Análisis Sistematizado de la Incorporación de la Usabilidad en el Desarrollo de Software

Juan Marcelo Ferreira Aranda, Laurent Gianina Díaz Molas, y Luis Gilberto Salinas
Facultad Politécnica
Universidad Nacional de Asunción
San Lorenzo, Paraguay
{jmferreira1978,laumolas,lg.salinas}@gmail.com

***Resumen*— La calidad es uno de los principales retos de la construcción de software. Mientras que la usabilidad es considerada un atributo de calidad, desde la Ingeniería de Software se la asocia tradicionalmente con los requisitos no funcionales y la aplicación de las recomendaciones de usabilidad quedan postergadas al final del desarrollo donde los problemas son más costosos y, en ocasiones, imposibles de resolver. En este artículo presentamos un análisis sistematizado incorporando la usabilidad desde los inicios del desarrollo. Durante el proceso, evaluamos el impacto de tal incorporación y evidenciamos que el esfuerzo de usabilidad se reduce sustancialmente al distribuir la sobrecarga de trabajo por las distintas etapas del desarrollo. Además, considerando la usabilidad al inicio hemos evidenciado una reducción de costos al no requerir profesionales HCI y al mantener actualizada la documentación asociada al producto software anticipando una mejora en soporte, entrenamiento y productividad.**

Palabras Clave*— *Mecanismo de Usabilidad, Patrones de Programación, Pro- ceso de Desarrollo, Pautas de Desarrollo

I. INTRODUCCIÓN

En el marco de la calidad de un sistema software, la usabilidad es un atributo importante a tener en cuenta así como la seguridad, el rendimiento, la mantenibilidad, entre otros [3, 8, 9]. Tradicionalmente, la Ingeniería de Software (IS) trata la usabilidad como parte de los requisitos no funcionales limitándose a una propiedad exclusiva de la presentación de la información [17]. Debido a la naturaleza del sistema y a las necesidades del usuario, a menudo se debe ir más lejos y no basta con tener en cuenta la presentación para obtener un software usable. También, la comunidad *Human Computer Interaction* (HCI) ha propuesto recomendaciones para mejorar la usabilidad y varias de ellas tienen impacto directo en la funcionalidad del producto software. Así las disciplinas de la IS y la comunidad HCI comparten las mismas metas para recopilar las necesidades del usuario en la construcción de sistemas usables, pero trabajan de manera aislada [17]. Estudios recientes han evaluado la relación entre la usabilidad y los requisitos funcionales sugiriendo que la usabilidad debe ser tenida en cuenta desde las etapas iniciales de la construcción como requisitos funcionales para evitar costosos cambios posteriores [1, 4, 5, 10]. De ahí, la incorporación de las características de usabilidad agrega complejidad adicional al proceso de desarrollo. En este artículo analizamos la incorporación de la usabilidad en el desarrollo de un producto software desde sus inicios. Concretamente evaluamos los siguientes mecanismos de usabilidad (MUs) formalizados como patrones de programación: *Abort Operation*, *Progress Feedback* y *Preferences*. Se utilizan unas Pautas de Desarrollo de Mecanismos de Usabilidad (PDMUs) para estos tres MUs [15]. Estas pautas proponen soluciones en forma de patrones para la captura y posterior incorporación de la usabilidad en las distintas etapas del desarrollo incluida la programación. Para ello abordamos el desarrollo de un producto software desde la perspectiva de dos desarrolladores con distintos niveles de conocimiento en usabilidad partiendo de la educción y especificación de requisitos, pasando por el análisis y diseño hasta la implementación. Cada desarrollador introduce la usabilidad de acuerdo a las recomendaciones de las PDMUs en las distintas etapas del desarrollo. Durante el proceso evaluamos el impacto de incorporación de los MUs. Cada evaluación aporta datos que proporcionan una estimación del esfuerzo adicional requerido para incorporar cada MU en el proceso de desarrollo del software. Finalizamos demostrando que el esfuerzo de incorporar la usabilidad en cada etapa del desarrollo suscita mejoras importantes en distintos aspectos al final del proceso tanto para el desarrollador como para el usuario.

II. PAUTAS DE DESARROLLO DE MECANISMOS DE USABILIDAD O PMDUS

Las Pautas de Desarrollo de Mecanismos de Usabilidad (PDMUs) *–Develop- ment Guidelines for Usability Functionalities –*, a ser utilizadas por los desarrolladores, incluyen los diferentes artefactos para la incorporación de los MUs en las distintas etapas de

desarrollo. Estas pautas son el resultado de las investigaciones de Rodríguez y colegas [15, 16] donde se buscan soluciones reutilizables para análisis, diseño e implementación de algunos mecanismos. Las soluciones obtenidas dan cobertura a las tres etapas principales del desarrollo. En resumen, las mencionadas pautas abordan cuatro aspectos importantes:

1. Requisitos: para la captura de requisitos de usabilidad se utiliza la guía de educción de requisitos por cada mecanismo. En [19], [15] se encuentran las guías de educción para los mecanismos tratados en este artículo.

2. Análisis y Diseño: se trata del análisis y diseño preliminar de los componentes para cumplir con las responsabilidades asociadas al mecanismo de usabilidad. Se presenta un conjunto de diseños reutilizables a nivel de diagramas de clase y secuencia para la incorporación de la usabilidad en la etapa de diseño. En [15] se encuentran los patrones de diseño para los tres MUs.

3. Escenarios de aplicación: trata sobre la descripción de los posibles escenarios de aplicación de cada mecanismo. Los distintos escenarios son esquematizados en una estructura de árbol. Asimismo, se provee una relación entre las recomendaciones de la guía de educción de requisitos de usabilidad, los casos generales resultado de las preguntas de educción, sus interpretaciones a nivel de aplicaciones web y el estado final del sistema. En [15] se encuentra esta información.

4. Patrón de diseño e implementación: hace referencia al patrón de diseño del MU y su correspondiente patrón de programación en *php*, *.Net* y *Java*. La solución está basada en un patrón porque representa la estructura reconocida dentro de la comunidad de la ingeniería de software para proponer soluciones reutilizables. La plantilla del patrón tiene diferentes secciones: un nombre, la descripción del problema al que va dirigido, el contexto de aplicación. Según el tipo de mecanismo las demás secciones varían. En [15] se encuentra, en detalle, la información acerca del patrón de diseño e implementación para los tres MUs.

III. Trabajos Relacionados

Varios trabajos analizan la relación entre la usabilidad y el desarrollo de software. Bass y colegas [2] han estudiado las implicancias de la usabilidad en momentos específicos del desarrollo fuera de la interfaz gráfica del usuario. Partiendo de una serie de escenarios ellos describen la interacción de un *stakeholder* con un sistema software atendiendo ciertos aspectos de usabilidad. Como resultado de su experimento se ilustra cómo el diseño de estos escenarios requiere modificaciones no sólo en la interfaz del sistema sino también en otras partes de sus funcionalidades produciendo evidencias sustanciales que la relación entre la usabilidad y la arquitectura del software va mas allá de separación de la interfaz de usuario y las funcionalidades básicas.

Por otro lado, los investigadores del proyecto STATUS [18] también han analizado la relación entre usabilidad y la arquitectura de software. Ellos utilizaron una línea de razonamiento para describir, en base a su experiencia, cómo algunas cuestiones de usabilidad están relacionadas con la arquitectura de software. Por ejemplo, para la característica de cancelación, los componentes que procesan acciones necesitan poder interrumpirse y las consecuencias de las acciones volverse atrás [6, 7]. Así, se ilustran intuitivamente las relaciones entre las características de usabilidad y el diseño del software.

Con posterioridad, las investigaciones avanzaron un paso más adelantando las cuestiones de usabilidad no ya al momento de diseño arquitectónico, sino a la etapa de requisitos. Para incorporar la usabilidad en las primeras etapas del desarrollo, se ha propuesto una serie de guías para la captura de requisitos que se utilizan durante la educción de mecanismos de usabilidad y el proceso de especificación [13,19]. Su aproximación consiste en definir un conjunto de pautas que faciliten la labor de los analistas en la captura de los requisitos de usabilidad, independientemente de su conocimiento en el área. Estas guías ayudan al analista a entender las implicaciones y conocer cómo educir y especificar los mecanismos de usabilidad para un sistema software.

Adicionalmente, dichas investigaciones proveen datos sobre el impacto de incluir las recomendaciones de usabilidad en un sistema software [10,11]. Los datos recolectados muestran que la construcción de ciertos componentes de usabilidad en un sistema software implica cambios realmente significativos en el diseño.

IV. Metodología de Trabajo

Para llevar a cabo el análisis de la incorporación de los MUs en las diferentes etapas de desarrollo se utiliza la metodología orientada a objetos [20] basándose en los siguientes modelos:

Modelo de requisitos: el modelo de requisitos se expresa como modelo de casos de uso.

Modelo de diseño: las funcionalidades de los casos de uso se expresan en diagramas de clase y de secuencia.

Modelo de implementación: los casos de uso se instrumentan bajo la arquitectura web MVC2.

En la Figura 1 se muestra el esquema seguido para realizar el an´alisis sistematizado de la incorporación de MUs. El sistema tomado como caso de estudio automatiza la tramitación de proyectos de ley en el Poder Legislativo del Congreso Nacional Paraguayo y se desarrolla desde la perspectiva de dos desarrolladores (A y B), uno con experiencia previa y otro sin experiencia alguna en la aplicación de MUs respectivamente. Los artefactos para la incorporación de la usabilidad son proporcionadas por el Experto. El procedimiento fue como sigue: (a) Cada desarrollador recibe la definición del problema junto con el alcance del sistema y procede a realizar las actividades propias del desarrollo (a) Al final de cada etapa del desarrollo entregamos a cada desarrollador los artefactos de usabilidad

(b) Los desarrolladores introducen las características de usabilidad haciendo uso de los artefactos proporcionados (c) Finalmente, evaluamos el impacto de dicha incorporación registrando datos en relación a las funcionalidades del sistema y las relacionadas a la usabilidad.

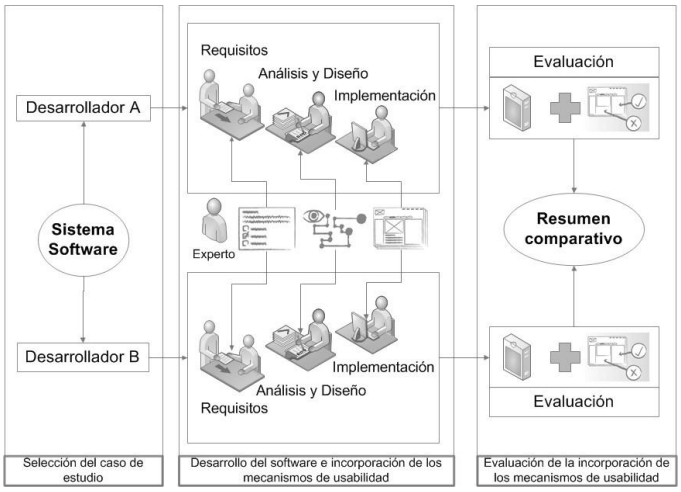

Figura 1. Incorporación y evaluación de los mecanismos de usabilidad en el desarrollo

Los productos obtenidos por cada desarrollador durante el proceso de desarrollo evolucionan con la incorporación de la usabilidad. Por ejemplo, el documento de requisitos preliminar se transforma en un documento de requisitos final que contiene la incorporación de los mecanismos. De manera similar, en el diseño, se transforman los diagramas y se mide el impacto que implica dicha transformación y el incremento en código en términos de programación.

V. Incorporación de MUs en las tapas de Desarrollo

El proceso de desarrollo comprende tres etapas iniciando por la educción y especificación de requisitos, pasando por el an´alisis y diseño hasta la implementación. Los productos obtenidos en cada etapa (ERS, diagramas, código fuente) evolucionan con las características de usabilidad. Los desarrolladores, haciendo uso de las recomendaciones proporcionadas por las PDMUs, introducen transfor- maciones en los distintos productos de cada etapa para converger a un sistema software con niveles deseables de usabilidad.

Educción Y Especificación de Requisitos

El punto de partida es la elaboración del documento de los requisitos propios del sistema. Dos desarrolladores A y B participan en esta etapa y cada desarro- llador produce su documento de requisitos (ERS4). La comprensión y definición de las funcionalidades o requisitos del sistema no se logran de una sola vez, sino que se produce por mejora iterativa. Así, varias sesiones de educción de requisitos son requeridas para producir la descripción estable de una funcionalidad. Con las versiones estables de los requisitos de ambos desarrolladores extraemos aquellas funcionalidades comunes en las cuales se realizará la incorporación de los MUs. La información necesaria sobre los mecanismos es capturada gracias a un conjunto de preguntas que proporcionan las guías para la educción de requisitos de usabilidad [13, 19]. Estas guías se encuentran en las PDMUs [15]. La Figura 2 muestra un fragmento de ERS de un caso de uso del desarrollador A donde se ve la incorporación de los MUs *Abort Operation* y *Progress Feedback* en textos resaltados; negrita-cursiva (flujo alternativo) y negrita-subrayado (flujo básico) respectivamente. Debido a que el caso de estudio, se basa en la reingeniería de un sistema existente, se han incluido elementos referentes a diseño de pantalla en la ERS.

Caso de Uso: Derivar Proyecto	
Resumen: Derivar el estudio de un proyecto a las comisiones.	
Pre-condiciones: • El usuario ha ingresado en el sistema y está en Asuntos Entrados.	
Post-condiciones: • El Proyecto queda pendiente en las comisiones donde fue derivado y desaparece de la lista de tareas pendientes del usuario Asuntos Entrados.	
Flujo Básico de Eventos	
Usuario	**Sistema**
1. El usuario ingresa al listado de pendientes.	2. El sistema lista los proyectos pendientes del usuario. Se muestra el nro. de expediente, acápite, etapa, subetapa y fecha. **Mientras se procesa la consulta se muestra una barra de progreso indicando la tarea activa y finaliza con un mensaje indicando la cantidad de registros recuperados.**
3. El usuario selecciona un proyecto para derivar a la/s comisión/es.	4. El sistema despliega la lista de comisiones.
5. El usuario selecciona la/s comisión/es.	6. El sistema valida la selección de la/s comisión/es, almacena la información asignando la etapa y la subetapa de la siguiente tramitación. Además, saca el proyecto de los pendientes del usuario. 7. El sistema informa al usuario que la operación se realizó de manera satisfactoria y muestra el listado de pendiente.
Flujos Alternativos	
3.1. El usuario selecciona el proyecto para visualizar su detalle y el sistema despliega la información del proyecto a ser derivado. *5.1. El usuario selecciona una comisión y* • *selecciona otra opción de menú. El sistema detecta el cambio de estado y solicita confirmación al usuario para guardar los cambios:* ○ *Si se aceptan los cambios, el sistema almacena los cambios y dirige la acción al menú seleccionado. Si ocurre algún error durante la actualización de los datos, el*	

Figura 2. Fragmento de ERS de un caso de uso con los MUs *Abort Operation* y *Progress Feedback*

ANÁLISIS Y DISEÑO

La etapa del análisis y diseño inicia a partir del documento de requisitos del sistema con la usabilidad integrada. En esta etapa los mecanismos de usabilidad adoptan la formalización de patrones de usabilidad contenidos en las PDMUs [15]. Cada desarrollador, haciendo uso de los requisitos obtenidos, elabora el diseño basándose en diagramas UML de clases y de secuencias. Estos diagramas evolucionan con la incorporaci´on de los MUs. La integración de la usabilidad se realiza analizando previamente el patrón de diseño propuesto, identificando los conectores necesarios para el acople con el diagrama de clases propio del sistema elaborado por el desarrollador. El resultado de la incorporación de MUs en el diagrama de clases elaborado por el desarrollador A se visualiza en el fragmento de la Figura 3. Los nuevos diagramas que aparecen sombreados corresponden a los diseños de los mecanismos de usabilidad, cada uno diferenciado por un tipo de letra distinto.

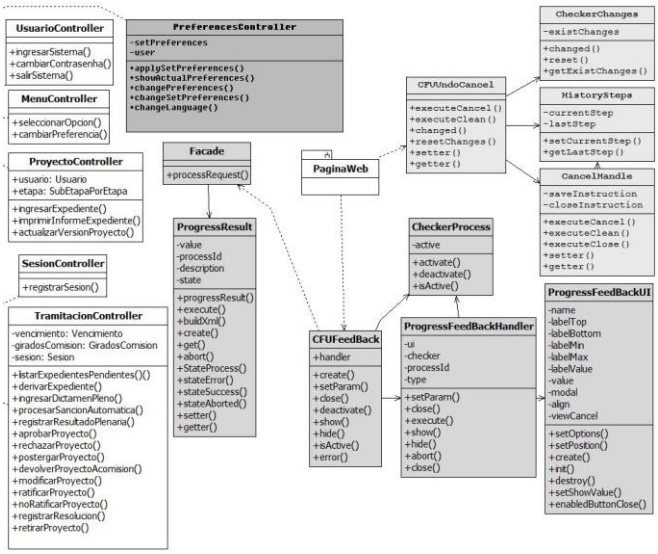

Figura 3. Fragmento del diagrama de clases con los MUs incorporados por el desarrollador A

IMPLEMENTACIÓN

Tomando como punto de partida la especificación de las clases elaboradas durante el diseño se procede a describir esas clases de forma detallada y en el lenguaje de programación deseado, en este caso *Java*. Posteriormente cada desarrollador incorpora iterativamente cada MU siguiendo el patrón de programación equivalente en ese lenguaje. Así, ambos desarrolladores A y B inician esta etapa con la definición de la estructura de almacenamiento mediante el diagrama entidad-relación (ER5). La construcción del sistema se realiza bajo la arquitectura de tres capas (MVC). Cada capa se implementa siguiendo las convenciones de código para aplicaciones *Java* basadas en web (*Java Server Faces* o JSF). Con las funcionalidades codificadas cada desarrollador introduce los mecanismos de usabilidad haciendo uso de los patrones contenidos en las PDMUs [15]. En la Figura 4 se muestra un fragmento del código de usabilidad añadido a una funcionalidad del sistema para el MU *Progress Feedback* por el desarrollador A resaltado en recuadro sombreado junto con su representación UI. Tenga en cuenta que los códigos de usabilidad contenidos en las PDMUs pueden ser introducidos a nivel de formularios UI, de clases intermedias (sesiones, controladores) e incluso como estructuras adicionales representadas por tablas en la base de datos.

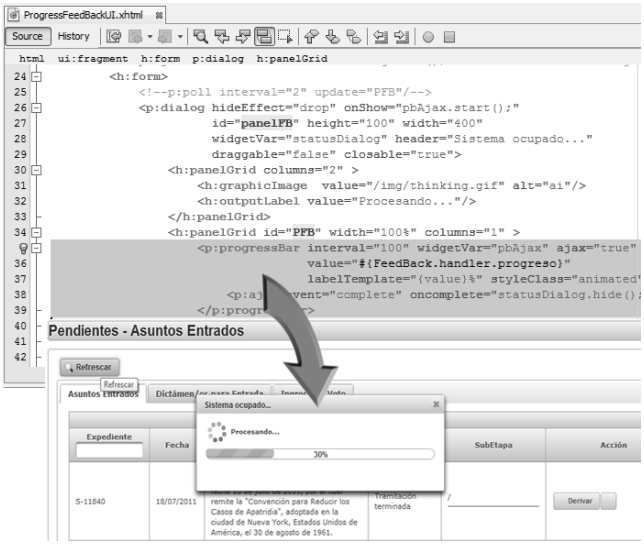

Figura 4. Fragmento de código JSF con el MU *Progress Feedback* incorporado por el desarrollador A

VI. Aplicación de las PDMUs y Evaluación del Impacto

La incorporación de MUs produce cambios en los productos de cada etapa susceptibles de ser evaluados. Estos cambios son expresados en términos cuantitativos (p. ej. incrementos en los productos de cada etapa) y cualitativos (p. ej. percepción del esfuerzo de comprensión e incorporación de los artefactos asociados a cada MU).

Educción y Especificación de Requisitos

Durante la etapa de requisitos se ha trabajado con las guías de educción de mecanismos de usabilidad [19], [15]. Cada desarrollador ha realizado una lectura rápida de la información de cada mecanismo de usabilidad para tener una idea del problema al cual apunta. Dentro de las apreciaciones percibidas por los desarrolladores para considerar la usabilidad en esta etapa se destacan el lenguaje directo, claro y adaptable en la que están formuladas las cuestiones de usabilidad en las mencionadas guías. Igualmente, las preguntas apuntan hacia aspectos reales de la usabilidad materializ´andose posteriormente en artefactos concretos (requisitos, diseño, código, etc.). Por el contrario, la utilización de las guías requiere conocer previamente el dominio del problema y algunos escenarios ejemplos de usabilidad a fin de discutir, adaptar o recomendar su utilidad en aquellas funcionalidades del sistema potencialmente candidatas. Para cuantificar el impacto de los mecanismos de usabilidad en la educción de requisitos hemos seguido el análisis de impacto de la usabilidad en el diseño software realizado en [12] donde se calculan el porcentaje de casos de uso que se expanden como consecuencia de los MUs. Las unidades son bajo, medio y alto dependiendo en que intervalo caen (por debajo de 33 %, entre 33 % y 66 %, por encima de 66 %).

En la Tabla 1 se muestra una evaluación semejante aplicada al sistema tomado como caso de estudio y el análisis ha revelado un nivel de impacto medio-alto de los mecanismos *Abort Operation* y *Progress Feedback* quedando el *Preferences* como el de menor grado. Sin embargo, pese a la diferencia en la cantidad de funcionalidades entre desarrolladores (28 y 36), el nivel de impacto obtenido tiende hacia las mismas proporciones permaneciendo el mecanismo de usabilidad *Abort Operation* en el nivel más alto. Esta situación proporciona una idea del probable retrabajo que recaería en la implementación del MU *Abort Operation* si se pospone la usabilidad en las etapas finales del desarrollo. También, con los resultados obtenidos se validan los estudios realizados previamente sobre el análisis del impacto de la usabilidad en el diseño del software [12].

TABLA 1. Impacto de los mecanismos de usabilidad en los requisitos

MUs	Nivel de impacto A	Nivel de impacto B
Abort Operation	21/28 – Alto 75 %	23/36 – Medio 64 %
Progress Feedback	13/28 – Medio 46 %	12/36 – Medio 33 %
Preferences	1/28 – Bajo 4 %	1/36 – Bajo 3 %

Análisis y Diseño

Los resultados y reacciones manifestadas por los desarrolladores en esta eta- pa subrayan la necesidad de mejora en los artefactos contenidos en las PDMUs incluyendo indicaciones concretas en lo que respecta a su utilización. Si bien las PDMUs proveen diseños preliminares reutilizables para introducir la usabilidad en los diagramas, estos diseños carecen de suficiente claridad y expresividad para utilizarlo directamente requeriendo un análisis exhaustivo previo y en ocasiones, la asistencia del experto en usabilidad por parte de los desarrolladores. La evaluación del impacto de la incorporación de los MUs en esta etapa ha revelado resultados interesantes. Independientemente de la pericia del desarrollador, la incorporación de los tres MUs en el diagrama de clases (DC) muestra un bajo nivel de impacto (Tabla 2). Es aceptable pensar en el bajo nivel de impacto de todos los MUs porque los diagramas de clases solo describen la estructura del sistema mostrando sus clases, atributos y las relaciones entre ellos sin interacción alguna. Por otro lado, en la misma tabla, la incorporación del MU *Abort Operation* y *Progress Feedback* en el diagrama de secuencia (DS) señala una complejidad de interacciones media-

baja. Para el caso del *Preferences*, se ha obtenido un nivel de impacto bajo en los diagramas de secuencia. Esta evaluación muestra, aunque ilustrativamente, que el *Abort Operation* supone una complejidad importante en la implementación, a excepción del *Progress FeedBack* y *Preferences*.

TABLA 2. Impacto de los mecanismos de usabilidad en el diseño

MUs	Desarrollador A		Desarrollador B	
	D	D	D	D
Abort Operation Progress Feedback Preferences	Bajo 10% Bajo 15% Bajo 7%	Medio 35% Bajo 9% Bajo 2%	Bajo 8% Bajo 10% Bajo 4%	Medio 47% Bajo 18% Bajo 1%

IMPLEMENTACIÓN

Tras la incorporación de los distintos MUs en el código del sistema, hemos observado interesantes posturas adoptadas por los desarrolladores A y B ante los MUs formalizados como patrones de programación de usabilidad. Por un lado, los artefactos propuestos para el MU *Preferences* resultaron más comprensibles e intuitivos que el MU *Abort Operation* y el MU *Progress Feedback* debido al reducido número de escenarios y un nivel de interacción casi despreciable con el sistema implementándose nuevas funcionalidades con mínimas modificaciones en las existentes. Por otro lado, la existencia de escenarios interconectados en el MU *Abort Operation* y la comprensión de los mismos ha generado interrogantes al desarrollador en lo que respecta al esfuerzo/beneficio del mecanismo ya que el sistema tomado como caso de estudio carece de funcionalidades consideradas críticas. A pesar de que el MU *Progress Feedback* no posee escenarios interconectados se ha observado una reacción similar al MU *Abort Operation* en los desarrolladores A y B, porque obtener la información de avance exige el control constante del proceso junto con sus interacciones y las consultas implementadas en el sistema resultaron relativamente simples no justificando el esfuerzo de implementar tal mecanismo.

Los datos resultantes de la incorporación de los MUs en esta etapa revelan un bajo nivel de impacto independientemente del mecanismo y del desarrollador. Más detalladamente, en la Figura 5, los segmentos situados al tope de cada pila evidencian que los MUs *Abort Operation* y *Progress Feedback* tienen un impacto similar en términos de incremento de nuevas clases en la codificación y comparativamente un mayor incremento es señalado por el MU *Preferences*. Tenga en cuenta que los incrementos derivados del *Abort Operation* y *Progress Feed- back* se acentúan cuanto más funcionalidades requieran estos dos mecanismos, no así el *Preferences* cuyos incrementos en artefactos permanecen inalterables con el número de funcionalidades. La cuantificación en término de incremento en métodos tiende hacia las mismas proporciones (Figura 6).

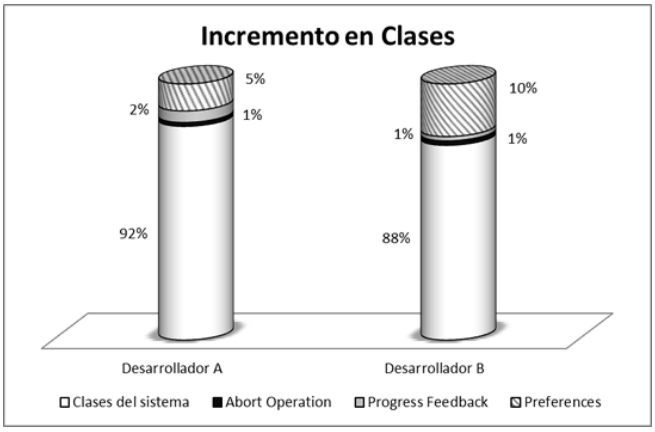

Figura 5. Nivel de impacto en Clases por MU de los desarrolladores A y B

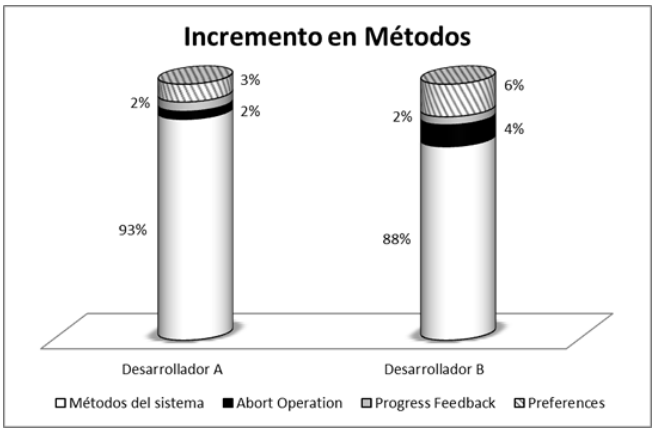

Figura 6. Nivel de impacto en Métodos por MU de los desarrolladores A y B

VII. DISCUSIÓN DE RESULTADOS

A nivel global, los resultados obtenidos revelan que el esfuerzo de dotar a un sistema con características de usabilidad se distribuye a lo largo de todo el proceso, atenuando la sobrecarga de trabajo que implicaría tener en cuenta estas cuestiones de usabilidad únicamente en la etapa final del desarrollo. Esta situación conlleva beneficios al productor de software y al usuario. Entre ellas, considerando ya las cuestiones de usabilidad desde sus inicios hemos evidenciado que la documentación asociada al producto software es completa y menos susceptible a errores reduciendo costos, por ejemplo, reimpresión y distribución de manuales. También, como los desarrolladores han utilizado unos artefactos de

usabilidad no se necesita la intervención de profesionales HCI o ingenieros especializados para los proyectos software donde requieran niveles deseables de usabilidad. Trasladándonos al ámbito del usuario, el producto software con niveles deseables de usabilidad introducirá mejoras en los costos de soporte y entrenamiento. En este sentido, cuando un producto software es entendible y manejable el entrenamiento será mínimo no requiriendo soportes frecuentemen-te. Asimismo, es de esperar que la usabilidad mejore la productividad ya que el usuario sentirá dominio del sistema estableciendo sus configuraciones personales, teniendo el control de sus acciones y estando informado del avance de las operaciones que realiza. En cuanto a los artefactos de usabilidad utilizados en este trabajo, podemos afirmar que las soluciones proporcionadas en las PDMUs resultan bastante prometedoras. Estas soluciones están definidas como artefactos concretos para el tratamiento de la usabilidad desde las etapas tempranas del desarrollo. Dado que se trata de una propuesta preliminar, muchos de los artefactos contenidos son potencialmente reutilizables y seguirán en mejora. Aún cuando el proceso inicial para el empleo de los diferentes artefactos requiere una inversión de tiempo importante, el aprendizaje adquirido como práctica inicial de usabilidad queda materializado en la experiencia del desarrollador permitiendo la adaptación rápida de la solución para futuros problemas, que en la mayoría de los casos, resulta en una suerte de *copy & paste*.

Es fácil notar que ciertos mecanismos de usabilidad se comportan mejor en problemas específicos y plataformas concretas. Por ejemplo, el MU *Abort Operation* es plenamente justificable cuando los cambios de estados en el sistema afectan datos sensibles y necesitan que el usuario sea alertado cuando realiza alguna acci´on con ellos. Por el lado del *Preferences*, este mecanismo encaja en sistemas que requieren variedad de configuraciones y preferencias particulares por usuario como colores, letras, tamaños e idiomas. En tanto que en sistemas de consultas masivas y procesamiento en lotes resulta esencial el empleo del MU *Progress Feedback* para informar al usuario sobre el avance de una tarea siempre y cuando la plataforma permita dicha operación. Sea cual fuere el caso, el trabajo adicional de la usabilidad estará ligado a la naturaleza del sistema y las decisiones del usuario.

VIII. Conclusiones Y Líneas Futuras

Una de las dificultades que gira en torno a la usabilidad es el costoso impacto de considerarlo al final del desarrollo. Con esta investigación hemos evidenciado que la tarea de incorporar usabilidad desde las etapas iniciales reduce este impacto ya que toda la sobrecarga de trabajo se distribuye a lo largo de las etapas evitando rehacer el sistema al final del proceso para obtener un producto software usable. Teniendo en cuenta la usabilidad desde las etapas iniciales del desarrollo, las características de usabilidad se ven plasmadas en los documentos generados en las distintas etapas. Así toda la documentación asociada al producto software esta actualizada, completa y menos susceptible a errores. Sin embargo si la usabilidad queda postergada al final del proceso, por lo general la documentación queda relegada y por ende desactualizada. Gracias a los artefactos de usabilidad estudiados en este trabajo no se requirió de profesionales HCI o Ingenieros en usabilidad. Los desarrolladores utilizando esos artefactos o patrones han sido capaces de dotar de usabilidad al producto software. Con esto también se reduce costos de contratar profesionales HCI para obtener proyectos software con niveles deseables de usabilidad. Anticipamos una mejora en productividad, entrenamiento y soporte porque al entregar un producto software entendible y manejable el entrenamiento al usuario será mínimo, no requerirá soportes frecuentemente y sus tareas serán ejecutadas eficientemente.

Adicionalmente destacamos otras aportaciones:

Experiencia en desarrolladores con distintos niveles de usabilidad. Hemos realizado la evaluación del impacto y esfuerzo de introducir la usabilidad en el proceso de desarrollo desde la perspectiva de dos desarrolladores con distintas habilidades en usabilidad.

Mejoras en los artefactos asociados a los mecanismos de usabilidad. Las mejoras incluyen la restructuración de las pautas, los escenarios de aplicación y los patrones de programación en *Java*. Estas mejoras facilitan la utilización y comprensión de las pautas para futuros casos.

Datos cuantitativos y cualitativos. El análisis sistematizado de la usabilidad aporta datos acerca del impacto y esfuerzo durante el desarrollo. Los datos complementan la evaluación y proveen informaciones importantes al desarrollador que desea trabajar con características funcionales de usabilidad.

Con este análisis permanecen abiertas varias líneas de investigación. Las siguientes líneas parecen prometedoras:

Ampliación del espectro de casos de los MUs. La cobertura de más casos pro- mueve la agregación de

más datos y mejoras en los artefactos para alcanzar una solución reutilizable e integrable en librerías o *plugins*.

Evaluación aplicada a otra metodología y proyecto. La utilización de otras metodologías y otros proyectos extenderá la aplicación de las PDMUs en proyectos de desarrollo similares.

Análisis del impacto utilizando otras métricas. Siguiendo un marco de referencia similar, abordar otro conjunto representativo de métricas como Factor de acoplamiento (*Coupling Factor –CF–*), Líneas de Código (*Lines of Code per method –LOC–*), entre otros, aportará un complemento válido al análisis y al mejoramiento de los artefactos de la usabilidad.

Estudio de costos adicionales. Los datos cuantificados en esta investigación proporcionan una base para realizar una estimación de los costos adicionales derivados en un proyecto software considerando a la usabilidad como aspecto relevante.

Finalizando esta investigación, resaltamos la importancia que va adquiriendo la usabilidad en el proceso de desarrollo del software con el fin de obtener sistemas software cada vez más usables. El presente trabajo se encamina en esa dirección.

REFERENCIAS

1. Barbacci, M. R., Ellison, R., Lattanze, A. J., Stafford, J. A., Weinstock, C. B., and Wood, W. G. Quality Attribute Workshops, Third Edition, 2003.
2. Bass, L., and John, B. E. Linking usability to software architecture patterns through general scenarios. *Journal of Systems and Software 66*, 3 (2003), 187–197.
3. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., Macleod, G. J., and Merrit, M. J. *Characteristics of Software Quality*. TRW series of software technology. North-Holland, 1978.
4. Bosch, J., and Juristo, N. Designing software architectures for usability. *25th International Conference on Software Engineering, 2003. Proceedings. 3-10 May 2* (2003), 757–758.
5. Cysneiros, L. M., and Kushniruk, A. Bringing Usability to the Early Stages of Software Development Usability Ontology References. *Engineering* (2003).
6. De Andrés, A., Bosh, J., Charalampos, A., Chatley, R., Ferre, X., Folmer, E., Juristo, N., Magee, J., Menegos, S., and Moreno, A. M. Usability Attributes Affected by Software Architecture. Deliverable. 2, 2010.
7. Folmer, E., and Bosh, J. Architecting for Usability; a Survey. *Systems and Software 70*, 1-2 (2004), 61–78.
8. Golden, E. *Early-Stage Software Design for Usability*. PhD thesis, Carnegie Mellon University, 2010.
9. IEEE Std 1061. IEEE Standard For A Software Quality Metrics Methodology Revision And Reaffirmation. *Proceedings of IEEE International Symposium on Software Engineering Standards 1998*, IEEE Std 1061-1998 (1998), 278–278.
10. Juristo, N. Impact of Usability on Software Requirements and Design. *Software Engineering International Summer Schools ISSSE 2006-2008* (2009), 55–77.
11. Juristo, N., and Moreno, A. Moving usability forward to the beginning of the software development process. *Human Computer Interaction*, Bass 1999 (2005).
12. Juristo, N., and Moreno, A. A Glass Box Design: Making the Impact of Usa- bility on. *Ifip International Federation For Information Processing Part II* (2007), 541–554.
13. Juristo, N., Moreno, A., and Sanchez-Segura, M.-I. Guidelines for Eliciting Usability Functionalities. *IEEE Transactions on Software Engineering 33*, 11 (Nov. 2007), 744–758.
14. Panach Navarrete, J. I. *Incorporación de Mecanismos de Usabilidad en un Entorno de Producción de Software Dirigido por Modelos*. PhD thesis, Universidad Politécnica de Valencia, 2010.
15. Rodríguez, F. D. Programming Patterns for usability functionalities, 2010.
16. Rodríguez, F. D., and Acuña, S. T. Implementación de una Solución Reuti- lizable para una Funcionalidad de Usabilidad. *XVII Jornadas de Ingeniería del Software y Bases de Datos* (2012), 14.
17. Seffah, B. A., and Metzker, E. The Obstacles and Myths of Usability and Software Engineering. *Communications of the ACM 47*, 12 (2004), 70–76.
18. STATUS. Software Architecture that supports Usability (STATUS), 2012.
19. USEP. Usability Elicitation Patterns (USEPs), 2006.
20. Weitzenfeld, A. *Ingeniería de Software Orientada a Objetos con UML, Java e Internet*, 1ra ed. ed. Thompson, 2005.