Modelo de Inclusión de la Funcionalidad UNDO/REDO

H. Merlino, O. Dieste, P. Pesado, R. García-Martínez

Programa de Doctorado en Ciencias Informáticas. Facultad de Informática. Universidad Nacional de La Plata.
Grupo de Investigación en Sistemas de Información. Departamento de Desarrollo Productivo y Tecnológico. Universidad Nacional de Lanús.
Grupo de Ingeniería de Software Empírica. Facultad de Informática. Universidad Politécnica de Madrid.
Instituto de Investigaciones en Informática LIDI. Facultad de Informática. UNLP - CIC
hmerlino@gmail.com, odieste@fi.upm.es, ppesado@lidi.info.unlp.edu.ar, rgarcia@.unla.edu.ar

Resumen. Se propone la utilización de un servicio de software (SaaS) junto a una metodología para realizar en forma ordenada la inclusión de la funcionalidad de Undo/Redo dentro de una aplicación nueva o existente.

Palabras Clave: Funcionalidad UNDO/REDO, inclusión de servicios metodología

1. Introducción

Los principios de usabilidad son un conjunto de buenas prácticas de software, las cuales hacen que una aplicación tenga una interacción acorde a las características y expectativas del usuario [1], dentro de estos principios se encuentra la funcionalidad de Undo/Redo, la cual permite que un usuario pueda deshacer o rehacer una acción, de aquí en mas se nombrará indistintamente al patrón como Undo/Redo o solo Undo. La inclusión de esta funcionalidad en un sistema nuevo o existente no es un proceso trivial, una de las razones es que su inclusión generalmente se realiza en una etapa avanzada del desarrollo de sistemas [1], cuando las decisiones claves de diseñado ya han sido tomadas.

Estos principios de usabilidad han tomado la forma de patrones de usabilidad, los cuales han sido concebidos con el objetivo, que el desarrollo de software sea simple y predecible [2], estos patrones se pueden definir como mecanismos utilizados durante el diseño del sistema para proporcionar al software un conjunto de característica específica de usabilidad [1]. Algunos patrones de usabilidad definidos en la literatura son: Feedback, Undo/Redo, Cancel, Form/Field Validation, Wizard, User Profile y Help [3]. El principal escollo para la aplicación de estos patrones es la no existencia de un entorno de trabajo, de aquí en mas Framework, que contemple todo el proceso de inclusión del artefacto software "Funcionalidad de Usabilidad" en una aplicación nueva o existente, de aquí en más aplicación anfitriona, haciendo hincapié en aspectos arquitectónicos, de diseño y de performance asociados con los patrones de usabilidad. Esto significa que los patrones tienen que ser aplicado ad hoc en cada sistema; esto

implica que el costo del desarrollo del sistema se incrementará como resultado de la mayor carga de trabajo causada por cada diseño e implementación de las funcionalidades de usabilidad, además, ciertas características de usabilidad quedarán fuera del desarrollo para reducir ese esfuerzos.
El objetivo de este trabajo es desarrollar un Framework para los principales patrones de usabilidad. Se ha seleccionado para comenzar el patrón Undo/Redo, este es un patrón de uso común en la literatura [4]; esto justifica la selección del mismo para el Framework. Hay otros motivos de carácter técnico que sostienen la decisión de empezar con el patrón Undo; es que este patrón, comparte gran parte de su infraestructura con otro patrón que es Cancel y también esto se aplica pero en menor grado a patrones como Feedback y Wizard.
Varios autores han definido alternativas al patrón Undo, estas se centran en dominios particulares, en especial en el área de los editores de texto [5] [6], aunque los conceptos subyacentes pueden ser exportables a otros dominios; estas propuestas se definen a nivel alto, sin una implementación real que demuestre su capacidad de ser reutilizable en diferentes tipos de sistemas, en consecuencia, estas propuestas no resuelven el problema en un sentido amplio.
En este artículo se presenta un nuevo enfoque para la funcionalidad Undo/Redo, esta propuesta resuelve un subconjunto de casos (modelo de operaciones sin estado) en forma eficiente, demostrando la importancia de disponer de una solución automatizada para la inclusión de la funcionalidad en sistemas nuevos o existentes.
Se ha implementado un Framework el cual consta de un servicio (SaaS) de Undo/Redo y la metodología para su inclusión en forma ordenada y progresiva en una aplicación anfitriona; para este desarrollo se han tomado como idea base Frameworks como ser Spring [7] e Hibernante [8], por último se ha hecho mucho hincapié en el hecho que la aplicación anfitriona deba recibir un conjunto reducido y sencillo de modificaciones para poder incluir la funcionalidad de Undo/Redo.
Este artículo se estructura como sigue. La sección 2 describe el estado de la cuestión; la sección 3 presenta el problema a resolver; la sección 4 presenta la solución propuesta y por último, la sección 5 discute brevemente y se presentan las principales aportaciones de nuestro trabajo.

2. Estado de la Cuestión

La funcionalidad de Undo/Redo es una característica muy utilizada y central en toda una gama de aplicaciones como ser, los procesadores de texto, hojas de cálculo, editores gráficos, etc. A modo de ejemplo se puede hacer referencia a Bates y Ryan [6] y Baker y Storisteanu [9] que han patentado métodos para la implantación de la funcionalidad de Undo/Redo en los editores de documentos en entornos monousuario.
Existen soluciones específicas para los editores de texto de uso compartido que soportan funcionalidad de Undo/Redo, como en Sun [10] y Chen y Sun [11] y Yang [12]. La razón de la cantidad de soluciones presentadas para editores de texto es su relativa simplicidad. Conceptualmente, un editor es un contenedor de objetos con ciertas propiedades (forma, posición, etc.), por lo tanto, el Undo es relativamente fácil de implementar, trata de almacenar el estado del contenedor en unidades de tiempo i, i

+1, ..., i + n; luego, cuando el comando Undo es invocado, el contenedor se ejecuta a la inversa i + n, i + n-1, i. Una derivación de las soluciones propuestas para los editores de texto es una implementación alternativa de Undo/Redo para sistemas de correo electrónico como Brown y David [13], estas soluciones se implementan dentro del editor de texto que posee sistemas de correo electrónico.

Los problemas de Undo en entornos multiusuario también han atraído una atención significativa, Abrams y Oppenheim [14] han propuesto mecanismos para el uso de Undo en entornos distribuidos, Abowd y Dix [4] proponen un marco formal de referencia para poder definir la funcionalidad de Undo. En los entornos distribuidos, la solución tiene que manejar la complejidad de los cambios de los datos compartidos esto se realiza por medio de un archivo de historial de cambios [15].

Varios trabajos han proporcionado información sobre los aspectos internos de Undo, como ser Mancini y Dix [16] que describe las características del proceso de Undo, del mismo modo, Berlage [17] propuso la construcción de un método de Undo en base a comandos en entornos gráficos, Burke [18] ha trabajado sobre el concepto de una infraestructura de Undo y Korenshtein [19] da las pautas para realizar un modelo de Undo selectivo.

Otro aspecto en el cual se ha estado trabajado es la elaborado es un modelo de representación de las acciones realizadas por los usuarios en Washizaki y Fukazawa [20], esta es una estructura dinámica de los comandos ejecutados en forma histórica.

La funcionalidad de Undo mediante la representación de modelos de grafos ha sido ampliamente desarrollada por Berlage [17], aquí se presenta una distinción entre el Undo lineal por medio de un archivo histórico y un no lineal, el cual es representado por un grafo, donde se pueden abrir diferentes ramas de acuerdo a las acciones del usuario. Edwards [21] también presenta una estructura de grafo donde a diferencia del trabajo anterior las ramas del árbol representan un nuevo conjunto de acciones realizadas por el usuario. Dix [22] trabajo sobre un grafo en forma de cubo para representar la historia de las acciones realizadas por el usuario. Por su parte Edwards [23] modela las acciones de Undo en forma de hilos paralelo de ejecución. Milestoning y Rollback [24] han utilizado un registro donde se almacena temporalmente las acciones, este modelo ha sido ampliamente utilizado por su sencillez. Todos estas alternativas de representación de las acciones de Undo en forma de comandos son válidos, pero no es una tarea sencilla de implementar, ya que crear una nueva rama de acción y la unión de dos ramas existentes no es una acción trivial, pues se deben conocer todas los posibles caminos que puede tomar un usuario; en consecuencia puede ser recomendable generar una estructura lineal ordenada por tiempo, esta estructura puede ser una cola, que es fácil de implementar y administrar.

Históricamente, se han utilizado para representar el Undo/Redo el patrón “Command” [25], Fayard, Shumidt [26] y Meshorer [27] esto sirve para mantener una lista de comandos ejecutados por el usuario, pero no es suficiente para crear un Framework que sea sencillo de incluir en sistemas existentes. La funcionalidad de Undo/Redo también se ha asociado a los mecanismos de excepción para revertir una acción que falla como en Shinnar et.al. [28], este modelo sólo se invoca ante una falla.

También se ha trabajado en patentes, como el método para la construcción de un proceso de Undo/Redo en un sistema, como en Keane y Mitchell [29] curiosamente, en este trabajo se presenta lo contrario de un proceso de Undo, es decir, ejecutar nuevamente la acción de deshacer el deshacer. Nakajima y Wash [30] definen un

mecanismo para la gestión de múltiples niveles de Undo/Redo, Li [31] describe un algoritmo de Undo/Redo, y Martínez y Rhan [32] presentan un método de administración gráfica de Undo/Redo, basado principalmente en un modelo de interfaz gráfica.
El mayor problema con lo antes referido es una vez más, son difíciles de adoptar en los procesos de desarrollo de software fuera del dominio de editor de textos. La única excepción notable a esto es un patrón a nivel de diseño llamado Memento [33]. Este modelo recupera un objeto a un estado anterior y proporciona un mecanismo independiente de la implementación que se pueden integrar fácilmente en un sistema; el inconveniente es que este patrón no es fácil de incluir en un sistema existente. Además, Memento sólo restaura un objeto a un estado anterior, no se considera ninguna de las otras opciones que un patrón de Undo debe incluir.
Las soluciones presentadas están optimizadas para casos particulares y son difíciles de aplicar a otros dominios, por el otro lado, es necesario incluir una gran cantidad de código asociado a la funcionalidad de Undo en la aplicación anfitriona.

3. Problema

Para un sistema software la inclusión de la funcionalidad Undo/Redo puede ser una funcionalidad primordial o ser una funcionalidad deseable.
En los de primer tipo, funcionalidad primordial, un diseñador contempla la construcción de esta en el cuerpo principal de la aplicación; estas son aplicaciones que no pueden ser concebidas sin la existencia de la funcionalidad de Undo/Redo, en esta categoría se incluyen los procesadores de texto, un usuario de este tipo de sistemas espera la existencia de la misma. Aquí se incluye planillas de cálculo, gestores de correos electrónicos, sistemas de mensajería instantánea, entre otras.
Por otro lado las aplicaciones que se enrolan en la segunda categoría, funcionalidad deseable, el diseñador enfrenta el problema que debe incluirla para que la aplicación sea desde el punto de vista de la usabilidad completa, pero esto insume tiempo y esfuerzo para una funcionalidad que no es el núcleo de la aplicación, en esta categoría se enrolan aplicaciones como gestores de administración de base de datos, sistemas de carga de datos con interfaz de usuario, entre otros. Como se puede observar en esta última categoría los usuarios del sistema al manejar un ingreso de datos están propensos a generar errores en la carga que deben ser corregidos, los usuarios desearían poder corregir esto rápidamente, con lo cual se podría inferir que la funcionalidad es deseable e importante.
Enfocados en esta segunda categoría el presente trabajo intentara dar respuesta a los siguientes disparadores; (a) es posible desarrollar una infraestructura tal que, la inclusión de la funcionalidad de Undo/Redo sea simple y rápida en una aplicación nueva o existente; (b) es posible definir una metodología para la detección de las características de Undo/Redo que una aplicación necesita.

4. Solución Propuesta

En la siguiente sección se detallan distintas aproximaciones para satisfacer la funcionalidad e Undo/Redo, en esta se realiza una comparación entre ellas y se fija posición ante cada una, las misma son:

Ad Hoc: Es la inclusión de la funcionalidad de Undo dentro de una aplicación particular escribiendo todo el código, no se reutiliza código anterior ni de las lecciones aprendidas en otras aplicaciones y se dimensiona según las necesidades de cada sistema

Patrones: Se incluye la funcionalidad de Undo dentro de una aplicación siguiendo los lineamientos de una solución probada, todo el código se incluye en la aplicación anfitriona. La desventaja de la realización de esta alternativa de diseño es la inclusión de toda la complejidad de la funcionalidad Undo/Redo implica un considerable incremento en la cantidad de líneas de código de la aplicación en cuestión, con la consecuente propensión a agregar errores en el mismo. Esta alternativa es viable en los casos en que la funcionalidad de Undo/Redo sea un elemento central de la aplicación.

Software as a Service (SaaS): La aplicación anfitriona incluye invoca determinada funcionalidad a través de un servicio externo, en este caso el Undo, toda la complejidad del mismo se encuentra encapsulada dentro de este y en la aplicación. Esta solución tiene la ventaja de reducir la cantidad de código necesario dentro de la aplicación anfitriona, además permite mantener la complejidad requerida para cada aplicación, pues el servicio tiene el conjunto máximo de características de la funcionalidad de Undo/Redo. La desventaja de esta solución en forma aislada que el método de inclusión desde la aplicación anfitriona queda de lado del programador de la aplicación anfitriona y a su experiencia con la funcionalidad Undo/Redo.

Framework: Como definición de Framework se dará la utilización de un conjunto de rutinas encapsuladas dentro de una API (Application Program Interface), la cual tiene un cierto orden de invocación. En este caso pueden ser combinados dentro de esta estrategia la utilización de patrones y SaaS.

Metodología: Esta estrategia abarca además de la utilización de un microframework, que se ha detallado anteriormente, la utilización de un conjunto de pasos para detectar las características de Undo necesarias para la aplicación anfitriona y un conjunto de tareas post implementación para el mejoramiento perfectivo de la misma. Esta última alternativa es donde se enrolaría la solución propuesta en este trabajo. Se ha seleccionado un conjunto de características deseables para una implementación de una solución en un sistema y se las ha comparado (Tabla 1).

Tabla 1. Comparación. (*Escala*= NC no cumple la condición, CP cumple parcialmente, CT cumple totalmente la condición)

Alternativa	Repetible	Escalable	Auditable	Perfectible	Transferible	Integrable
Ad Hoc	NC	NC	CP	CP	NC	NC
Patrones	CT	CP	CT	CT	CT	CP
SaaS	CT	CT	NC	NC	CT	CP
Framework	CT	CT	CP	CP	CT	CT
Metodología	CT	CT	CT	CT	CT	CT

Como se puede observar la implementación de una metodología es la alternativa mas sólida para el Undo. La construcción de la funcionalidad de Undo/Redo bajo la modalidad de servicio genera que toda la complejidad de la funcionalidad Undo/Redo se ocultada por el servicio, las ventajas de esta aproximación son permite construir sobre un servicio la complejidad que se desee para trabajar con la funcionalidad en cuestión, pues la misma puede ser reutilizada a través de distintas aplicaciones, con lo cual el esfuerzo de su construcción queda justificado.
El servicio cuenta con un conjunto de pantallas para su configuración y administración, donde se volcaran datos como aplicación a ser soportada de ser necesario usuarios autorizados para su administración y/o uso, carga de los requisitos de usabilidad para la aplicación y análisis histórico de uso. El servicio como tal provee una interfaz para solicitar el almacenamiento temporal de datos o consulta de los mismos.
Las fases del proceso propuesto con descripción de tareas, entradas y salidas se presentan en la Tabla 2, el flujo del mismo en la Figura 1 y en la Tabla 3 de presentan los detalles de cada fase potenciales problemas y soluciones.
El poseer un método para la definición de los datos a ser incluidos en una posible invocación del servicio de Undo/Redo es importante para poder concretar en forma satisfactoria la inclusión del servicio en la aplicación anfitriona.
Como introducción a los conceptos detallados debajo se define el concepto de Unidad Lógica de Cambio (ULC), este hace referencia a un conjunto de datos que deben ser tratados como una unidad indivisible a efectos de la coherencia de la información de sistema.

5. Conclusiones

En este trabajo se ha propuesto el diseño de una Framework para la funcionalidad de Undo/Redo para ser agregado en una aplicación de software, la cual se alinea con las características de funcionalidad necesaria, pero no para aplicación que sea central la funcionalidad de Undo/Redo, como ser un procesadores de texto. La característica más sobresaliente de este Framework es el tipo de información que almacena para poder deshacer o rehacer las operaciones de los usuarios, datos de entrada en lugar de estados de objeto en la memoria o los comandos ejecutados por el sistema.
Un servicio de Undo/Redo tiene algunas ventajas significativas con respecto a otros modelos presentados, en primer lugar, la sencillez de su inclusión en un software ya existente, el esfuerzo en comparación con los otros modelos es significativamente menor (ver Tabla 1). En segundo lugar, la independencia que provee el servicio en relación a la aplicación anfitriona permite crear modelos mas sofisticados de Undo para cada dominio pues toda esa complejidad se encapsula en el servicio mismo.
Las futuras líneas de investigación que se están contemplando son: (a) la creación de un pre-compilador, (b) la detección automática de los campos susceptibles a un proceso de Undo, (c) ampliar el marco a otras plataformas.

Tabla 2. Fases del proceso propuesto.

PASOS	ENTRADA		TAREA	SALIDA	
	DENOMINACION	REPRESENTACION		DENOMINACION	REPRESENTACION
1. Análisis de Usabilidad (E1-AU)	Requerimientos de Sistema (E1-AU-RS)	Documento	Detección de Requerimientos de Usabilidad (E1-AU-AU)	Requisitos de Usabilidad (E1-AU-RU)	Documento
	A partir de los casos de uso del sistema se intenta detectar los requisitos			*Se obtienen los requisitos de usabilidad definidos para el sistema*	
2. Detección de Unidades Lógicas de Cambio (E2-ULC)	Requisitos de Usabilidad (E1-AU-RU)	Documento	Detección de ULC (E2-ULC-DE)	Definición de ULC (E2-ULC-DI)	Documento
	Estos requisitos son la base para detectar las ULC			*Conjunto de información que se tratara como una unidad*	
3. Detección de Puntos de No Retorno (PNR) (E3-PNR)	Diseño de ULC + RS (E2-ULC-DI) (E1-AU-RU)	Documento	Detección de PNR (E3-PNR-DE)	Definición de PNR (E3-PNR-DF)	Documento
	El conjunto de los Requisitos y las ULC permiten detectar los PNR			*Hitos en la utilización del sistema que no permiten volver a tras al mismo*	
4. Prueba de Usabilidad Original (E4-PUO)	Sistema Anfitrión	Software	Prueba de usabilidad de sistemas (E4-PUO-U)	Reporte de Usabilidad (E4-PUO-AU)	Documento
	Se le aplica al sistema original una prueba para validar su estado			*Se obtiene un reporte con el análisis de usabilidad del mismo*	
5. Prueba de Stress (E5-PS)	Sistema Anfitrión	Software	Ejecución de la Prueba de Stress (E5-PS-EPS)	Reporte de Prueba de Stress (E5-PS-APS)	Documento
	Se le aplica al sistema original una prueba para validar su estado			*Se obtiene un reporte con el análisis de stress del mismo*	
6. Configuración de Servicio (E6-CS)	Diseño de ULC + Definición de PNR + Permisos de Acceso (E2-ULC-DI) (E3-PNR-DF)	Software + Documento	Configuración de Servicio (E6-CS-CS)	Servicio Configurado (E6-CS-SC)	Software
	Ser recolecta la información necesaria para la configuración del sistema			*Se obtiene el servicio configurado para el sistema anfitrión*	
7. Método de Inclusión del Servicio (E7-MIS)	Definición de ULC (E1-AU-RU)	Documento	Evaluación de Métodos de Inclusión (E7-MIS-EMI)	Método de Inclusión (E7-MIS-MI)	Documento
	A partir de las ULC detectadas se analiza el mejor método para incluir el servicio en el sistema anfitrión			*Definición del mejor método para inyectar el servicio en le sistema*	
8. Inclusión del Servicio (E8-IS)	Sistema Anfitrión + Método de Inclusión (E7-MIS-MI)	Software + Documento	Programación o rotulado de archivos fuentes (E8-IS-PRF)	Sistema Anfitrión + Invocación a Servicio (E8-IS-IS)	Software
	Se procede a incluir en el sistema las invocaciones al servicio.			*Invocaciones al servicio desde el sistema anfitrión.*	
9. Prueba de Usabilidad (E9-PU)	Sistema Anfitrión + Invocación al Servicio (E4-PUO)	Software	Prueba de usabilidad de sistemas (E9-PU-U)	Aceptación de Usabilidad (E9-PU-AU)	Documento
	Se lleva a cabo una nueva prueba de usabilidad			*El resultado de esta no debería variar del anterior sumado el servicio*	
10. Prueba de Stress (E10-PS)	Sistema Anfitrión + Invocación a Servicio (E5-PS-APS) (E8-IS)	Software	Ejecución de la Prueba de Stress (E10-PS-EPS)	Aceptación de Prueba de Stress (E10-PS-APS)	Documento
	Se lleva a cabo una nueva prueba de stress			*El resultado de esta no debería variar del anterior sumado el servicio*	
11. Evaluación (E11-EV)	Reporte de Usabilidad + Reporte de Prueba de Stress + Aceptación de Usabilidad + Aceptación de Prueba de Stress (E5-PS-APS) (E4-PUO-AU) (E9-PU-AU) (E10-PS-APS)	Documento + Documento + Documento + Documento	Evaluación de Resultados (E11-EV-R)	Aceptación Inclusión de Servicio (E11-EV-AIS)	Documento
	Evaluación final de los documentos generados en las pruebas, los valores obtenidos antes y después de la inclusión deberían ser similares.			*Aceptación final del proceso*	

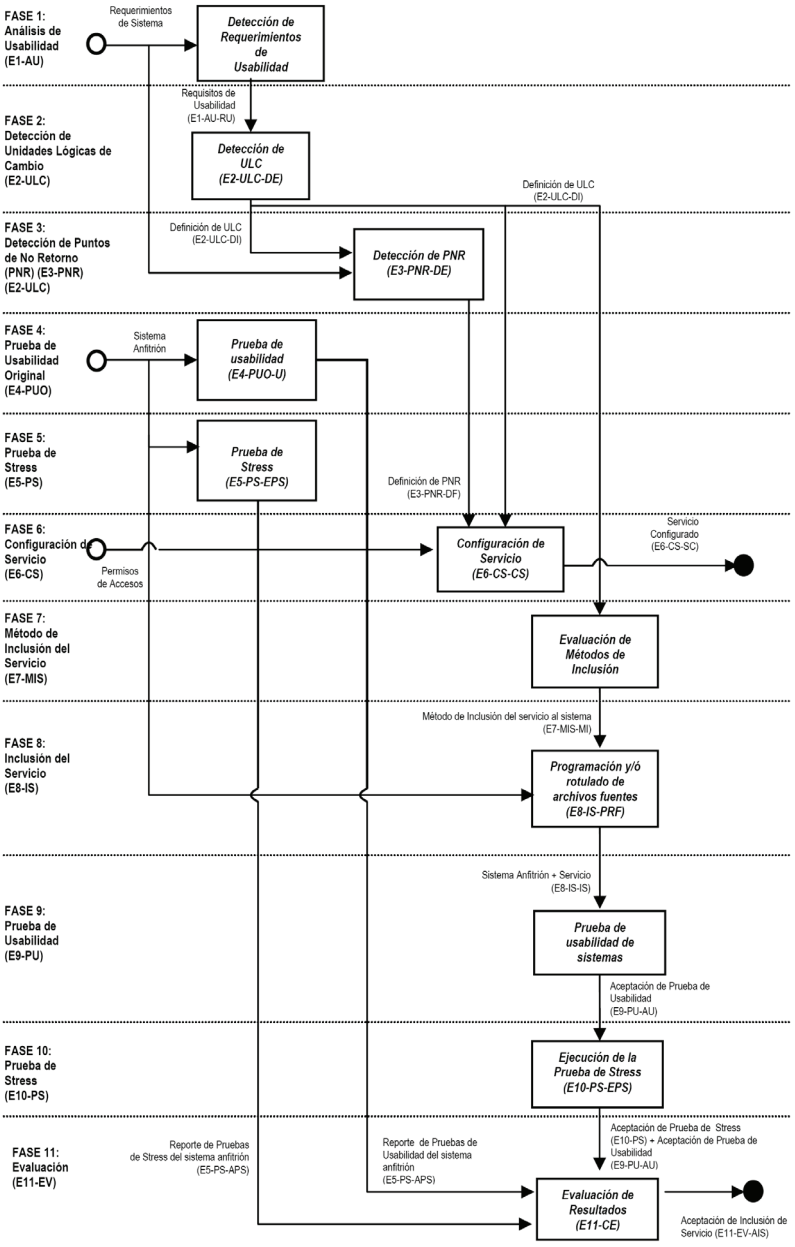

Fig. 1. Flujo de proceso propuesto.

Tabla 3. Detalles de cada fase, potenciales problemas y soluciones propuestas.

FASE	DETALLE	POTENCIALES PROBLEMAS	SOLUCION PROPUESTA
1.- Análisis de Usabilidad (E1-AU)	Es el método por el cual a través de los requisitos generales del sistema se extraen los de usabilidad, en relación con la funcionalidad de Undo/Redo. Este proceso puede requerir nuevas sesiones de relevamiento con el usuario para recolectar toda la información necesaria para poder definirlos, esto se debe a que en ciertas circunstancias se dejan de lado el proceso de relevamiento de los requisitos de usabilidad para centrarse en los requisitos de funcionalidad del sistema en cuestión.	En este punto surge el primer desafío con respecto al proceso de educción de los requisitos de usabilidad. A diferencia de los requisitos funcionales, los cuales se podría definir, como los objetivos que debe cumplir un sistema para justificar su existencia, los requisitos de usabilidad específicamente con respecto a la funcionalidad Undo/Redo no son lineales como pueden ser los antes referenciados; esto se debe a que la funcionalidad de Undo/Redo esta relacionada con la propensión de los usuarios del sistema a cometer errores o a cambios durante la operatoria del sistema. Como se puede deducir poder detectar de ante mano posibles fallas en la inclusión de datos es un proceso que si no se cuenta con información previa de la operatoria del sistema es extremadamente difícil poder inferirlo.	Tomar los requisitos funcionales de sistema, inferir posibles requisitos de usabilidad relacionados con el Undo/Redo, de no poder hacerlo, realizar las reuniones necesarias con el usuario, iterar hasta obtener un conjunto de datos mínimos para poder trabajar con la siguiente etapa del proceso.
2.-Detección de Unidades Lógicas de Cambio (E2-ULC)	En este punto se definen las ULC que existen en el sistema y que son relevantes para soportar el mecanismo de Undo. La entrada de esta tarea son los requisitos de usabilidad educidos en la etapa anterior, con este conjunto de requisitos se definen los conjuntos de datos que serán rotulados con diferentes códigos para las ULC.	El potencial problema que se presenta en esta etapa es la falla en la detección de las ULC, esto trae aparejado que los datos almacenados en el servicio sean o insuficientes o excesivos para el proceso de Undo/Redo que solicita el usuario.	Realizar un proceso iterativo partiendo de una primera versión de las ULC y iterar hasta encontrar la versión mas apropiada de las ULC, es necesario hacer notar que las ULC no son un conjunto fijo y único de datos que deberán ser tratados como una unidad indivisible, sino que estos con el tiempo pueden variar y por otra parte distintos usuarios pueden generar distintos patrones de errores.
3.- Detección de Puntos de No Retorno (PNR) (E3-PNR)	Los Puntos de No Retorno (PNR) son estadios del sistema que por motivos propias al domino o a limitaciones practicas del sistema no se puede retroceder mas allá de este, estos suelen ser mas comunes en ambientes multiusuario donde la modificación de un usuario hace que la pila de modificaciones realizada por otro usuario queda invalidad. Detalle: Aquí se definen, según lo detallado en el apartado Puntos de No Retorno, los PNR que deberán ser considerados para la inclusión del mecanismo de Undo. En determinada circunstancias los datos no pueden ser vueltos hacia atrás por cuestiones de consistencia de datos, como ser la actualización en una base de datos. Este tipo de situaciones de lógica de datos dentro de un sistema tienen injerencia sobre el servicio de Undo, pues este debe reconocer estas situaciones para no almacenar datos no validos.	En este punto se presenta la problemática de reconocer este tipo de situaciones, aquí existen dos alternativas: (a) que sean conocidas de antemano estas situaciones y se sepan en que momento suceden, (b) que sean conocidas de antemano y no se sepa cuando suceden. En caso de ser del primer grupo estas pueden ser resultas al momento de inicializar el servicio para que interactúe con el sistema. Las del segundo tipo son mas complejas y es necesario interactuar en tiempo real con el sistema para que el servicio este informado de cuando se produjo una situación de no retorno. Esto provoca que el sistema, Back End, comunique por algún medio al servicio esto para que el mismo marque los datos como no validos. Aquí es donde la lógica propuesta del servicio que interactúa con el Front End, queda desvirtuada y es necesario dar una solución elegante a la misma.	Reglas de detección de PNR: • ¿Existen alguna acción por parte del usuario y/o sistema que no permite volver hacia atrás al sistema? • ¿Existe alguna restricción de tiempo que impida volver hacia a tras al sistema? En caso que alguna de estas preguntas sea afirmativa existe un PNR y el servicio de undo debe de ser informado de este para marcar como no retornable los datos anteriores a este momento dado.
4.- Prueba de Usabilidad Original (E4-PUO)	El objetivo de la prueba de usabilidad es la de poder reconocer la complejidad que posee la interfaz de usuario de un sistema, esto permite detectar problemas relacionados con la comprensión de la interfaz y el conocimiento del usuario mediante algunos de los métodos de calculo de usabilidad, como ser CPM-GOMS [34], para poder tener una medida que sirva para comparar luego de haber sido agregado el servicio a la aplicación anfitriona, esta evaluación en el caso de ser similar demostraría que la inclusión del servicio no ha agregado ruido al sistema anfitrión y el usuario ha asimilado el cambio.	Que la evaluación de la usabilidad del sistema anfitrión antes de inyectar el servicio no sea satisfactorio con los niveles de calidad esperados.	A nivel del proceso de inclusión del servicio no tiene efecto, pues esta prueba es hecha solo para verificar luego que se incluyo el servicio que no se ha modificado en nada al sistema anfitrión.
5.- Prueba de Stress (E5-PS)	El objetivo de una prueba de stress es el de evaluar la curva de respuesta del sistema ante el uso simultaneo de diferentes usuarios. El concepto que se aplica es el mismo al de la Etapa 4, se realiza la prueba de stress antes de inyectar el servicio para tener una medida con que comparar en la prueba de stress posterior a la inyección del servicio, ambas deberían ser similares.	Que la evaluación de stress del sistema no sea satisfactoria con los niveles de calidad esperados.	A nivel del proceso de inclusión del servicio no tiene efecto, pues esta prueba es hecha solo para verificar luego que se incluyo el servicio que no se ha modificado en nada al sistema anfitrión.
6.- Configuración de Servicio (E6-CS)	En esta etapa, a través de las distintas páginas de configuración que provee el servicio se carga la información definida como entrada para generar la base necesaria para que el servicio puede recibir y responder los requerimientos del sistema a través de sus usuarios. El método de generación seleccionado es del estilo "Wizard", el cual guía al encargado de generar la nueva instancia del servicio Undo/Redo por una secuencia ordenada de pasos y datos a ingresar, una vez finalizado este proceso se genera automáticamente el nuevo modelo de servicio quedando listo para su uso pro la aplicación anfitriona.		
7.- Método de Inclusión del Servicio (E7-MIS)	Esta etapa que puede ser realizada en paralelo junto a E6-CS evalúa cual es el método mas apropiado para que el servicio sea incluido en la aplicación anfitriona, este puede ser en forma programática por el propio desarrollador o utilizando un método de rótulos para que un programa evalúe los mismo e inserte en forma automática las referencias al servicio.		
8.- Inclusión del Servicio (E8-IS)	Se procede a incluir las invocaciones al servicio en la aplicación anfitriona según lo definido en la etapa anterior.		
9.- Prueba de Usabilidad (E9-PU)	Es la evaluación mediante algunos de los métodos de evaluación de usabilidad, por ejemplo GOMS, el mismo intenta cuantificar cuan fácil es para un usuario aprender el uso de la nueva funcionalidad dentro del sistema, el resultado de esta tarea es la evaluación de la coherencia de la funcionalidad insertada con el resto de la aplicación, es de esperar que se mantenga un mismo conjunto de principios de usabilidad.	Que la evaluación de valores diferentes a estado anterior, es decir al sistema sin servicio.	Se deberá revaluar el método de inclusión para no afectar la usabilidad del sistema anfitrión.
10.- Prueba de Stress (E10-PS)	Mediante de alguna de las herramientas existentes para la generación de pruebas de stress de sistemas se evalúa el tiempo de respuesta del mismo sumada la invocación al servicio de Undo/Redo, este debería ser el mismo al antes realizado. En esta etapa además de evaluarse el tiempo de respuesta se evalúa posibles fallas en la funcionalidad de Undo/Redo si la misma comparte datos entre distintos usuarios.	Que la evaluación de valores diferentes a estado anterior, es decir al sistema sin servicio.	Se deberá revaluar el método de inclusión para no afectar la usabilidad del sistema anfitrión.
11.- Evaluación (E11-EV)	Aquí se evalúan los resultados anteriores de las pruebas de usabilidad y stress con el obtenido luego de haber inyectado el servicio, es de esperar que sendos resultados sean similares pues la inclusión del servicio no debería insertar ruido en la aplicación anfitriona. Este proceso da por concluido los pasos de evaluación, inclusión y revisión del servicio y su inyección en la aplicación anfitriona. Luego de esto el servicio y la aplicación están listas para entrar en producción.		

6. Financiamiento

Las investigaciones que se reportan en este artículo han sido financiadas parcialmente por el Proyecto de Investigación 33B066 del Departamento de Desarrollo Productivo y Tecnológico de la Universidad Nacional de Lanús.

7. Referencias

1. 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)
2. Ferre, X; Juristo, N; and Moreno, A. 2004. Framework for Integrating Usability Practices into the Software Process. Universidad Politécnica de Madrid.
3. Juristo, N; Moreno, A; Sanchez-Segura, M; Davis, A. 2005. Gathering Usability Information through Elicitation Patterns.
4. Abowd, G.; Dix, A. 1991. Giving UNDO attention. University of York.
5. 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 University Australia.
6. Bates, C. and Ryan, M. 2000. Method and system for UNDOing edits with selected portion of electronic documents. PN: 6.108.668 US.
7. Spring framework. http://www.springsource.org/. Page Valid at 2011/12/05.
8. Hibernate framework. http://www.hibernate.org/. Page Valid at 2011/12/05.
9. Baker, B. and Storisteanu, A. 2001. Text edit system with enhanced UNDO user interface. PN: 6.185.591 US.
10. Sun, C. 2000. Undo any operation at time in group editors. School of Computing and Information Technology, Griffith University Australia.
11. Chen, D; Sun, C. 2001. Undoing Any Operation in Collaborative Graphics Editing Systems. . School of Computing and Information Technology, Griffith University Australia.
12. Yang, J; Gu, N; Wu, X. 2004. A Documento mark Based Method Supporting Group Undo. Department of Computing and Information Technology. Fudan University, China.
13. Brown, A; Patterson, D, 2003. Undo for Operators: Building an Undoable E-mail Store. University of California, Berkeley. EECS Computer Science Division.
14. Abrams, S. and Oppenheim, D. 2001. Method and apparatus for combining UNDO and redo contexts in a distributed access environment. PN: 6.192.378 US.
15. Berlage, T; Genau, A. 1993. From Undo to Multi-User Applications. German National Research Center for Computer Science.
16. Mancini, R., Dix, A., Levialdi, S. 1996. Reflections on UNDO. University of Rome.
17. Berlage. T. 1994. A selective UNDO Mechanism for Graphical User Interfaces Based On command Objects. German National Research Center for Computer Science.
18. Burke, S. 2007. UNDO infrastructure. PN: 7.207.034 US.
19. Korenshtein, R. 2003. Selective UNDO. PN: 6.523.134 US.
20. Washizaki, H; Fukazawa, Y. 2002. Dynamic Hierarchical Undo Facility in a Fine-Grained Component Environment. Department of Information and Computer Science, Waswda University.
21. Edwards, W; Mynatt, E. 1998. Timewarp: Techniques for Autonomous Collaboration. Xerox PARC.
22. Dix, A; Mancini, R; Levialdi, S. 1997. The cube – extending systems for undo. School of Computing, Staffordshire University. UK.
23. Edwards, W: Igarashi, T; La Marca, Anthony; Mynatt, E. 2000. A Temporal Model for Multi-Level Undo and Redo.
24. O´Brain, J; Shapiro, M. 2004. Undo for anyone, anywhere, anytime. Microsoft Research.
25. Buschmann, F; Meunier, R; Rohnert, H; Sommerlad, P; Stal, M. 1996. Pattern-Oriented Software Architecture: A System Of Patterns. John Wiley & Sons.
26. Fayad, M.; Shumidt, D. 1997. Object Oriented Application Frameworks. Comunications of the ACM, 40(10) pp 32-38.
27. Meshorer, T. 1998. Add an undo/redo function to you Java app with Swing. JavaWord, June, IDG Communications.
28. Shinnar, A; Tarditi, D; Plesko, M; Steensgaard, B. 2004. Integrating support for undo with exception handling. Microsoft Research.
29. Keane, P. and Mitchell, K. 1996. Method of and system for providing application programs with an UNDO/redo function. PN:5.481.710 US.
30. Nakajima, S. and Wash, B. 1997. Multiple level UNDO/redo mechanism. PN: 5.659.747 US.
31. Li, C. 2006. UNDO/redo algorithm for a computer program. PN: 7.003.695 US.
32. Martinez, A. and Rhan, M. 2000. Figureical UNDO/redo manager and method. PN: 6.111.575 US.
33. Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1994. Design Patterns: Elements of Reusable Object-Oriented Software, Addison- Wesley.