INSTITUTO DE INVESTIGACIÓN BY DESARROLLO EN INFORMATICA AVANZADA
REPORTE TECNICO: IIDIA-RT-2009-01
14/07/09

Unidad Lógica de Cambio en el Mecanismo de Usabilidad UNDO

Hernán Merlino, Ramón García-Martínez, Oscar Dieste

Programa de Doctorado en Ciencias Informáticas. Facultad de Informática. UNLP
Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. UBA
Grupo de Ingeniería de Software Experimental. Facultad de Informática. UPM
hmerlino@itba.edu.ar, rgarciamar@fi.uba.ar, odieste@fi.upm.es

Resumen. El presente trabajo intenta proveer un mecanismo con un alto porcentaje de automatización para un proceso UNDO, el cual pueda ser fácilmente incluido en un sistema por hacer como en uno ya hecho. Se presenta el concepto de Unidades Lógicas de Cambio que es el concepto rector para la implementación de la propuesta de UNDO aquí presentada.

1. Introducción

Uno de los aspectos que abarca la ingeniería del software (IS) es la de desarrollar interfaces de usuarios para que los sistemas puedan interactuar con los seres humanos; esta área de IS se denomina interfaz humano computador (HCI, de sus siglas en inglés). La usabilidad de un producto ha sido definida en la norma ISO 9241 como, la característica que se le agrega a un producto determinado para que este pueda ser utilizado en forma efectiva, eficientemente y satisfactoria por parte de quien utilice este producto. Se puede definir el HCI como una ciencia aplicada la cual provee un conjunto de teorías y métodos para ser utilizados en el desarrollo de software [Carroll, 2003] Se puede definir a los patrones de usabilidad, como mecanismos que pueden ser aplicados en el diseño de sistemas para proporcionar una particularidad específica de la usabilidad [Ferre, *et al.*, 2003].
En este artículo se presenta el estado de la cuestión sobre el mecanismo de usabilidad UNDO (sección 2), se identifican algunos problemas en su construcción (sección 3), como solución de cara a estos problemas se introduce las unidades lógicas de cambio (sección 4) y se presenta un esbozo de la Dinámica de las Unidades Lógicas de cambio en el proceso UNDO (sección 5).

2. Estado de la cuestión

Centrándose en computadores personales con sistemas basados en un gestor de ventanas, con capacidades de multiprocesamiento se pueden detallar un conjunto de

características comunes a todos los sistemas; una de estas características comunes es la brindar al usuario la posibilidad de volver hacia atrás una tarea ejecutada.
En el dominio de HCI a este mecanismos de usabilidad se lo conoce con el nombre de proceso UNDO, que se traduce al español como deshacer, entendiéndose como acepción de deshacer; la tarea de volver al estado inmediatamente anterior a una acción dada.
Gran parte del trabajo que se ha llevado a cabo en HCI se ha centrado en modelos predictivos, los cuales se encargan de estimar el tiempo que le consumirá al usuario entender la interfaz sin tener un conocimiento previo de esta. [Stuart *et al*., 1986], también se han desarrollados modelos descriptivos, estos dan una descripción detallada del modelo de solución del problema [Carroll, 2003] también se han desarrollado métodos para predecir el impacto que tendrá sobre el diseño la inclusión de factores de usabilidad, este es el denominado GOMS (Goals, Operators, Methods, and Selection rules) [Stuart *et al*., 1986]; se han detallado modelos de interfaz para distintos dispositivos desde PC a teléfonos celulares, descripción de tipografías y gráficos [Cogswell, 2004]
Se han propuesto patrones para recobrar estados de sistemas en ambientes de colaboración [Qin, 2001] pero solo abarcan caso relacionados con esta problemática. Otros investigadores [Abowd y Dix, 1991] abordan el problema de los editores compartidos y el uso del UNDO desde el uso de métodos formales.
También se ha trabajado con las generalidades del proceso UNDO [Mancini *et al*, 1996] en este sentido existen patentes de varios métodos de UNDO como ser (a) método para incluir un proceso UNDO y redo en un sistema [Keane y Mitchell, 1996] (b) mecanismo para manejar un sistema UNDO redo multi nivel [Nakajima y Wash, 1997], (c) método para realizar un UNDO en un editor de documentos [Bates y Ryan, 2000], (d) un método para administrar gráficamente UNDO y redo [Martinez y Rhan, 2000], (e) editor de texto con interfase de UNDO [Baker y Storisteanu, 2001], (f) método para la utilización de UNDO redo en ambientes distribuidos [Abrams y Oppenheim, 2001], (g) UNDO selectivo [Korenshtein, 2001], (h) algoritmo para UNDO y redo [Li, 2006] ; e (i) infraestructura de UNDO [Burke, 2007] . Todas dan generalidades de UNDO sin definen implementaciones detalladas para determinadas situaciones. Berlage [1994] ha propuesto la construcción de un método UNDO en entornos gráficos basados en comandos.
A nivel de diseño de patrones se ha desarrollado un mecanismo llamado Memento que hace las veces de un mecanismo UNDO, pues este restaura un objeto al estado anterior.

3. Problemas del UNDO

Uno de los mayores problemas con que se enfrenta alguien que debe construir un proceso de UNDO es la de proveer un mecanismo que sea independiente de la implementación que se esta haciendo y que pueda ser fácilmente integrado en un sistema ya existente como en uno nuevo, vale aclarar, que el patrón Memento [Gamma *et al*., 1994] es un avance pues implementa un mecanismo que puede ser reutilizado pero los objetos a restaurar deben ser programados para cada caso, esto se

aplica con cierta facilidad para sistemas que se están construyendo, la implementación del mismo en sistemas ya construidos se torna mas compleja.
En este contexto, en un proceso UNDO la definición de la información susceptible a ser vuelta a un estado anterior es uno de los escollos a salvar para poder automatizar el proceso, una propuesta de solución a esto es el concepto que se presenta en este trabajo: Unidades Lógicas de Cambio, estas dan un mecanismo para reconocer la información que puede ser afectada por el UNDO.

4. Solución Propuesta: Unidad Lógica de Cambio

Un problema que se ha presentado durante el análisis del proceso UNDO es, la especificación de a que partes del sistema se les debe aplicar un proceso UNDO; este planteo surge del análisis hecho sobre la grilla de datos primeramente y se puede comprobar que esta problemática es afín a los demás circuitos evaluados. Una alternativa de solución a esto es la definición de Unidades Lógicas de Cambio, la cuales serán explicadas en detalle a través de un conjunto de ejemplos.
Centrándose en el ejemplo de la grilla, al hacerse necesario implementar un proceso de UNDO surge la siguiente pregunta, a que se le debe aplicar el proceso de UNDO, a las celdas en forma individual, a la grilla en forma global, o alguna combinación de estas; este cuestionamiento solo parece tener repuesta a nivel de requerimiento de usuario, es este el que debería definir en primera instancia como necesita que se aplique el proceso UNDO para la grilla en cuestión.
El problema que se encuentra aquí es que en muchos casos al usuario le es poco claro que es un procesos de UNDO y a que se le debe aplicar; para esto se ha definido el concepto de Unidad Lógica de Cambio la cual permite al diseñador de sistemas realizar un análisis del artefacto a implementar, en este caso la grilla, este genera una lista con todas las opciones que el reconoce que pueden ser sujetas a un proceso UNDO, esta lista es presentada nuevamente al usuario y este define cuales deben ser aplicadas; siguiendo con el ejemplo que estamos desarrollando la lista de las opciones a la que se le puede aplicar un proceso UNDO en una grilla son:

- A nivel de toda la grilla: cuando se selecciona volver a tras un valor de una celda determinada, no solo esa celda es restaurada al valor solicitado, sino que todas las celdas de la grilla son restauradas al momento que la celda seleccionada tenia ese valor.
- A nivel de celda: cuando se selecciona volver a tras un valor de una celda determinada, solo ese valor es restaurado, los demás valores de las otras celdas siguen sin cambio.
- A nivel de columna: cuando se selecciona volver a tras un valor de una celda determinada, no solo esa celda es restaurada al valor solicitado, sino que todas las celdas de la columna a la que pertenece la celda seleccionada.
- A nivel de fila: cuando se selecciona volver a tras un valor de una celda determinada, no solo esa celda es restaurada al valor solicitado, sino que todas las celdas de la fila a la que pertenece la celda seleccionada.

Estas son solo 4 de las opciones más comunes a la que se podría aplicar un proceso UNDO a una grilla, existen otras tales como la selección por rango de celdas, etc. Como se puede observar estas definiciones son mas propias de un diseñador que de un usuario, pero en ultima instancia el usuario es el que conoce que debe ser aplicado y que no, es por esto que estas 4 Unidades Lógicas de Cambio son presentadas al usuario y este define cuales son apropiadas para el sistema solicitado.
Es decir que se puede definir a la Unidad Lógica de Cambio como una unidad básica de un sistema a la cual se le puede aplicar un proceso UNDO y el mismo quedara en un estado estable.
Este concepto puede ser trasladado a otros casos, como por ejemplo un formulario para carga de datos, el alta de los mismos escapa al ámbito del UNDO y es propio de la base de datos, pero una vez que el registro ha sido dado de alta y es consultado por uno o mas usuarios, se presenta la situación en la que se desea volver a tras algún valor de los modificados durante la consulta, vale aclarar que el proceso UNDO solo es aplicable a la edición actual y no a ediciones de datos anteriores, una lista posible de Unidades Lógicas de Cambio podría ser:

- A nivel de Formulario: al seleccionar un campo del formulario y aplicar el proceso UNDO, este no solo es aplicado al campo en cuestión, sino que todos los demás campos del formulario le es aplicado el proceso UNDO.
- A nivel de Campo: al seleccionar un campo del formulario y aplicar el proceso UNDO, este solo es aplicado al campo en cuestión.

Estas son solo dos de las alternativas mas comunes, se podría contemplar la opción de aplicar el proceso UNDO a campos que están relacionados, es decir que si se intenta volver a un valor anterior un campo que esta relacionado con uno o mas campos, el cambio se aplica al conjunto de estos.
Otro caso al que puede ser aplicado este concepto es a un procesador de texto en el cual los documentos pueden ser compartidos en línea, un documento puede ser modificado por dos o mas personas al mismo tiempo, pero para que esto sea posible se deben definir las Unidades Lógicas de Cambio que permitirán definir cuales son las mínimas áreas de lock en las cuales podrán trabajar los usuarios que comparten el documento y estas a su vez servirán de base para implementar el proceso UNDO. Una lista tentativa puede ser:

- A nivel de Documento: en este caso el documento no podría ser compartido por mas de un usuario.
- A nivel de Párrafo: dos usuarios podrían modificar el mismo documento pero no 2. podrían modificar el mismo párrafo.
  A nivel de oración: dos usuarios podrían hacer modificaciones en el mismo párrafo pero no en la misma oración.

A un nivel de mayor granularidad no seria posible trabajar pues se perdería el sentido semántica de la oración. Se puede observar una correlación entre la Unidad Lógica de Cambio y el lock que se produce de la información para que una aplicación tenga consistencia y estabilidad. De lo antes mencionado se puede inferir que la implementación de un proceso UNDO es un ciclo iterativo entre el diseñador y el usuario.

5. Dinámica del Proceso UNDO con Unidades Lógicas de Cambio

La implementación de un proceso de UNDO en un sistema es una cuestión no trivial, se debe definir a que procesos se les debe aplicar el UNDO, a modo de ejemplo se puede tomar una grilla de datos, seria de esperar que el proceso de UNDO pueda volver a un momento dado cualquier valor de una celda, pero surgen otros cuestionamientos, es necesario, de estar trabajando en un entorno grafico, que se vuelva a la posición donde se encontraba el cursor y/o el puntero del ratón al momento al que se retrotrae el UNDO.
Una característica importante a tener en cuenta en un sistema al momento de implementar un proceso UNDO es si los dados susceptibles de ser aplicable un proceso UNDO son compartidos o no, esta característica es al margen que el sistema sea multiusuario. Esto hace que el proceso de UNDO a implementar tenga un alcance distinto que si no compartiera datos. Si se toma como ejemplo una grilla de datos y el usuario que trabaja con ella solo modifica una porción de datos que es consultada por el proceso de UNDO que se debe implementar es la sucesión de modificaciones que el mismo aplica en forma invertida y no se presenta ningún problema de resolución de incompatibilidad por la concurrencia con otro usuario, el ejemplo se puede observar en la figura 1.
En dicha figura se puede observar la implementación de un mecanismo sencillo de UNDO, en el cual entre la copia local que el usuario esta trabajando y la versión global de la grilla se intercala un mecanismo de filtro, el cual todas las modificaciones que el usuario realice se almacenan en una cola local de modificaciones. Cuando el usuario solicita una acción de UNDO sobre los datos de su grilla local las modificaciones hechas son tomadas de la cola local y presentadas, el cual selecciona y el proceso se vuelve a repetir, el ejemplo se puede observar en la figura 2.

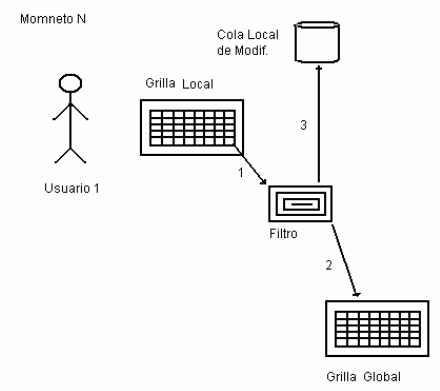

Fig. 1. Carga

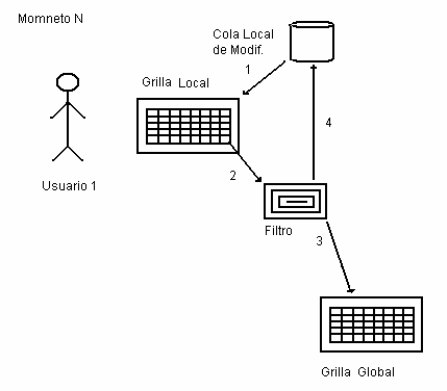

Fig. 2. Selección

Por otra parte en ambientes multiusuarios donde los datos son compartidos por los usuarios al mismo tiempo, una variable a manejar es la apreciación del usuario de lo que debe entregar la acción de ejecutar un proceso UNDO, si sé esta trabajando sobre una base de datos y se están modificando registros de la misma y dos usuarios están

trabajando sobre el mismo registro si el usuario 1 en el momento n modifica el valor del registro, a su vez el usuario 2 en el momento n + 1 modifica el mismo registro, seguido a esto el usuario 1 en el momento n + 2 desea volver al valor anterior, el concepto de valor anterior que entiende el usuario 1 es el valor anterior que el vio es decir n – 3 unidades de tiempo hacia atrás, aclaración, en este punto no hace referencia a mecanismos que provee una base de datos, solo se hace referencia a la percepción por parte del usuario sobre lo que debe hacer el proceso UNDO, figura 3.

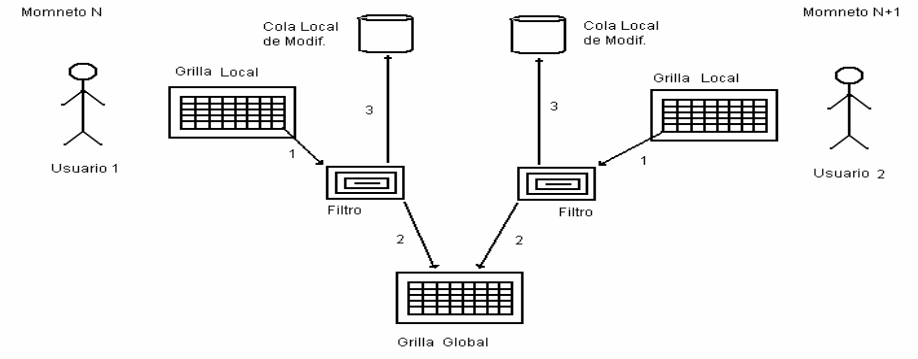

Fig. 3. Carga multiusuario

En la figura 1 se detalla el proceso UNDO, en el momento n el usuario 1 modifica la grilla, su versión local de la misma, en una implementación simple se podría agregar un filtro donde toda información que se halla enviado a la grilla global y sea producto de una modificación también se agregue a una copia local de la cola de modificaciones, esta servirá para implementar el proceso de UNDO, en el momento n+1 el usuario 2 realiza otra modificación sobre la grilla y el proceso se repetirá, al finalizar se obtendrá una nieva versión de la grilla global y dos copias globales distintas. A renglón seguido el usuario 1 desea ejecutar un proceso UNDO, figura 4.

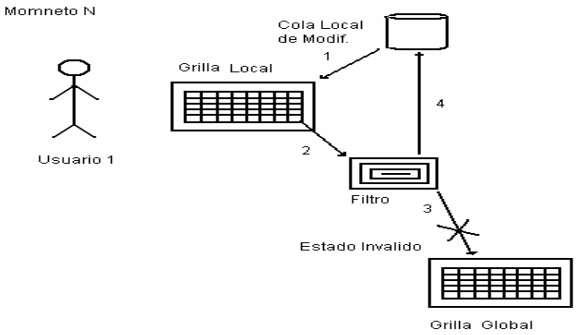

Fig. 4. Selección UNDO

Este toma los datos de su cola local de modificaciones y genera los cambios y se repite el proceso de actualización en la grilla global, pero los mismos son rechazados

pues la versión de la grilla ha caducado, pues el usuario 2 realiza cambios sobre la misma que el usuario 1 no contemplo.
Aquí se observa como la percepción del usuario de lo que se debería hacer en un proceso UNDO puede ser contrario a lo que se puede hacer con el proceso UNDO.
Es decir que el diseñador de un proceso UNDO debe evaluar y compatibilizar dos puntos de vista distintos por un lado lo que el sistema como tal debe aplicar como proceso UNDO para que el mismo quede en un estado valido como sistema y por otro lo que el usuario percibe de la aplicación de un proceso UNDO.
Siguiendo esta línea de análisis se puede inferir que el proceso el usuario espera volver hacia atrás lo que es conocido por él, entiéndase lo que él vio anteriormente en la aplicación, esto se relaciona con el concepto de refresco de las pantallas de usuario.
Si los usuarios reciben la información que otros están actualizando el proceso de UNDO se reduce a un esquema de agenda basado en tiempo de llegada; esto es similar a un esquema de agenda serial ya utilizado en bases de dato.
Por otra parte si no es utilizado un esquema de refresco de pantalla, se presenta la problemática expresada anteriormente y es necesario recurrir a un esquema de Commitment Ordering (CO), que es un concepto de serialización que permite manejar el concepto de serialización global de forma efectiva en ambientes heterogéneos con múltiples administradores de recursos (AR) autónomos. CO implementa el algoritmo Strong-Strict Two phase Locking [Raz,Y, 92]. que generara agendas serializables.
Según lo antes expuesto se puede deducir que un proceso UNDO esta relacionado de forma muy estrecha con el esquema de actualización de información que se elija para la aplicación.
Por otra parte se presenta el problema de que tipo de información se debe guardar en estas agendas, al intentar proyectar un método UNDO genérico, nos encontramos que el mismo debe soportar la manera mas genérica el proceso UNDO, una alternativa de solución a este dilema es la de considerar un mecanismo de serialización pero en este caso de objetos, este proceso también es conocido con el nombre de deshidratación e hidratación de objetos, la implementación del mismo es se toma un objeto con la información necesaria para poder ejecutar el UNDO y el mismo una vez agregado en la agenda se serializa y se mantiene en forma latente hasta su eventual des-serialización o hidratación, para que sea aplicado el proceso UNDO. En la figura 5 se detalla el proceso antes referido.
En el mismo dejan de existir las versiones globales de las colas de modificaciones para ser centralizadas por un filtro global, el cual además se encarga de resolver mediante la agenda cualquier problema de inconsistencia de datos, luego estos son enviados a la grilla global y a la cola global de modificaciones, por ultimo los usuarios cuando deseen utilizar un proceso UNDO, trabajan con una copia virtual de la cola global de modificaciones la cual se encuentra actualizada.
Para que el modelo se complete, como se ha referenciado en párrafos anteriores que el proceso UNDO esta muy relacionado con el refresco de pantallas, se presenta en la figura 6 el ultimo paso para completar el ciclo de un proceso UNDO, que es el refresco de las pantallas de los distintos usuarios con las modificaciones que se realizan.
En la figura anterior se puede observar que el mecanismo para resolver el problema de percepción por parte del usuario es unificar la cola de modificaciones para todos los usuarios y mantener actualizada la vista local que los mismos tienen de la grilla.

La automatización de este modelo no se podría llevar adelante sin la incorporación del concepto de Unidades Lógicas de Cambio.

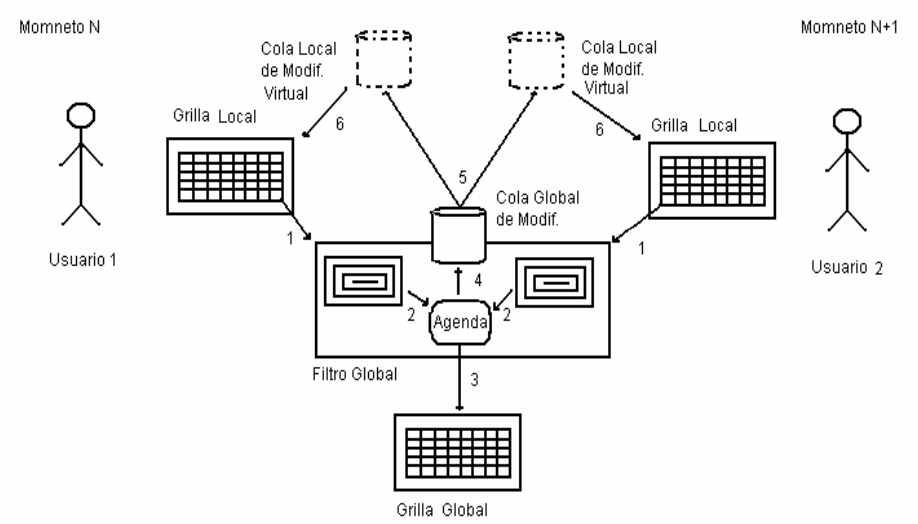

Fig. 5. Carga de modficaciones

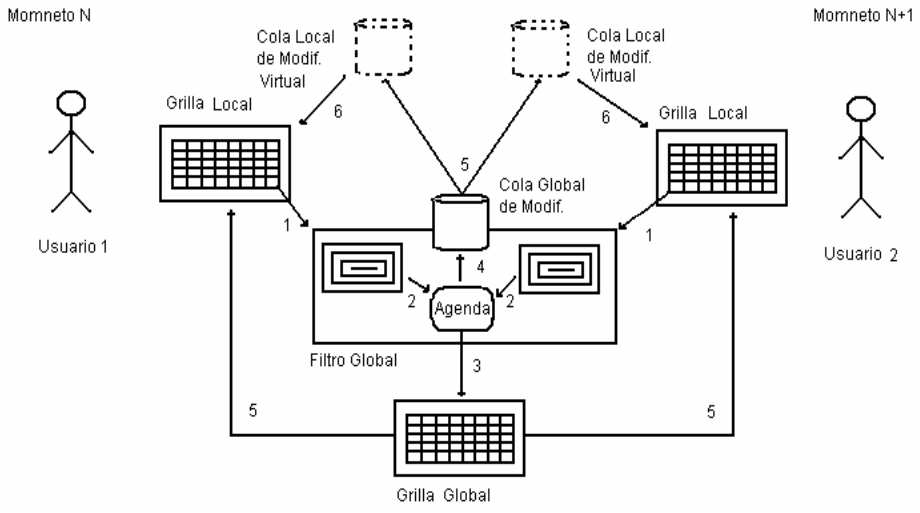

Fig. 6. Actualización de la vista

Se presenta un modelo de diseño simplificado como se puede observar en el diagrama de secuencia de la Figura 7 y el diagrama de clases de la Figura 8.

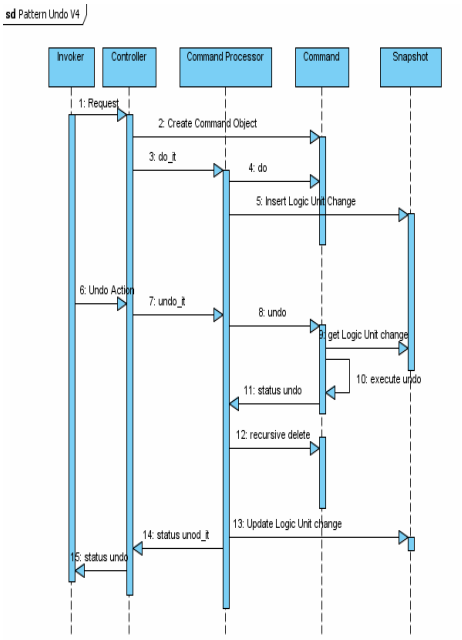

Fig. 7. Diagrama de Secuencia

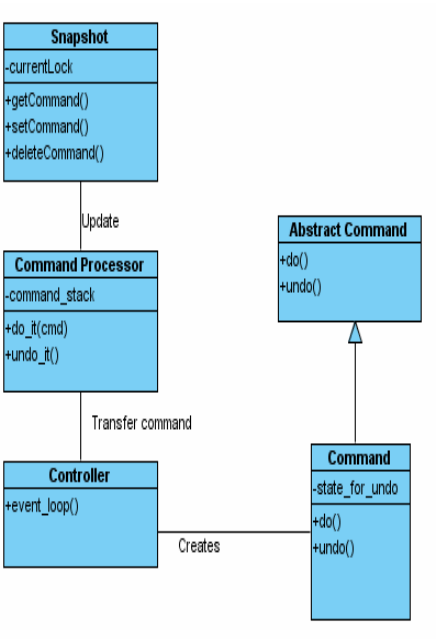

Fig. 8. Diagrama de Clase

El siguiente diagrama de actividad se representa (Figura 9) la carga de información de los cambios a través de los distintos Swimlines. En la figura 10 se representa la sucesión de pasos a realizar pro los distintos Swimlines, en el momento de utilizar esa información.

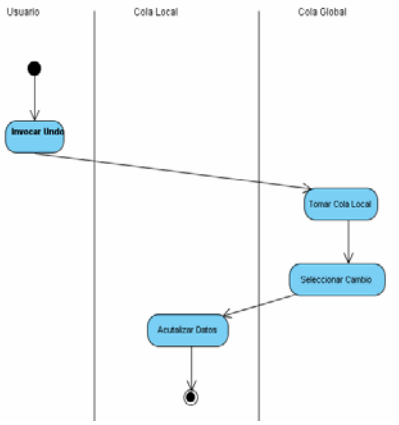

Fig. 9. Diagram de Actividad

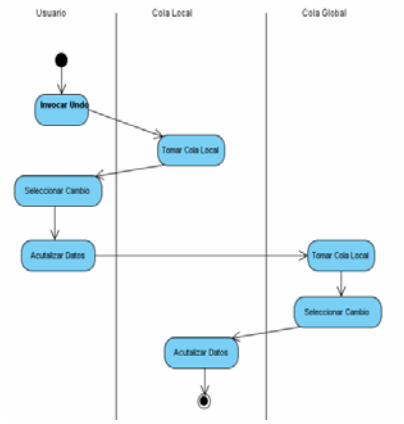

Fig. 10. Diagrama de Actividad

La siguiente maquina de estados (Figura 11) representa los diferentes pasos a seguir pro el sistema.

En la siguiente abstracción (Figura 12) se detalla la forma en que el proceso UNDO es agregado a un sistema, se a tomado como ejemplo una aplicación que trabaja en un ambiente Web, en la estructura de la mensajería que se utiliza para comunicar al cliente con el servidor, se puede nombras como ejemplo el XML, se agregan las Unidades Lógicas de Cambio (ULC) estas al ser recibidas por el servidor son pasadas a través de un filtro, en este sentido se puede utilizar el patrón Filter, o adecuar el sistema para que el mismo soporte el paradigma de programación orientada a Aspectos, este filtro toma la información relacionada con las ULC y genera el circuito detallado en el diagrama de actividad 1 (Figura 9), por otro lado actúa como un mero conector para enviar el resto de la mensajería al sistema para que sigua su camino habitual.

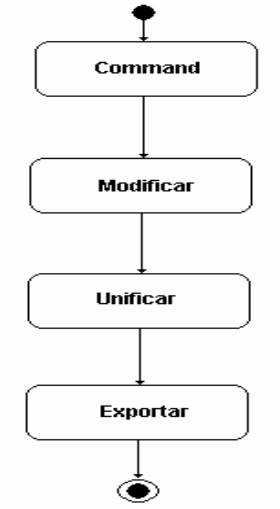

Fig. 11. Estados

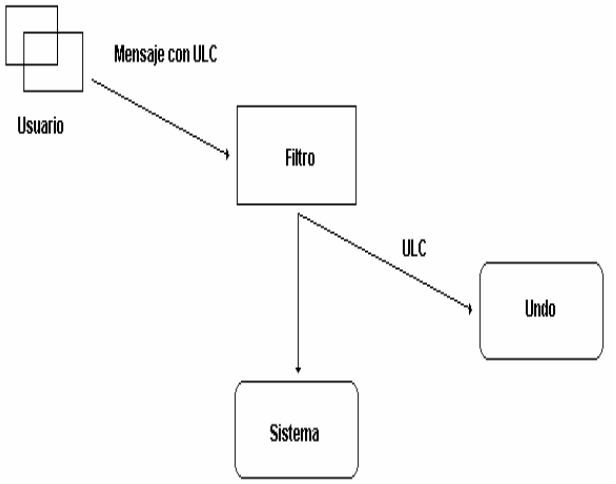

Fig. 12. Abstraccion

Se puede observar que la inclusión de un mecanismo UNDO tanto sea en un sistema por hacer como en un sistema existente es sencilla desde el punto de vista de la intersección del UNDO con el sistema, pues una vez que se resuelve la extracción de las ULC, ambos se mantienen independientes.

6. Conclusiones

La implementación de un mecanismo de UNDO es una tarea no trivial (en particular si se considera su utilización en un ambiente multiusuario) la cual ha tendido a ser realizada en forma especifica para cada sistema.
En este trabajo se han propuesto las unidades lógicas de cambio para abstraer la morfología del objeto ha ser cambiado a un estado previo del mismo por el mecanismo UNDO del sistema. En la propuesta se ha buscado un alto grado de automatización e independencia.

Queda identificada como futura línea de trabajo la utilización de las unidades lógicas de cambio como infraestructura a ser utilizada como base de un mecanismo de REDO (como contraparte del UNDO).

7. Referencias

Abowd, G.; Dix, A. 1991. *Giving UNDO attention*. University of York.

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

Baker, B. y Storisteanu, A. 2001. *Text edit system with enhanced UNDO user interface*. PN: 6.185.591 US.

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

Berlage. A. 1994. *A selective UNDO Mechanism for Graphical User Interfaces Based On command Objects*. German National Reseach Center for Computer Science.

Burke, S. 2007. *UNDO infrastructure*. PN: 7.207.034 US.

Carroll, J. 2003. *HCI Models, Theories and Frameworks*. Morgan Kaufmann.

Cogswell, J. 2004. *Design Highly Useable software*. Sybex.

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

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

Keane, P. y Mitchell, K. 1996. *Method of and system for providing application programs with an UNDO/redo function*. PN:5.481.710 US.

Korenshtein, R. 2003. *Selective UNDO*. PN: 6.523.134 US.

Li, C. 2006. *UNDO/redo algorithm for a computer program*. PN: 7.003.695 US.

Mancini, R., Dix, A., Levialdi, S. 1996. *Reflections on UNDO*. Universidad de Roma.

Martinez, A. y Rhan, M. 2000. *Graphical UNDO/redo manager and method*. PN: 6.111.575 US.

Nakajima, S. y Wash, B. 1997. *Multiple level UNDO/redo mechanism*. PN: 5.659.747 US.

Qin, X. y Sun, C. 2001. *Efficient Recovery algorithm in Real-Time and Fault-Tolerant Collaborative Editing Systems*. School of computing and Information Technology Griffith Univesity Australia.

Raz, Y. 1992. *The principle of commitment ordering, or guaranteeing serializability in a heterogeneous environment of multiple autonomous resource managers using atomic commitment*. 18th International Conference on Very Large Data Base, Morgan Kaufman,

Stuart, C., Thomas, M., Allen, N. 1986. *The psychology of human-computer interaction*. CRC.