalk, le informa a quien le envía el mensaje, que no lo entiende. O dicho más formalmente, que dicho mensaje no forma parte del protocolo del receptor.

Como hemos mencionado, para ingresar las estimaciones, es preciso que los participantes del experimento se hallen registrados previamente. Si un participante pretende acceder al sistema sin estar debidamente registrado, debería recibir un mensaje de advertencia. Para ejemplificar el proceso de depuración remota, vamos a suponer que en el proceso de implementación, nos hemos olvidado de crear el método encargado de advertir al usuario. Por lo tanto, al producirse dicha condición, el objeto responsable de tal comportamiento, enviará un mensaje *"DoesNotUnderstand"*.

Cuando una situación de este tipo se produce, se dispara una excepción. Las excepciones en Smalltalk son objetos que referencian al contexto original de ejecución en el cual se produjo dicha excepción. De esta forma, se guarda la información correspondiente a la excepción producida y se invoca al depurador (debugger).

Esta situación anómala, que podría resultar catastrófica en otros entornos, en Seaside es rápidamente subsanable, incluso en forma remota. La ventana del depurador, nos presenta toda la información necesaria para solucionar el problema, (ver figura 6.6). Allí podemos observar, que el problema consiste en que el componente *SDWLoginComponent* no reconoce el mensaje *loginFailed* y esto se debe a que aún no lo hemos programado. Por lo tanto, es un buen momento para hacerlo.

Cuando la aplicación se halla corriendo en modo de *"desarrollo"*, es posible activar los *"halos"* de los componentes. Estos "halos" enmarcan el componente y le agregan tres íconos, como se observa en la figura 6.7.

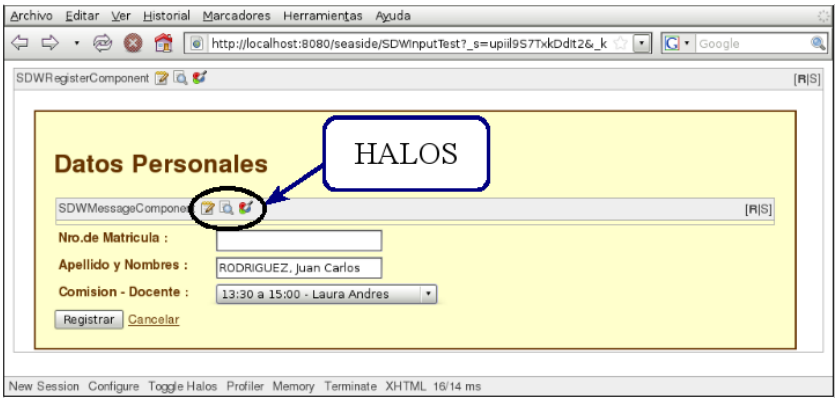

Figura 6.7: Halos en Seaside

Estos íconos nos permiten acceder en forma remota a tres herramientas muy importantes. Un *"inspector"*, con el que podemos conocer el estado interno del componente, un *"explorador de Clases" (Class browser)* y un *"editor de hojas de estilo" (CSS editor)*.

El *"explorador de Clases" (Class browser)* es una de las herramientas más importantes de Smalltalk, ya que con ella es posible recorrer toda la jerarquía de clases del sistema, crear nuevas clases y agregar, editar o eliminar métodos a las mismas. Es decir, es la herramienta de programación por excelencia.

Por lo tanto, presionando en el ícono correspondiente, podemos acceder en forma remota al *"explorador de Clases" (Class browser)*, como se observa en la figura 6.8.

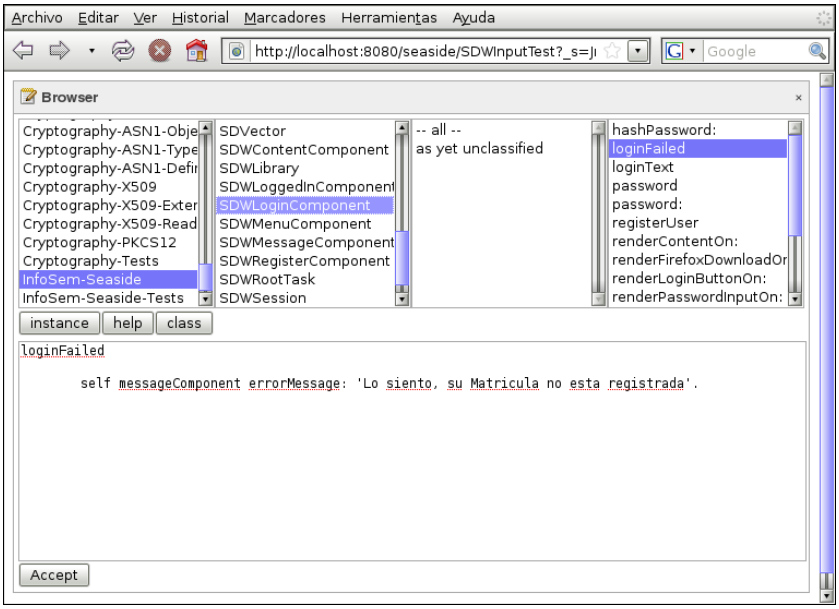

Figura 6.8: Acceso remoto al Class Browser

Seaside brinda un acceso transparente al "Class browser", como si se estuviera utilizando un entorno Smalltalk en forma local. De esta forma, es posible crear, en forma remota, el método faltante *(loginFailed)* y grabar los cambios. Automáticamente, la aplicación exhibirá su nuevo comportamiento y el error desaparecerá.

Este proceso se denomina "recompilación en caliente" (*hot recompilation*) o "recompilación de métodos al vuelo" (*on the fly method recompilation*), esto permite recompilar el método mientras la aplicación sigue corriendo, sin necesidad de instalar una nueva versión, ni de de reiniciar la sesión. Ni siquiera ha sido necesario detener su ejecución. Algo realmente impensable en otras plataformas de desarrollo.

Para poder apreciar las ventajas de este mecanismo, es preciso hacer una comparación con lo que hubiera sucedido en un entorno tradicional. En ese caso, al producirse un error, la aplicación hubiera dejado de funcionar. Luego, una vez detectado el problema, se debería corregir y recompilar toda

la aplicación, es decir, aún aquellas partes que no fueron modificadas. Una vez hecho esto, habría que sustituir la versión existente en el servidor, por la nueva versión corregida y por último, habría que iniciarla nuevamente. Un proceso mucho mas tedioso y lento, sobre todo, en la etapa de puesta a punto del sistema.

En este capítulo, se han presentado algunas de las características más distintivas de Seaside, tomando como ejemplo, una implementación concreta *(InfosemWeb)*. Aún quedan muchas otras por mencionar, como la persistencia de los datos o el manejo de concurrencia, pero el tratamiento de estos temas excedería los alcances del presente trabajo.

Capítulo 7

Conclusiones

En el presente trabajo se analizaron las características de las aplicaciones dinámicas de Internet y se presentó un enfoque diferente para su implementación. Nos referimos a *Seaside*, un framework de desarrollo, con características técnicas muy particulares que lo diferencian de los entornos tradicionales, además de ser una alternativa nueva, y muy poco conocida fuera de los ambientes de programación Smalltalk.

Seaside surge como un proyecto en contra de la corriente, con ideas innovadoras y hasta contrarias al "status quo". Tal vez, eso se deba a que *Seaside* surgió de un entorno muy particular llamado *Squeak*[1], que siempre priorizó la experimentación y el descubrimiento de nuevas alternativas.

Y eso no es casual, en *Squeak*, se fusionaron los conceptos provenientes de *Smalltalk-80*, *LOGO* [2] y *Self*[3] y se plasmaron las ideas de Alan Kay, Dan Ingalls, Adele Goldberg y Seymour Papert, entre otros. Con la finalidad de crear, no solamente un nuevo lenguaje de programación, sino un medio para expresar ideas, mas cercano a los seres humanos, que a las computadoras.

La potencia de *Squeak* para desarrollar aplicaciones educativas quedó demostrada en los proyectos desarrollados por SqueakLand (www.squeakland.org) y Squeakpolis (squeak.educarex.es/Squeakpolis), ahora *Seaside* brinda la posibilidad de llevar esa potencia a la web.

Seaside es un proyecto nuevo, y probablemente aún no estén dadas las

[1]Squeak es el sucesor de Smalltalk-80, y actualmente se distribuye bajo licencia del MIT.

[2]LOGO es un lenguaje de programación creado por Seymour Papert. Se trata de un lenguaje de alto nivel, en parte funcional y en parte estructurado, de muy fácil aprendizaje, razón por la cual suele ser el lenguaje de programación preferido para trabajar con niños y jóvenes.

[3]Self es un lenguaje de programación orientada a objetos, basado en el concepto de prototipos. Fue desarrollado por David Ungar y Randall Smith, en los laboratorios de XeroX PARC.

condiciones para que se transforme en una alternativa masiva. En la actualidad, existen varios ejemplos exitosos de aplicaciones dinámicas de Internet desarrolladas con *Seaside*, en funcionamiento. Sitios, como Dabble DB (www.dabbledb.com)[4], reserveTravel (www.reservetravel.com), US Medical Record Specialists (www.usmedrec.com), etc. Sin embargo, la mayoría de ellos son emprendimientos comerciales. Es decir, más allá de algunos proyectos de investigación o de ciertas iniciativas mixtas como Run BASIC (www.runbasic.com), aún no existen aplicaciones educativas, desarrolladas en *Seaside*, que hayan alcanzado una difusión masiva.

Esto abre un amplio campo de posibilidades para un próximo trabajo, no sólo en lo que respecta a la utilización de *Seaside*, sino también de otros proyectos relacionados. Es de destacar, que si bien *Seaside* es un excelente punto de partida para desarrollar aplicaciones educativas, el desarrollo de las mismas requiere de conocimientos de programación orientada a objetos en general y de Smalltalk en particular. Por esta razón, recientemente han surgido otras iniciativas como *Magritte*, *Pier*, *Aida/Web* y *Scribo*, tendientes a facilitar enormemente esta tarea.

Si consideramos que la mayoría de las aplicaciones están constituidas por una gran cantidad de objetos que modelan el dominio, para interactuar con dichos objetos es necesario crear diferentes vistas, editores, reportes, consultas, etc., lo que representa una tarea tediosa, repetitiva y propensa a generar errores. Por esta razón, Lukas Renggli ha creado *Magritte*, un framework meta-descriptivo totalmente dinámico, que se integra prefectamente con *Seaside* para resolver este problema.

Otra de sus creaciones es un sistema de gestión de contenidos *(CMS)* [5], denominado *Pier*, basado íntegramente en *Magritte*. De esta forma, mediante un lenguaje sencillo y acotado, es posible utilizar *Pier* para crear aplicaciones dinámicas y manejar sus contenidos con extrema facilidad. Estas nuevas herramientas, abren un sinnúmero de posibilidades para la creación de aplicaciones con contenidos educativos, ya que simplifican enormemente las tareas de implementación y mantenimiento.

Por último, es importante destacar que el crecimiento explosivo de las aplicaciones dinámicas de Internet, ha incrementado notablemente en el último tiempo la difusión de *Seaside*, incluso fuera del ámbito de *Smalltalk*. Esto ha creado una importante sinergia, en la que tanto *Seaside* como *Smalltalk* se potencian mutuamente.

[4]Creado por Avi Bryant, el creador de Seaside.

[5]Un Sistema de gestión de contenidos (Content Management System) es un programa que permite la creación y administración de contenidos en páginas web.

Bibliografía

[1] B. J. Allen-Conn and Kim Rose. *Powerful Ideas in the Classroom.* Viewpoints Research Institute, Inc., August 2003.

[2] I.I. Bejar, R. Chaffin, and S. Embretson. A taxonomy of semantic relations. *Cognitive and psychometric analysis of analogical problem solving.*, pages 56–91, 1991.

[3] Tim Berners-Lee. Information management: A proposal. http://www.w3.org/History/1989/proposal.html, 1989.

[4] Jorge Luis Borges. *Ficciones.* Alianza, 2006.

[5] Diego Gómez Deck. *Programando con Smalltalk.* GuadaLinex. EditLin, 1ra. edition, 2005.

[6] Diego Gómez Deck and José L. Redrejo Rodriguez. Squeak in spain as part of the linex project. In *C5 '04: Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing*, pages 160–165, Washington, DC, USA, 2004. IEEE Computer Society.

[7] Georg Heeg eK. Seabreeze - an editing tool for seaside. vst.ensm-douai.fr/Esug2008Media/uploads/1/seaBreeze.pdf, 2008.

[8] Marcela Prieto Ferraro, Dra. Begoña Gros Salvat, Dr. Francisco José, and García Peñalvo. "modelos para la elaboración de materiales hipermedia adaptativos para el aprendizaje", 2003.

[9] Marcio Gallio, Roger Soares, and Ian Oeschger. Inner-browsing: Extending web browsing the navigation paradigm. http://devedge-temp.mozilla.org/viewsource/2003/inner-browsing/index-en.html, 2003.

[10] Jesse James Garrett. Ajax: A new approach to web applications, February 2005.

[11] Adele Goldberg. Why smalltalk? *Commun. ACM*, 38(10):105–107, 1995.

[12] Adele Goldberg and David Robson. *Smalltalk. The Language.* Addison-Wesley, 1989.

[13] Begoña Gros Salvat. *El Ordenador Invisible.* Editorial GEDISA, 2000.

[14] C. Huapaya, F. Lizarralde, and G. Arona. Una propuesta para el diagnostico del estudiante en infosem. *II Congreso de Tecnología en Educación y Educación en Tecnología.*, pages 220–227, 2007.

[15] Dan Ingalls. Squeak - a usable smalltalk written in itself. http://wiki.squeak.org/squeak/1985, 1996.

[16] D.H.H. Ingalls. Design principles behind smalltalk. *Byte*, 6(8):286–298, 1981.

[17] Alan C. Kay. A personal computer for children of all ages. XeroX Palo Alto Research Center, 1972.

[18] Wilf R. Lalonde and John R. Pugh. *Inside Smalltalk.* Prentice Hall, 1990.

[19] John W. Maxwell. Tracing the dynabook: A study of technocultural transformations., 2006. Doctor of Philosophy Thesis. University of British Columbia.

[20] Rob McCool. The common gateway interface. http://hoohoo.ncsa.uiuc.edu/cgi/, 1995.

[21] Florian Moritz. Rich internet applications (ria). a convergence of user interface paradigms of web and desktop, 2008. Diploma Thesis. University of Applied Science Kaiserslautern.

[22] Tim O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software. *O'Reilly (http://www.oreilly.com/)*, September 2005.

[23] Seymour Papert. *Mindstorms: Children, Computers and Powerful Ideas.* Basic Books, 1980.

[24] Michael Perscheid, David Tibbe, Martin Beck, Stefan Berger, Peter Osburg, Jeff Eastman, Michael Haupt, and Robert Hirschfeld. *An Introduction to Seaside.* Software Architecture Group (Hasso-Plattner-Institut), 2008.

[25] Marina Polo. El diseño instruccional y las tecnologías de la información y la comunicación. *Docencia Universitaria, Vol II, No 2 SADPRO - UCV*, 2001.

[26] Prieto, Leighton, García, and Gros. Metodología para diseñar la adaptación de la presentación de contenidos en sistemas hipermedia adaptativos basados en estilos de aprendizaje. *Teoría de la Educación: Educación y Cultura en la Sociedad de la Información*, 6 (2), October 2005.

[27] Lukas Renggli. Seaside – the heretic web framework. Agile-Swiss, Geneva, Switzerland. http://www.lukas-renggli.ch/smalltalk/seaside/heretic.pdf, April 2007.

[28] Bob Ryan. Dynabook revisited with alan kay. http://www.squeakland.org/resources/articles/article.jsp?id=1007, 1991.

[29] Ryan Stewart. Rias: Rich learning for higher education. *EDUCAUSE Review*, 43(2):68–69, March/April 2008.

[30] J. Vivas. Método distsem: Procedimiento para la evaluación de distancias semánticas. *Memorias de las XI Jornadas de Investigación . Psicología, Sociedad y Cultura.*, pages 321–322, July 2004.

[31] J. Vivas, C. Huapaya, F. Lizarralde, G. Arona, A. Comesaña, L. Vivas, A. García Coni, and M González. *Instrumentos para la evaluación de la Memoria Semántica. Método y aplicaciones.* 2007.

[32] S. Wasserman and K. Faust. Social network analysis. *Methods and Applications*, 1998.

Índice de figuras