INGENIERÍA DE SOFTWARE E INGENIERÍA DEL CONOCIMIENTO: DOS DISCIPLINAS INTERRELACIONADAS

UNIVERSIDAD DE MEDELLIN

SELLO EDITORIAL
UNIVERSIDAD DE MEDELLÍN

INGENIERÍA DE SOFTWARE E INGENIERÍA DEL CONOCIMIENTO: DOS DISCIPLINAS INTERRELACIONADAS

1a. edición: 2014

ISBN: 978-958-8815-31-2

Catalogación bibliográfica - Universidad de Medellín. Biblioteca Eduardo Fernández Botero. María Isabel Quintero Bedoya.

Editor:
Leonardo David López Escobar
Dirección electrónica: ldlopez@udem.edu.co
Universidad de Medellín. Medellín, Colombia
Cra. 87 No. 30-65. Bloque 20, piso 2.
Teléfonos: 340 52 42 - 340 53 35
Medellín - Colombia

Corrección de estilo:

Diseño portada:
Claudia Castrillón Álvarez
claudiadisenografico@gmail.com

Distribución y ventas:
Universidad de Medellín
e-mail: selloeditorial@udem.edu.co; www.udem.edu.co
Cra. 87 No. 30-65 / Teléfono: 340 52 42 - Medellín, Colombia

Diagramación:
Hernán D. Durango T.
hernandedurango@gmail.com

Impresión: Xpress Estudio Gráfico y Digital S.A.
Av. Américas No. 39-53 / PBX (+57 1) 602 0808 - Bogotá, Colombia

Capítulo XVI

Modelo de servidor de proximidad especializado en usabilidad para aplicaciones web

Hernán Merlino
Oscar Dieste
Patricia Pesado
Ramón García-Martínez

INTRODUCCIÓN

Como se demuestra en el trabajo previo, es posible utilizar un modelo para apoyar la funcionalidad de UNDO/REDO bajo la modalidad de Software as a Service (SaaS). [1-4]. Los patrones de usabilidad se concibieron con el objetivo de hacer del desarrollo de software algo simple y predecible [5]. En el desarrollo de sistemas, los requisitos generales de usabilidad se incluyen en una fase avanzada del desarrollo del sistema [6], cuando hay poco tiempo de desarrollo y las decisiones clave de diseño ya se tomaron.

El proceso evolutivo por el cual se llegó a la construcción de este modelo se inicia con la experiencia adquirida a partir de un proceso *ad hoc* y luego evoluciona a un modelo patrón que especifica un conjunto de las mejores prácticas validadas que se pueden utilizar en otros diseños. Con este conocimiento, los diseñadores generan soluciones más complejas en menos tiempo, lo que conduce a definir arquitecturas. Desde este punto de la evolución de SaaS, el modelo es un paso natural [3]. Se ha seleccionado

Programa de Doctorado en Ciencias Informáticas. Universidad Nacional de La Plata
Grupo de Investigación en Sistemas de Información. Universidad Nacional de Lanús.
Grupo de Ingeniería de Software Empírica. Universidad Politécnica de Madrid.
Instituto de Investigaciones en Informática LIDI. Facultad de Informática. UNLP-CIC
hmerlino@gmail.com, rgarcia@.unla.edu.ar

la función de usabilidad UNDO/REDO se ha justificado en [4] y sistemas candidatos para este modelo se definen en [2]. Este documento se centra en el Servidor Proxy para integrar todos los requisitos de usabilidad a través del modelo SaaS.

Este capítulo se organiza de la siguiente manera: se presenta el estado de la cuestión (sección 2), se centra el problema de modelo de servidor de proximidad especializado en usabilidad para aplicaciones web (sección 3), se propone una solución (sección 4) consistente en un modelo de arquitectura compuesta por un módulo de detección (sección 4.1), un módulo de seguridad (sección 4.2), un módulo de administración (sección 4.3), un módulo de traducción (sección 4.4), un módulo de trazabilidad (sección 4.5), y un módulo de optimización (sección 4.6); se proporciona un ejemplo (sección 5) y se presentan conclusiones parciales de la línea de investigación (sección 6).

16.1 ESTADO DE LA CUESTIÓN

UNDO/REDO es una característica muy extendida y es importante en toda la gama de editores gráficos o de texto, como por ejemplo, los procesadores de texto, las hojas de cálculo, los editores gráficos, etc. Como es natural una gran parte del trabajo relacionado con UNDO/REDO se ha centrado en una u otra es estas aplicaciones. A modo de ejemplo se puede citar a [7] y [8] que han patentado dos métodos para la funcionalidad UNDO/REDO en editores de documentos en entornos de usuario único.

Existen soluciones específicas para los editores de texto para trabajo en grupo que soportan la funcionalidad UNDO/REDO como en [9] y en [10-11]. La razón para el auge de trabajos relacionados a la funcionalidad UNDO/REDO en el contexto de los editores de texto es su relativa sencillez.

Los problemas de UNDO/REDO en entornos multiusuario también atraen una atención significativa, Abrams y Oppenheim [12] proponen mecanismos para el uso de UNDO/REDO en entornos distribuidos, Abowd y Dix [13] proponen un marco de descripción formal y Qin y Sun [14] proponen la arquitectura en tiempo real para los sistemas de colaboración. En los entornos distribuidos, la solución tiene que lidiar con la complejidad de los cambios a los datos, esto en términos generales se soluciona con un archivo de historial de cambios [15].

Un grupo de investigadores han desarrollado el concepto de patrones de usabilidad basado en la experiencia adquirida en el manejo de la funciona-

lidad [16]. En [5-6] se define un modelo en el que se incluye la usabilidad como un elemento a ser considerado desde el inicio del desarrollo.

En [17] se aborda el problema de la usabilidad del software durante el desarrollo y los detalles de un proceso de obtención requisitos relacionados con la facilidad de uso. En el preámbulo de este trabajo también se da un conjunto de medidas para evaluar la usabilidad de un artefacto de software, a saber: (a) facilidad de aprendizaje, (b) eficiencia, (c) fiabilidad y (d) satisfacción.

En [1] se dan las pautas a tener en cuenta para la integración de las prácticas de usabilidad dentro de un proceso de ingeniería de software, incluyendo los pasos detallados para la evaluación de las características de usabilidad que se desea incluir.

Varios trabajos han arrojado luz sobre los aspectos internos de UNDO/REDO, como en [18] donde se trató de describir el proceso de UNDO/REDO según sus características.

Otro aspecto importante que se viene trabajando es el método de representación de las acciones realizadas por los usuarios, en [19] se presentando una estructura dinámica de comandos que representa la historia de comandos implementados.

Se han registrado patentes como el método para la construcción de un proceso UNDO/REDO [20], a su vez [21] definen un mecanismo para la gestión de un UNDO/REDO de múltiples niveles.

El mayor problema con el trabajo previo es que, una vez más, son difíciles de adoptar en los procesos de desarrollo de software fuera del dominio editor de documentos. La única excepción notable a esto es un mecanismo a nivel de diseño llamado Memento [22].

Por otra parte, la utilización de servicios de software para la construcción de arquitecturas de sistema es una estrategia cada vez más utilizada por diferentes empresas [23], en tal sentido es importante definir sus principales características [24]: (a) los servicios son autónomos y modulares (b) existen servicios de integración, (c) los servicios están débilmente acoplados, (d) los métodos de localización son transparentes, (e) los servicios son módulos compuestos de diversos componentes.

En la bibliografía referida a la infraestructura como un servicio (IaaS), se ha propuesto el uso de un servidor de proximidad (Proxy Server) [25], que

proporciona una capa de abstracción entre las interfaces de programación (API) publicadas por cada proveedor de IaaS y el cliente que consume este servicio, esta capa permite que las invocaciones del cliente sean siempre las mismas para los diversos proveedores, evitando así la integración para cada proveedor de servicio IaaS, esta capa actúa como traductor. Esta propuesta es innovadora y permite a los consumidores de servicios pueden centrarse en la orquestación de sus sistemas y la complejidad de la comprensión de la interfaz propuesta por cada quede limitada el servidor de proximidad.

16.2 PROBLEMA

Las capas de una aplicación para el mantenimiento y la capacidad de adaptación constituyen un tema central en la literatura de la ingeniería de software, según se detalla en el trabajo [22] y [25], por otra parte la usabilidad es un aspecto que tiende a quedar para las etapas posteriores del desarrollo, por lo general debido al esquema de tiempo ajustado con el que cuenta el diseñador, diseñador como se ha definido en [6]. También se ha demostrado que es posible realizar un modelo de SaaS para la funcionalidad de UNDO/REDO [1-4].

En determinadas implementaciones el uso de servicios de software en aplicaciones desarrolladas para ambientes Web, puede generar problemas de seguridad pues al invocar un servicio donde el dominio es diferente de donde se arranco la aplicación puede obtenerse una alerta de seguridad. Por otra parte en el caso de proliferar los proveedores de servicios de software especializados en usabilidad, un arquitecto de sistemas se enfrentaría a la misma situación que la descripta para IaaS.

16.3 SOLUCIÓN PROPUESTA

Al generar un mayor nivel de abstracción entre la aplicación principal y la usabilidad del sistema mediante un modelo de servidor *proxy*, se intenta simplificar la construcción de software, permitiendo la integración de diversos servicios, internos y externos, para la construcción de aplicaciones. Esta es una alternativa válida para aplicaciones legadas, pues permite utilizar actualizar esas aplicaciones sin la necesidad de modificaciones en ellas.

El servidor *proxy* de usabilidad provee las siguientes funciones: (a) Módulo de detección, (b) Módulo de Seguridad, (c) Módulo de Gestión, (d) Módulo de Traducción, (e) Módulo de Trazabilidad, y (f) Módulo de Optimización.

16.3.1 Módulo de Detección

Este módulo es responsable de reconocer automáticamente qué dispositivo se conecta con el servidor para seleccionar la interfaz de usuario adecuada. Esto permite a los diseñadores de la interfaz desarrollar un conjunto heterogéneo de interfaces y catalogar en el servidor, que reconoce al usuario que se conecta y, en función de las características establecidas de interfaces catalogadas, interactúa con el dispositivo del usuario, según corresponda.

Para la catalogación se utiliza una estructura de árbol jerárquico, en donde el nodo raíz es un modelo de interfaz genérica, la cual permite que el *proxy* entregue un conjunto de interfaces en la ausencia de un conjunto definido para ese dispositivo.

La estructura de árbol, a su vez, permite interfaces híbridas, por ejemplo, integrar una aplicación móvil nativa para un dispositivo específico con un conjunto de páginas almacenadas en el servidor. Esto proporciona la ventaja de hacer que las aplicaciones móviles sean más simples, pues no es necesario especializarse cada una de las interfaces de diferentes dispositivos móviles.

16.3.2 Módulo de Seguridad

Este módulo permite resolver el problema de seguridad relacionado con el acceso a las diferentes plataformas que conforman la propia aplicación. Las credenciales de SaaS deben realizar la solicitud para acceder a cada uno de estos servicios. El servidor almacena las credenciales diferentes y las credenciales de usuario relacionadas, permitiendo así que el usuario tenga un único conjunto de credenciales que dan permiso a la aplicación. En este punto se suele utilizar una base de datos NoSQL, la base de datos Redis [26], lo que le permite almacenar y recuperar datos y establece un formato rápido de clave y valor.

16.3.3 Módulo de Administración

Aquí es donde se pueden configurar los servicios a los que se accede mediante el servidor de proximidad, además de almacenar las credenciales de cada uno. Por otra parte, es donde se crea el flujo de la aplicación, la secuencia de pantallas que verá el usuario, lo que permite crear una aplicación rápida y sencilla.

16.3.4 Modulo de Traducción

Este módulo se encarga de transferir solicitudes de los usuarios del servicio en cuestión y obtiene la respuesta a ellas, llevándolas al formato del dispositivo de acceso. La posibilidad de tener un módulo de traducción permite una gran flexibilidad respecto de la comunicación con los diferentes servicios, ya que éste crece según las necesidades.

16.3.5 Módulo de Trazabilidad

Este módulo se encarga de registrar todas las acciones que realiza el usuario y almacenarlas en un archivo de registro. Este archivo se utiliza en otro módulo para el análisis de la interfaz, para las mejoras consecuentes.

16.3.6 Módulo de Optimización

Este módulo interactúa con dos módulos de servidor, a saber: el módulo de detección y el módulo de trazabilidad. Del módulo de trazabilidad toma los archivos que detectan automáticamente a los usuarios que utilizan aplicaciones, con el fin de generar los grupos de afinidad. Esto se logra mediante la aplicación de un algoritmo de red neuronal, basado en el modelo de Kohonen [27], comúnmente conocido como SOM (*Self Organizing Maps*). La ventaja de utilizar un modelo de red neuronal de aprendizaje no supervisado es detectar grupos sin necesidad de conocer con antelación los estilos de trabajo y acceso de los usuarios.

Además, se usa la tecnología de inducción basada en árboles [28-29], para analizar las diferentes formas con las que el usuario tiene acceso a ciertas funciones, lo que deja el registro de trazabilidad, donde se registran todas las páginas donde el usuario pasó. Mediante el análisis de esta información, el módulo de optimización interactúa con la administración para crear accesos directos a las páginas por las que el usuario navega. Esto es posible debido a que el módulo de gestión es el flujo de la aplicación. Por lo tanto, es posible reducir el número de accesos que se envían al servidor para cada usuario, obteniéndose un grado de parametrización en los accesos a las diferentes partes de la aplicación.

16.4 EJEMPLO

La aplicación de ejemplo que se construyó se compone de una tecnología *Proxy Server* construida sobre NodeJS [30], que es responsable de recibir todas las solicitudes mediante la interfaz de usuario y dirigirla a la correcta.

Para la construcción de la interfaz de usuario se ha utilizado HTML 5 y AngulareJS [31], todo ello bajo un modelo de aplicación Web de pagina única [32].

Nuestro servidor de proximidad se comunica con dos aplicaciones: (a) una aplicación adecuada de la operación realizada, PHP [33], que utiliza como framework a CodeIgniter [34], (b) Servicio UNDO/REDO, el modo SaaS, este es externo a la red en la que la aplicación y el servidor de proximidad están alojado (Figura 1).

El servidor proxy direcciona las peticiones según corresponda a los requisitos de la aplicación o en el propio servicio. La aplicación es una aplicación en la que se carga un conjunto de profesores para los cursos de asignación (Figura 2).

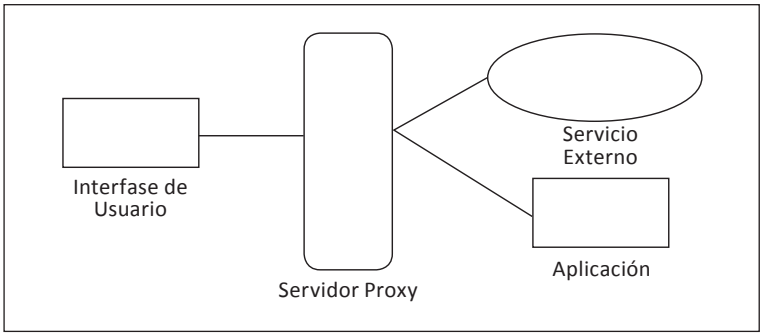

Fig. 1. Arquitectura de la Aplicación

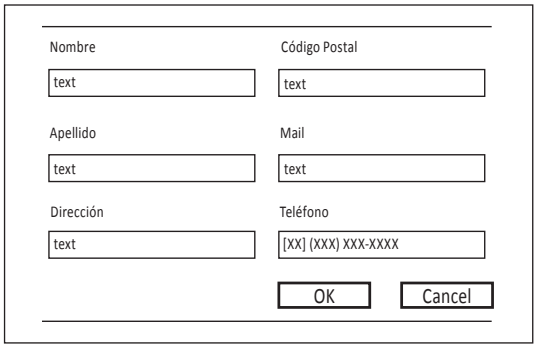

Fig. 2. Interfaz de la aplicación

A partir de este ejemplo, después de un mes de su puesta en funcionamiento, se detectaron las características que se anotan seguidamente.

El modelo SOM detecta dos tipos de usuarios claramente definidos, (a) sólo consultar la aplicación y (b) el uso de los nuevos maestros utilizados para cargar, esto permitió optimizar la seguridad mediante la reducción de los permisos de acceso para el primer grupo.

En algoritmo ID3, árbol de inducción, permitió la detección de un modo de navegación en los usuarios, por lo cual inserta en el menú principal de la aplicación un acceso directo a esta interfaz, lo que generó una reducción de las peticiones al servidor ya que los usuarios podrían acceder a lo más directamente a lo que necesitaban.

16.5 CONCLUSIONES

Un modelo de integración de características de uso del *proxy* es un modelo adecuado para reducir el estrés y simplificar el desarrollo de aplicaciones, permitiendo a los desarrolladores centrarse en las cuestiones fundamentales de la aplicación. También, se reduce la carga del servidor y se extiende su vida útil.

Como fruto de este proceso se ha demostrado que el uso de las nuevas tecnologías es un aspecto importante en la evolución del desarrollo de software en una empresa, este modelo de servidor de proximidad permite la integración de nuevas tecnologías de una manera controlada y sin la necesidad de grandes.

El siguiente objetivo de este equipo de investigación es añadir un módulo de distribución de carga para manejar las peticiones a diferentes servidores redundantes.

16.6 FINANCIAMIENTO

La investigación que se presenta en este Capítulo se financió parcialmente con las subvenciones UNLa-SCyT-33A105 y UNLa-SCyT-33B06 de la Universidad Nacional de Lanús (Argentina) y con las subvenciones TIN2008-00555 y HD2008-00046 del Ministerio de Ciencia e Innovación español (España).

REFERENCIAS BIBLIOGRÁFICAS

1. Merlino, H., Dieste, O., Pesado, H., García-Martínez, R.: Framework to Provide Highly Automated UNDO Capabilities on Software Systems. En Ingeniería de Software e Ingeniería del Conocimiento: Tendencias de Investigación e Innovación Tecnológica en Iberoamérica (Editores: R. Aguilar, J. Díaz, G.

Gómez, E- León), pp. 194-204. Alfaomega Grupo Editor. ISBN 978-607-70 7-096-2 (2010)

2. Merlino, H., Dieste, O., Pesado, H., García-Martínez, R.: Inclusion Process of UNDO/REDO Service in Host Applications. SOFTWARE ENGINEERING: METHODS, MODELING, AND TEACHING (LASES 2012), pp. 29-37. Edited by Pontificia Universidad Católica de Peru. ISBN N° 978-612-4057-84-7 (2012)

3. Merlino, H.; Dieste, O.; Pesado, P.; Garcia-Martinez, R.: Software as a Service: Undo. The 24 International Conference on Software Engineering & Knowledge Engineering (SEKE 2012), pp. 328-333. ISBN: 1-891706-31-4 (2012)

4. Merlino, H.; Dieste, O.; Pesado, P.; Garcia-Martinez, R.: Service Oreinted Architecture for Undo Functionality. 6th International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2012). http://www.confenis2012.be/ documenten/ 17MerlinoDiestePesadoRGM.pdf (2012)

5. Ferre, X; Juristo, N; and Moreno, A.: Framework for Integrating Usability Practices into the Software Process. Madrid Polit. University (2004)

6. Ferre, X., Juristo, N., Moreno, A., Sanchez, I.; A Software Architectural View of Usability Patterns. 2nd Workshop on Software and Usability Cross-Pollination (at INTERACT'03) Zurich, Suiza (2003)

7. Bates, C. and Ryan, M.: Method and system for UNDOing edits with selected portion of electronic documents. PN: 6.108.668 US (2000)

8. Baker, B. and Storisteanu, A.: Text edits system with enhanced UNDO user interface. PN: 6.185.591 US (2001)

9. Sun, C.: Undo any operation at time in group editors. School of Computing and Information Technology, Griffith University Australia (2000)

10. Chen, D; Sun, C.: Undoing Any Operation in Collaborative Graphics Editing Systems. School of Computing and Information Technology, Griffith University Australia (2001)

11. Yang, J; Gu, N; Wu, X.: A Document mark Based Method Supporting Group Undo. Department of Computing and Information Technology. Fudan University (2004)

12. Abrams, S. and Oppenheim, D.: Method and apparatus for combining UNDO and redo contexts in a distributed access environment. PN: 6.192.378 US (2001)

13. Abowd, G.; Dix, A.: Giving UNDO attention. University of York (1991)

14. Qin, X. and Sun, C.: Efficient Recovery algorithm in Real-Time and Fault-Tolerant. School of Computing and Information Technology Griffith University (2001)

15. Berlage, T; Genau, A.: From Undo to Multi-User Applications. German National Research Center for Computer Science. Collaborative Editing Systems. School of computing and Information Technology Griffith University Australia (1993)

16. Juristo, N; Lopez, M; Moreno, A; Sanchez, M.: Improving software usability through architectural patterns. ICSE'03-International Conference on Software Engineering (2003)

17. Juristo, N.; Moreno, A.; Sanchez-Segura, M.: Guidelines for eliciting usability functionalities, IEEE Transactions on Software Engineering, vol. 33, no. 11, pp. 744–758 (2007)

18. Mancini, R., Dix, A., Levialdi, S.: Reflections on UNDO. University of Rome (1996)

19. Washizaki, H; Fukazawa, Y.: Dynamic Hierarchical Undo Facility in a Fine-Grained Component Environment. Department of Information AND Computer Science, Waswda University. Japan (2002)

20. Keane, P. and Mitchell, K.: Method of and system for providing application programs with an UNDO/REDO function. PN:5.481.710 US (1996)

21. Nakajima, S., Wash, B.: Multiple levels UNDO/REDO mechanism. PN: 5.659.747 US (1997)

22. Gamma, E., R. Helm, R. Johnson, and J. Vlissides.: Design Patterns: Elements of Reusable Object-Oriented Software, Addison- Wesley (1994)

23. Binildas, CA; Malhar, Barai; Vincenzo, Caselli.: Service Oriented Architecture with Java. Packt Publishing, Birmingham – Mumbai (2008)

24. Endrei, M; Ang, J; Arsanjani, A; Chua, S; Comte, P; Krogdahl, P; Luo, L; Newling, T.: Patterns: Service-Oriented Architecture and Web Services. IBM, Redbooks (2004)

25. Shixing, Y.; Lee, D.; Singhal, S.: A Model-Based Proxy for Unified IaaS Management. Systems and Virtualization Management SVM, 2010 4th International DMTF Academic Alliance Workshop on, pp. 15–20. ISBN: 978-1-4244-9181-0. DOI: 10.1109/SVM.2010.5674747 (2010)

26, Redis.: http://redis.io. Valid Page: 2013-04-01 (2013)

27. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics Volume 43, Issue 1 , pp 59-69. Publisher Springer-Verlag. DOI: 10.1007/BF00337288, Print ISSN: 0340-1200, Online ISSN: 1432-0770 (1982)

28. Mingers, J.: An empirical comparison of selection measures for decision-tree induction. Machine Learning Volume 3, Issue 4 , pp 319-342. Kluwer Academic Publishers. DOI: 10.1007/BF00116837, Print ISSN: 0885-6125, Online ISSN: 1573-0565 (1989)

29. López De Mántaras, R.: A Distance-Based Attribute Selection Measure for Decision Tree Induction. Machine Learning Volume 6, Issue 1 , pp 81-92. Kluwer Academic Publishers-Plenum. DOI: 10.1023/A:1022694001379, Print ISSN: 0885-6125, Online ISSN: 1573-0565 (1991)

30. NojeJS.: http://nodejs.org. Valid Page: 2013-04-01 (2013)

31. AngularJS.: http://angularjs.org. Valid Page: 2013-04-01 (2013)

32. Mikowski, M & Powell, J.: Single Page Web Applications. Manning Publications Co. ISBN: 9781617290756 (2013)

33. PHP.: http://php.net. Valid Page: 2013-04-01 (2013)

34. CodeIgniter.: http://ellislab.com/codeigniter. Valid Page: 2013-04-01 (2013)

Fudan University.