ICT-UNPA-139-2016
ISSN: 1852-4516
Aprobado por Resolución N° 0561/16-R-UNPA

Framework Orientado a Aspectos de Recopilación Automática de Datos para la Evaluación de Usabilidad en Aplicaciones Web

Roberto Farias, Sandra Casas
{rfarias, scasas}@unpa.edu.ar
Grupo de I+D en Ingeniería de Software Pragmática
Instituto de Tecnología Aplicada - Unidad Académica Río Gallegos
Universidad Nacional de la Patagonia Austral

Resumen: La Usabilidad consiste de un conjunto de atributos que permiten evaluar el esfuerzo que deberá invertir un usuario para realizar una tarea determinada a través de la utilización de un software específico. En este sentido, una aplicación se considerará más usable cuanto menos esfuerzo requiera para su utilización.

En la actualidad, la World Wide Web, se ha constituido como un enorme repositorio de información que es presentada y puesta a disposición de los usuarios a través de diversos sitios o páginas web cuya finalidad es promover el acceso a la información y de esta manera, garantizar la igualdad de oportunidades a quienes la consumen. Por otro lado, la evolución de las tecnologías que le dan soporte, la han convertido en una plataforma tecnológica sobre la cual es posible desarrollar aplicaciones con un nivel de interacción y funcionalidad similar al de las aplicaciones de escritorio (WIMP).

Existen una gran variedad de trabajos sobre usabilidad web, centrados principalmente en aspectos tales como la navegación y la arquitectura de información de los sitios o páginas web. Sin embargo, la usabilidad de las aplicaciones web, que experimentan un alto grado de interacción con el usuario (action-oriented) similar al de las aplicaciones de escritorio, no puede ser evaluadas de la misma manera que en un sitio web (information-oriented) donde la interacción con el usuario es escasa y limitada. Si bien pueden encontrarse una diversidad de trabajos que tratan como objeto de estudio a la usabilidad en aplicaciones, en entornos de escritorio ó moviles, no se han reconocido trabajos referidos a la usabilidad de aplicaciones en entornos web.

En este trabajo, se propone un framework basado en AOP, capaz de dar soporte al proceso de evaluación de usabilidad, en aplicaciones web, durante la ejecución de tareas de usuario.

Palabras Claves: Usabilidad; Programación Orientada a Aspectos; Aplicaciones Web; Framework.

ICT-UNPA-139-2016
ISSN: 1852-4516
Aprobado por Resolución N° 0561/16-R-UNPA

Aspect-Oriented Framework of Automatic Data Collection for the Evaluation of Usability in Web Applications

Roberto Farias, Sandra Casas
{rfarias, scasas}@unpa.edu.ar
Grupo de I+D en Ingeniería de Software Pragmática
Instituto de Tecnología Aplicada - Unidad Académica Río Gallegos
Universidad Nacional de la Patagonia Austral

Abstract: The Usability consists of a set of attributes that allow evaluating the user effort to perform a certain task through the use of a specific software. Therefore, a software application is considered usable when it requires the least possible effort.

Actually, the World Wide Web, has become in a great repository of information that is presented and offers to users through various websites or pages whose purpose is to ensure access to information and thus, ensure equality of opportunities. On the other hand, the evolution of the technologies, that support it, have converted the World Wide Web in a technological platform on which applications can be developed with a level of interaction and functionality like to desktop applications (WIMP).

There are a variety of works about web usability, mainly they are focused on aspects such as the navigation and the information architecture of sites or web pages. However, the usability of Web applications, whose experience includes high degree of interaction with the user (action-oriented) like to desktop applications, can't be evaluated in the same way as a website (information-oriented), where the interaction is low and limited. While there are a variety of works, whose object of study is the usability in applications of mobile or desktop environments, works related to the usability of web applications have not been recognized .

In this paper, a framework based on AOP, that supports the process of evaluating of usability of web applications during execution of user tasks, is proposed.

Keywords: Usability; Aspect Oriented Programming; Web applications; Framework.

1. INTRODUCCIÓN

La usabilidad (Nielsen J. 1993) es una disciplina que procura la mejora en el uso del software, y ha sido reconocida como un aspecto relevante en la calidad del software.

El proceso de evaluación de la usabilidad web, puede orientarse a los sitios web ó a las aplicaciones web. Los sitios web, tienen por finalidad principal comunicar o poner a disposición de los usuarios, la información, de una manera clara, accesible y sencilla. Bajo esta orientación, la evaluación de usabilidad se centra en aquellos criterios o aspectos que afectan la navegación, la facilidad de comprensión, y la accesibilidad de la información. En este sentido, un sitio web usable será aquel que brinde soporte a las tareas o actividades que puede realizar un usuario (navegación, búsqueda, y recuperación de información y servicios), con efectividad, eficiencia, y satisfacción.

Por otro lado, las aplicaciones web, al igual que las aplicaciones de escritorio, se encuentran orientadas a dar soporte a los usuarios en la realización de una tarea específica. Este tipo de software, que utiliza la web como plataforma de ejecución y de servicios, se caracteriza por un alto grado de interacción con el usuario a través de una interfaz accesible por un navegador web. Bajo esta perspectiva, la usabilidad, debería evaluar aquellos aspectos que inciden en la efectividad, eficiencia y satisfacción con que el usuario logra realizar las tareas que se propone. Así mismo, dada la gran proliferación de aplicaciones web, en la actualidad, el proceso de evaluación y desarrollo de la usabilidad en este tipo de software ha comenzado a tener relevancia para muchas organizaciones que la consideran como una de las propiedades fundamentales para garantizar su éxito.

En la bibliografía actual, es posible encontrar numerosos trabajos en los que se proponen diversos métodos, o incluso combinaciones de ellos, para su evaluación automática. Sin embargo, estos trabajos se orientan a la evaluación de usabilidad en aplicaciones de software en entornos de escritorio ó en entornos móviles. No se han encontrado trabajos que, partiendo de las particularidades de los ambientes de ejecución web, realicen aportes para brindar soporte al proceso de evaluación automática de la usabilidad en aplicaciones web.

Una de las etapas más importantes y críticas del proceso de evaluación de la usabilidad, es la recolección de datos, ya que tiene impacto directo en el análisis y evaluación de la usabilidad. En las aplicaciones web, una de las técnicas de registro más utilizadas, es la de usar archivos de logs. Si bien, a través de esta técnica se recopila una gran cantidad de datos referidos al comportamiento de los usuarios, gran parte de esta información resulta irrelevante para los evaluadores de usabilidad. Por otro lado, dado que los archivos de log no están pensados para satisfacer las necesidades de información del proceso de evaluación de usabilidad, es necesario realizar un proceso de limpieza y transformación de los datos para facilitar su análisis e interpretación, aún así, puede que exista un conjunto de datos relevantes no registrados. Se observa la necesidad de aplicar enfoques que permitan recolectar datos, específicos y significativos, durante la ejecución de tareas de usuario en aplicaciones web, que sean lo suficientemente flexibles para ser reutilizados en diversas aplicaciones con arquitecturas y características similares.

En este trabajo, se propone un framework basado en AOP (Kiczales G. et al. 1997), capaz de recolectar, de manera automática, datos relevantes para la evaluación de usabilidad, en aplicaciones web, durante la ejecución de tareas de usuario.

La estructura de este trabajo es como sigue, la Sección 2 desarrollan las nociones básicas y el marco teórico en torno a los siguientes temas principales: Usabilidad, Aplicaciones Web, Programación Orientada a Aspectos (AOP), y Framework. La Sección 3 describe el problema identificado en la evaluación de usabilidad de aplicaciones web donde se carece de herramientas de soporte para su evaluación automática. La Sección 4, presenta la estructura del framework propuesto y describe cada uno de sus componentes principales, al respecto, en Sección 5, analiza los patrones de diseño que fueron implementados, en su desarrollo, en cuanto al problema o situación que cada uno trata de resolver, y la forma en que fueron aplicados. La Sección 6, presenta el caso de estudio empleado para analizar y evaluar el framework propuesto. La Sección 7 presenta las conclusiones alcanzadas.

2. NOCIONES PRELIMINARES

2.1. Usabilidad

La norma ISO 9126, publicada en 1992 con el título "Information technology –Software product evaluation: Quality characteristics and guidelines for their use" y en la cual se establecen las características de calidad para productos de software, la Usabilidad se define como la capacidad del producto software para ser entendido, aprendido, usado y ser atractivo para el usuario cuando se usa bajo condiciones específicas (ISO 2000).

Otro estándar internacional que refiere a la Usabilidad, es la norma ISO 9241 titulada: *"Ergonomic Requirements for Office Work with Visual Display Terminals"*. Este estándar, define la usabilidad como el grado en que un producto puede ser utilizado por usuarios específicos para lograr sus objetivos con efectividad, eficiencia y satisfacción en un contexto de uso definido. En esta definición, la efectividad se refiere a la exactitud y completitud con la que los usuarios logran sus objetivos, la eficiencia se refiere a los recursos empleados en relación con la exactitud y completitud con la que los usuarios logran sus objetivos, y la satisfacción se refiere al confort y la aceptabilidad del uso del sistema (ISO 1998). Atendiendo a esta definición, los problemas de usabilidad son aquellos que hacen que una aplicación sea inefectiva, ineficiente, y difícil de aprender y usar.

Jackob Nielsen, define la Usabilidad como el *grado de efectividad de la interacción entre operadores y sus máquinas*. Este autor, indica que no se trata de una simple propiedad unidimensional asociada a la interfaz de usuario. Al contrario, la Usabilidad, posee múltiples componentes y se encuentra tradicionalmente asociada con cinco atributos (Nielsen J. 1993):

- Facilidad de aprendizaje: Los sistemas deberían ser fáciles de aprender de modo que el usuario pueda, rápidamente, comenzar a trabajar con el sistema.

- Eficiencia: Los sistemas deberían ser eficientes al usarlos, de modo tal que una vez que el usuario haya aprendido a utilizar el sistema en cuestión, puede alcanzar un alto nivel de productividad.

- Fácil de recordar (memoria en el tiempo): La funcionalidad debería ser fácil de recordar, de tal modo que un usuario casual, luego de transcurrido un periodo de tiempo, sea capaz de volver a emplear el sistema sin tener que aprender todo nuevamente.

- Pocos errores: Los sistemas deberían tener una baja tasa de error de tal modo que los

usuarios cometan pocos errores durante el uso del sistema y, en caso cometerlos, puedan recuperarse fácilmente de los mismos. Por lo tanto, no deben ocurrir errores catastróficos.

- Satisfacción del usuario: El uso del sistema debería ser placentero. Los usuarios, deben estar subjetivamente satisfechos cuando usan el sistema (les debe gustar).

Estos principios, a su vez, pueden ser especializados y descompuestos en criterios de grano más fino que pueden ser verificados a través de diferentes métodos de evaluación.

Los atributos o principios identificados por Nielsen, para la usabilidad, así como también las dimensiones de usabilidad propuestas por la ISO 9241, son conceptos abstractos y no pueden ser directamente medidos. Para medirlos se les asocian distintas métricas, por ejemplo, el atributo eficiencia puede ser evaluado mediante la métrica que calcula el tiempo empleado por un usuario en terminar una tarea específica.

Formalmente, una métrica (medida) es un valor numérico o nominal asignado a características o atributos de un objeto computado a partir de un conjunto de datos observables y consistentes con la intuición (DeMarco T. 1986). De acuerdo a Tahir A. y Ahmad R. (2010), pueden ser estáticas ó dinámicas. Las estáticas son utilizadas para medir las características estáticas de una aplicación, como el tamaño del código o la complejidad del mismo. Las dinámicas permiten medir el comportamiento de la aplicación y se calculan durante su ejecución.

En el marco de la evaluación de usabilidad de una aplicación web, las métricas dinámicas permitirán capturar datos y obtener información sobre la experiencia personal del usuario cuando hace uso de la misma.

Las cuatro formas básicas de evaluación son: automática (se calculan las métricas durante la ejecución de la aplicación), empírica (la usabilidad es evaluada testeando la aplicación con usuarios reales), formal (usando modelos formales y fórmulas para el cálculo de medidas de usabilidad), e informal (basados en reglas generales y la habilidad y experiencia de los evaluadores). Si bien, el conjunto de actividades involucradas en el proceso de evaluación dependerá sustancialmente del método empleado, en general se agrupan en tres pasos básicos (Ivory M. y Hearst M. 2001): Captura de datos, Análisis y Crítica**.**

2.1.1. Testeo con usuarios

En (Ivory M. y Hearst M. 2001), los métodos de evaluación de la usabilidad, se clasifican en torno a cinco clases: Testeo con usuarios, Inspección, Indagación, Modelo Analítico, y Simulación.

Los métodos de testeo con usuarios, permiten al evaluador contar con información acerca de cómo las personas usan las computadoras y los problemas que poseen con la interfaz de la aplicación que está siendo evaluada. Durante el test de usabilidad, los participantes usan la aplicación, o un prototipo, para realizar un conjunto determinado de tareas mientras el evaluador registra los resultados del trabajo de los participantes. Luego, los evaluadores, usan estos resultados para determinar lo bien que la interfaz favorece a la realización de las tareas de los usuarios, así como también, otras medidas como el número de errores y el tiempo de finalización de tareas. La automatización en este tipo de métodos, ha sido usada

principalmente de dos maneras: para la captura automática de datos referidos al uso de la aplicación, y para realizar el análisis automático de estos datos de acuerdo a métricas o modelos.

Algunos métodos de evaluación de Testeo con usuarios son: protocolo de pensamiento manifestado, protocolo de preguntas, método de seguimiento, método tutorado, método de instrucción previa, aprendizaje por descubrimiento conjunto, medición de desempeño, análisis de archivos de log, test restrospectivo, y testing remoto.

2.2. Sitios Web vs Aplicaciones Web

En la actualidad, la World Wide Web tiene un impacto significativo en el acceso a la gran cantidad de información disponible a través de internet. De hecho, es ampliamente utilizada en diversos dominios tales como el comercio, el marketing, la educación, e incluso el gobierno, en los cuales no sólo se facilita el acceso a la información, sino también a diversos servicios, donde ambos recursos se encuentran orientados a una gran variedad de usuarios que poseen diferentes características, conocimientos, e intereses.

Es importante notar, que a lo largo del tiempo, las tecnologías web han evolucionado a tal punto que es posible experimentar con páginas o sitios web con cierto grado de interacción (se pueden realizar comentarios, suscripciones a contenidos, compartir en redes sociales, o establecer contacto a través de un formulario sencillo). En este sentido, los usuarios comienzan a tener una participación más activa, al dejar de ser simples consumidores de información. Por otro lado, este auge de tecnologías emergentes, ha transformado a la web en una plataforma robusta, provista de servicios y utilidades, sobre la cual es posible crear aplicaciones complejas que pueden ser accedidas a través de un navegador web. Resulta de interés para el trabajo propuesto, establecer las diferencias más importantes entre sitios web y aplicaciones web (ver Tabla 1).

	SITIOS WEB	***APLICACIONES WEB***
Definición	Conjunto de páginas o documentos que pueden tener contenido multimedia embebido (imágenes, videos, etc.). Estas son accedidas a través de un navegador.	Es un aplicación que se accede a través de un navegador.
Objetivo	Information-oriented Ofrecer información a los usuarios.	Action-oriented Es un tipo de software que asiste a los usuarios en la realización de una tarea u actividad específica.
Contenido	Estático	Dinámico
Nivel de interacción	Bajo. La información de las páginas web es, normalmente, estática (sólo se puede leer, no interactuar con ella). El usuario navega por el contenido del portal a través de enlaces. En ocasiones el usuario puede realizar comentarios, acceder a un formulario de contacto o compartir contenido en redes sociales.	Alto. Se aprecia una interacción bidireccional en la cual el usuario solicita a la aplicación la ejecución de determinadas operación y aguarda una respuesta (solución, datos, ó información), esta respuesta puede así mismo motivar la ejecución de nuevas tareas por parte del usuario en la aplicación.
Tecnología básica	HTML, Javascript, CSS, Servidor Web	HTML, Javascript, CSS, Servidor Web, algún lenguaje de programación para implementar la lógica de negocio (Java, Phyton, PHP, C#, etc), y un motor de base datos para soportar el modelo de datos (Mysql, Postgres, etc.)
Aspectos de usabilidad, de mayor relevancia, que se evalúan en cada caso	Interfaz; Diseño centrado en el usuario; Experiencia del usuario; Terminología usada en la interfaz (Lenguaje); Adecuada gestión de errores; Arquitectura de la Información (Sistema de organización, Sistema de etiquetado, Sistema de navegación, Sistema de búsqueda)	Interfaz; Diseño centrado en el usuario; Experiencia del usuario; Terminología usada en la interfaz (Lenguaje); Adecuada gestión de errores; Mensajes de información pertinentes (alertas, errores); Feedback (el sistema debería informar constantemente al usuario sobre la tarea que se encuentra realizando); Ayuda y Documentación

Tabla 1. Principales diferencias entre sitio web y aplicación web

De acuerdo a (Matera M. et al. 2006), las tareas principales que ofrece un sitio web se basan en la publicación de contenidos y servicios (information-oriented):

- Encontrar servicios e información deseada a través de la búsqueda directa o, descubrir información y servicios, a través de la navegación.

- Comprender la información presentada.

- Invocar y ejecutar servicios específicos de ciertas aplicaciones Web, tales como la solicitud y descarga de productos.

A este conjunto de tareas, se puede incluir la posibilidad de realizar comentarios y compartir contenido en redes sociales.

Una aplicación web, por otro lado, es un sistema de software que usa la World Wide Web para interactuar con el usuario (action-oriented). Estas interacciones pueden ser basadas en transacciones (home banking, ecommerce), basadas en workflow (e-government, business-to-business), colaborativas (plataformas e-leaning, foros, wiki), entre otras.

El desarrollo de nuevas tecnologías ha permitido crear aplicaciones capaces de introducir en la web el paradigma de las aplicaciones de escritorio, es decir, desarrollar aplicaciones con dos características fundamentales (Voces M. 2011):

- Una alta interactividad con el usuario a través de multitud de elementos de interacción, que antes sólo eran viables en entornos de escritorio.

- Una alta velocidad de respuesta a la interacción del usuario. La unidad de información mínima es la que desee el desarrollador: una etiqueta, una tabla o quizás toda la página.

Recientemente, la usabilidad ha sido reconocida como una de las propiedades fundamentales para garantizar el éxito de las aplicaciones Web.

2.3. Usabilidad Web

En el apartado anterior, se ha desarrollado el concepto general de Usabilidad, sin embargo, cuando se aplica al contexto de la Web, es necesario realizar algunos refinamientos sobre las definiciones para capturar la especificidad de esta plataforma que ha evolucionado sustancialmente desde sus orígenes.

Partiendo de la definición de la ISO 9241 (ISO 1998), es posible definir la usabilidad Web como la habilidad de los sitios Web y de las aplicaciones Web para soportar las tareas o actividades que puede realizar un usuario, con efectividad, eficiencia, y satisfacción.

Los principios de usabilidad de Nielsen, se aplican directamente al contexto de las aplicaciones web, dado que al igual que las aplicaciones de escritorio, asisten a los usuarios en la realización de tareas específicas y variadas. La diferencia principal radica en las plataformas sobre las cuales de ejecutan ambos tipos de aplicaciones.

Por otro lado, en el contexto de los sitios Web se debe realizar una aclaración con respecto a estos principios:

- Facilidad de aprendizaje. En sitios web, debe interpretarse como la facilidad para los usuarios de entender los contenidos y servicios de la página inicial, y de buscar información específica utilizando los enlaces disponibles y navegando por los hipertextos.

- Eficiencia de los sitios Web. Significa que los usuario que quieren encontrar algún contenido, pueden obtenerlo rápidamente a través de los enlaces disponibles.

- Fácil de recordar. Implica que, después de un periodo de no usar la aplicación, los usuarios son aún capaces de orientarse dentro del hipertexto, por ejemplo por medio de las barras de navegación que apuntan a páginas visitadas previamente (historial).

- Pocos errores. Significa que en el caso que los usuarios hayan ingresado a un enlace erróneamente, deberían ser capaces de regresar a su ubicación previa.

- Satisfacción del usuario. Se refiere a la situación en la que los usuarios sienten que tienen el control con respecto al hipertexto, gracias a la comprensión de los contenidos disponibles y de los comandos de navegación.

2.4. Programación Orientada a Aspectos (AOP)

El Desarrollo de Software Orientado a Aspectos (AOSD) (Aspect-Oriented Software Development 2004) provee un conjunto de técnicas para identificar, modularizar, representar y componer crosscutting concerns (CCC). El término CCC refiere a aquellos factores o funcionalidades transversales del software de calidad que no pueden ser efectivamente modularizados usando técnicas de desarrollo de software tradicionales (como los enfoques OO) y generan código mezclado y enmarañado. Ejemplos típicos de funcionalidad transversal son logging, traza, seguridad y persistencia.

Por otro lado, la AOP (Kiczales G. et al. 1997) es una técnica de programación cuyo objetivo es encapsular la funcionalidad transversal que genera código mezclado y diseminado, generalmente relacionado a requerimientos no funcionales, técnicos y/o de calidad. La AOP propone fundamentalmente una nueva clase de modularización que va más allá de los procedimientos generalizados: los aspectos (módulos capaces de localizar la implementación de un CCC).

Con el fin de encapsular e implementar este tipo de funcionalidad la AOP introduce cuatro nuevas abstracciones: join-point, pointcut, advice y aspectos:

- Un join-point es un punto bien definido en la ejecución de un programa, por ejemplo la invocación a un método.

- Un pointcut es una expresión que especifica las condiciones de ejecución de un segmento de código (advice), es decir, determina el conjunto de join-points que serán interceptados por el aspecto, y expone algunos de los valores del contexto de los mismos.

- Un advice es un segmento de código similar a un método que se ejecuta en cada join-points especificado en los pointcuts.

- Un aspecto es un tipo transversal que encapsula pointcuts, advices y características transversales estáticas. Un aspecto es la unidad de modularización de la AOP.

Los aspectos se integran (componen) en el sistema por medio de un proceso especial llamado tejedor (Piveta E. y Zancanela L. 2003). Actualmente hay extensiones de lenguajes convencionales que soportan AOP para las aplicaciones implementadas con esos lenguajes de programación convencionales. AspectJ (Kiczales G. et al. 2001) es uno de los soportes AOP para Java.

En los primeros estadios del paradigma una extensa literatura se orientó a la aplicación del enfoque a problemas reales tales loging, profiling, persistencia, monitorización, coordinación, distribución, reglas de negocio, interfaz gráfica, performance, traza, etc. en aplicaciones de usuario aunque con requerimientos complejos. En un nivel ya más avanzado y complejo, el uso y aplicación de aspectos ha sido considerado para middleware o como parte de frameworks, lo que se presenta como una posibilidad de mayor reuso.

2.5. Framework

Un "framework" es un conjunto de clases cooperantes que constituyen un diseño reutilizable para una clase específica de software (Deutsch L. 1989) (Johnson R. y Foote B. 1988). Por ejemplo, un framework puede estar orientado a la construcción de editores gráficos para dominios diferentes. Para el desarrollo de una aplicación concreta se crean subclases específicas de la aplicación a partir de clases abstractas del framework.

Esta es la concepción clásica de un framework (Gamma E. et al. 1994) en la cual su estructura sigue una visión puramente OO. Esta mirada se resignifica y reconstruye a la luz de nuevas formas de modularización como los aspectos, que ya han sido incorporados a diversos frameworks para el desarrollo de aplicaciones software como es el caso de Spring AOP (Laddad R. 2010) y de JBoss AOP (Fleury M. y Reverbel F. 2003). Incluso dan mejor soporte a mecanismos inherentes a un framework (inversión de control). Existe además un andamiaje de trabajos que recomiendan a la AOP/AOSD para el diseño e implementación de frameworks y middleware (Lagaisse B. et al. 2006) (Loughran N. et al. 2005) (Greenwood P. et al. 2007) (Sanen F. et al. 2006) (Rausch A. et al. 2003). En (Loughran N. et al. 2005) se presenta un profundo y exhaustivo estudio que analiza, evalúa, e integra diversos trabajos en el dominio de los siguientes concerns claves: persistencia, seguridad, manejo del contexto y movilidad, cuyo diseño e implementación son mejorados mediante AOSD.

Si bien, la aplicación de aspectos en el desarrollo de diferentes frameworks ha sido estudiado por varios autores, se utilizan mayormente para el diseño de los frozen spots (las partes inmutables del framework). (Rausch A. et al. 2003) utilizaron aspectos para conectar el núcleo del framework y las aplicaciones generadas. En (Santos A. et al. 2007) y (Arpaia P. et al 2008), los aspectos fueron utilizados para poner en práctica los hot-spots, sin embargo, no se han aplicado los patrones de diseño OA para este propósito.

Diseño de estructuras más complejas que implican algunos modismos de AspectJ fueron sugeridos por Kulesza U. et al. (2006). Estos autores proponen una forma de utilizar la extensión de join-points para diseñar hots-spots. (Hanenberg S. et al. 2003), (Laddad R.

2003), (Miles R. 2004), (Bynens M. y Joosen W. 2009) han propuesto una serie de patrones de diseño OA que se pueden aplicar con éxito para diseñar Frameworks OA. Vaira Ž. y Čaplinskas A. (2011) propusieron una forma de desarrollar puramente patrones de diseño OA para diseñar aspectos abstractos en los frameworks de dominio OA.

3. DESCRIPCIÓN DEL PROBLEMA

Tal cual se ha mencionado, la evaluación de usabilidad de una aplicación, consiste en realizar pruebas para obtener medidas e información que permitan identificar debilidades relacionadas al uso de la misma. Independientemente del método de evaluación empleado, o combinación de los mismos, la captura de datos para su análisis y crítica posterior, es una tarea fundamental.

A diferencia de los sitios web, donde la evaluación de usabilidad se centra, entre otros aspectos, en la arquitectura de la información (estructura, navegación, facilidad de búsqueda, etc.), las aplicaciones web requieren el análisis de patrones de interacción mucho más complejos, similares a las aplicaciones de escritorio. En este sentido, se puede requerir información contextual que solo puede ser obtenida del análisis dinámico de la aplicación.

Sin embargo, pese al grado de interacción que ofrecen una u otra (sitio vs aplicación web), no presentan cambios muy importantes en lo que se refiere a su arquitectura. Tanto las aplicaciones como los sitios web, se caracterizan por una arquitectura cliente - servidor. Esta arquitectura, define un modelo de interacción basado en un diálogo petición - respuesta: el cliente inicia el diálogo mediante el envío de peticiones, y el servidor presta el servicio respondiendo las peticiones recibidas. En este tipo de arquitecturas, la interacción con el usuario ocurre en el cliente a través de un navegador web o browser. Es muy importante notar que en el cliente, acontecen eventos que pueden resultar de interés para el proceso de evaluación de usabilidad. Ejemplos de este tipo eventos pueden ser: errores en el ingreso de datos, mensajes de alerta o de información, o incluso interacciones con algunos componentes de la interfaz. Estos eventos son gestionados directamente en el browser por el lenguaje javascript, y por lo general, no se notifica al servidor de la ocurrencia de los mismos, más aún, no se registra información contextual a menos que así se explicite en la lógica de programación del lenguaje javascript. En general, las técnicas de registro de datos, sólo capturan los datos de aquellos sucesos que son gestionados en el servidor, tales como la validación de credenciales, el envío de datos para ser almacenados en una base datos, consultas de datos, excepciones generadas en el servidor, entre otros.

Otro problema asociado a la captura de datos automatizada, radica en el volumen y la relevancia de los datos que servirán de entrada para la evaluación de usabilidad. Las técnicas de registro de datos se han caracterizado por usar archivos de logs. Las ventajas que ofrece este tipo de registro son: a) soporta la ejecución de las pruebas en laboratorios de usabilidad o en el ambiente natural del usuario; b) es transparente para los usuarios; c) permite con facilidad obtener información de múltiples usuarios; y d) permite detectar errores comunes y actuales.

Si bien, los registros de logs generados por servidores web aportan una gran cantidad de datos a los evaluadores de usabilidad, permitiéndoles contrastar comportamientos entre diferentes usuarios, también poseen una gran cantidad de información que les resulta irrelevante. Debido a la gran cantidad de datos que generan los logs y que estos no están pensados para posteriormente ser usados en la evaluación de la usabilidad, es que la tarea de análisis e

interpretación de los datos crece en complejidad resultando ser realmente problemática y propensa a errores.

Por lo tanto, los desarrolladores necesitan diseñar e integrar mecanismos de registros adecuados que respondan a las necesidades de información de los evaluadores de usabilidad. Una alternativa para dar solución a este problema, es añadir en las aplicaciones el código que realiza el logging de las actividades / tareas de usabilidad que realmente interesan al evaluador. Esta aproximación resulta ser invasiva, ya que altera la estructura interna de las aplicaciones al agregar cientos de líneas de código cuya remoción posterior es muy compleja. Bajo este contexto, se identifica la necesidad de aplicar enfoques flexibles, configurables, reutilizables y no intrusivos, que permitan la captura de datos automática para la evaluación de la usabilidad.

A partir de la aparición de las técnicas de separación de concerns (Hürsch W. y Lopes C. 1995), y en particular de la AOP, es posible diseñar módulos de código (aspectos) que realicen el seguimiento y log sin invadir y/o alterar la estructura interna de los sistemas. La posibilidad de realizar la recolección de los datos en forma dinámica, transparente para el usuario y sin invadir el código, ha provocado interés y varios enfoques se han presentado que aplican AOP a la evaluación de usabilidad en aplicaciones de escritorio y móviles (Tarta A. y Moldovan G. 2006), (Tarby J. et al. 2007), (Yonglei T. 2008), (Holzinger A. et al. 2011), (Kronbauer A. y Santos C. 2011), (Lettner F. y Holzmann C. 2012), (Yonglei T. 2012), (Kronbauer A. et al. 2012). Llamativamente no se han encontrado en la literatura enfoques que usen AOP para la evaluación de la usabilidad de aplicaciones WEB.

En este trabajo, se propone un framework de dominio para dar soporte a la recopilación automática de datos, en el proceso de evaluación usabilidad, para aplicaciones web.

4. FRAMEWORK AJMU-WEB

AJMU-WEB, es un framework de dominio que permite recolectar datos de aplicaciones web para la evaluación de usabilidad. El mismo, ha sido desarrollado de acuerdo al paradigma de AOP y brinda soporte a los métodos de evaluación de usabilidad de *Testeo con Usuarios* basados en el análisis de archivos de log (Ivory M. y Hearst M. 2001) (Fernandez A. 2009). En este caso, se recolectan datos en torno a las dimensiones propuestas por la ISO 9241: Eficiencia, Efectividad y Satisfacción. Para cada dimensión se plantean un conjunto de métricas que permiten identificar qué datos de la interacción serán recolectados. Los factores y métricas que soporta el framework se presentan en la Tabla 2.

FACTORES		***METRICAS / INDICADORES***	***Eventos Capturados por AJMU WEB***
Eficiencia		Tiempo en realizar la tarea	Inicio y Finalización de una tarea
Efectividad	*Errores*	Cantidad de Excepciones producidas durante la ejecución de la tarea	Excepciones ocurridas durante la ejecución de una tarea
		Cantidad de Accesos a la ayuda producidos durante la ejecución de la tarea	Accesos a la documentación de ayuda durante la ejecución de una tarea
	Completitud	Tareas finalizadas Cantidad de Tareas Finalizadas	Inicio y Finalización de una tarea Cierre de sesión
		Tareas no finalizadas Cantidad de Tareas no Finalizadas	Sucesos que indican que el usuario abandonó una tarea (ejemplo, cierre de la sesión)
Satisfacción		Satisfacción por la complejidad de la tarea realizada	Finalización de una tarea
		Satisfacción por el tiempo requerido en realizar la tarea	
		Satisfacción por el uso de la aplicación para realizar la tarea	
Otras		Perfil de usuario (edad y sexo)	Inicio de la sesión

Tabla 2 – Evaluación de Usabilidad de una Tarea: Factores y Métricas

En el diagrama de clases de la Figura 1, puede apreciarse la estructura del framework AJMU-WEB.
A continuación se describe para cada uno de sus componentes, su finalidad y su implementación.

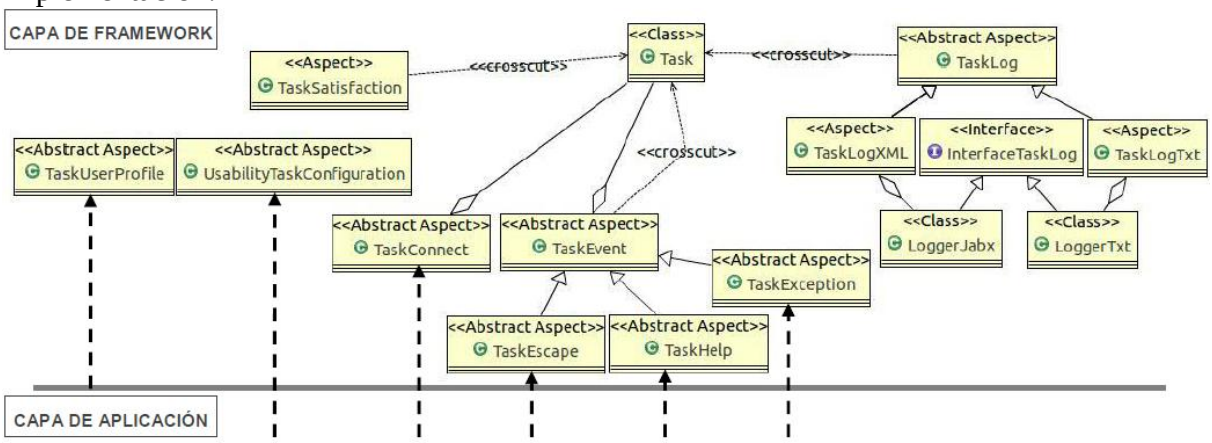


Figura 1 - Diagrama de Módulos de AJMU-WEB

4.1. Componentes del Framework

4.1.1. Tareas de usuario

El framework AJMU-WEB, se compone de un conjunto de módulos, aspectos, y clases. La clase Task representa una tarea de usuario y se compone de atributos y métodos. Entre los atributos, se encuentran las métricas que se calculan durante la ejecución de la tarea, e información sobre el estado de la misma.

Las métricas, que se calculan durante la ejecución de la tarea, son: la cantidad de excepciones ocurridas, la cantidad de accesos a la documentación o ayuda, y el grado de satisfacción del usuario.

```
public class Task {
       private String id, state, sexUser;
       private boolean complete;
       privatelong init,end;
       private int totalExceptions, totalAccessDocumentation, sat1, sat2, sat3,
       ageUser;
       public Task(String idTask) {
            id = idTask;
            complete = false;
            init = System.currentTimeMillis();
            end = 0;
            totalExceptions = 0;
            totalAccessDocumentation = 0;
            state = "initiated";
       }
       public void finalize(){
            end = System.currentTimeMillis();
            complete = true;
            state = "Finished";
       }
       public void noFinalize(){
            complete = false;
            state = "No Finished";
       }
       //métodos getter y setter
}
```

En AJMU-WEB, una tarea puede transitar, durante su ejecución, por alguno de los siguientes estados: Initiated(), Running(), Finished(), ó No_Finished(). El estado en el cual se encuentra una Tarea en un determinado momento, es indicado por el atributo estado (status) de la clase Task. Otros atributos de interés son: init y end, los cuales registran los momentos en los cuales inicia y finaliza una tarea respectivamente. Ambos atributos, son empleados posteriormente para calcular el tiempo que empleó el usuario en realizar la tarea.

Los métodos de esta clase, además de permitir el acceso al valor de sus atributos, realizan los cálculos de la métricas, y actualizan la información de estado de la tarea.

El aspecto que permite conectar la Tarea con la Aplicación, es TaskConnect. Se trata de un aspecto abstracto que cuenta con dos pointcuts abstractos: startTask y endTask. En el primero de ellos se debe definir los joinpoints que indican el inicio de la tarea (cuando comienza la tarea), y en el segundo, los joinpoints que indican el final de la tarea (cuando termina la tarea). La definición de estos joinpoints, para cada uno de estos pointcuts abstractos, se debe realizar en un aspecto concreto, para cada una de las tareas de usuario que se desee evaluar. Es decir que una vez definidas las tareas de usuario que se evaluarán, el desarrollador debe crear un aspecto concreto para cada una de las tareas, y en él definir los jointpoints para los pointcuts startTask y endTask (Figura 2).

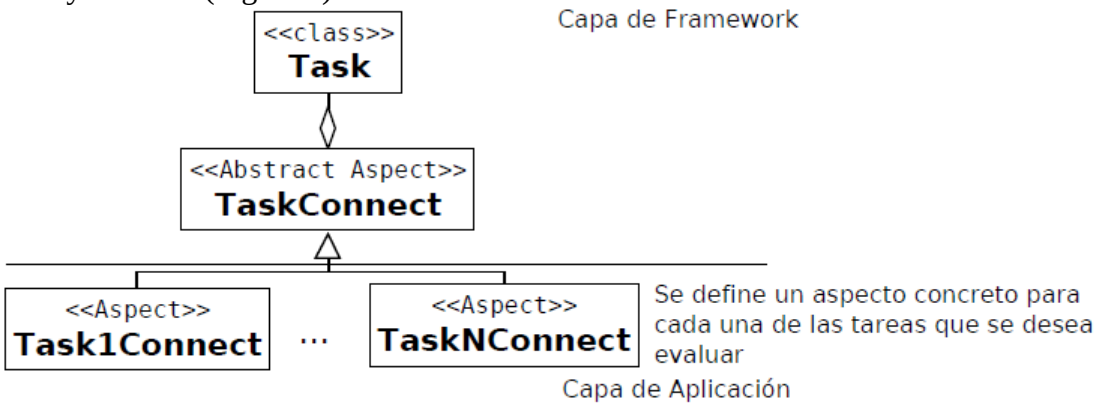


Figura 2 - Extensión de TaskConnect para cada una de las Tareas que se desea evaluar.

La instanciación de la clase Task, es posible a través de TaskConnect ya que este aspecto posee como atributo un objeto Task que será instanciado cuando, tras la ocurrencia del inicio de cada una de las tareas, el pointcut startTask sea activado y el advice asociado ejecutado. De manera similar, este aspecto lleva a cabo la terminación de la tarea cuando el método finalize es invocado por el advice asociado al pointcut endTask, luego de la ejecución de sus correspondientes jointpoints. Cada vez que una tarea es iniciada, y la clase Task instanciada, el id del objeto Task y el identificador de la Tarea iniciada, se insertan en una estructura de datos Hashtable denominada startedTasks. Luego, cada vez que una tarea es finalizada, el registro del objeto Task asociado se extrae de startedTasks. Al finalizar la etapa de ejecución de tareas por parte del usuario, startedTasks tendrá sólo aquellas tareas que no fueron finalizadas, es decir que, debido a determinadas circunstancias, el usuario las abandonó luego de haberlas iniciado.

```
abstract aspect TaskConnect {
   Task taskAnalyzed;
   Hashtable<Integer, String> startedTasks;
   abstract void setIdTask();  <- hot spot
   abstract pointcut starTask; <- hot spot
   abstract pointcut endTask;  <- hot spot
   before() : starTask(){
```

```
            this.setIdTask();
            taskAnalyzed = new Task(idTask);
            startedTasks.put(taskAnalyzed.hashCode(), taskAnalyzed.getId());
        }
        after() returning: endTask(){
            taskAnalyzed.finalize();
            startedTasks.remove(taskAnalyzed.hashCode());
            taskAnalyzed = null;
        }
    }
```

TaskConnect, también posee un método abstracto denominado setIdTask() que es invocado durante la instanciación del objeto Task. Este método, el cual debe ser definido por el desarrollador, permite identificar la tarea que se está ejecutando mediante un id numérico que debe ser asignado por el evaluador. Este identificador, permitirá reconocer los registros de logging correspondientes a las ejecuciones de una determinada tarea para luego realizar diferentes análisis que permitan determinar la usabilidad de la aplicación para la realización de dicha tarea.

4.1.2. Registro de datos

El registro de los datos de Logging, son realizados por el aspecto abstracto TaskLog. Este aspecto registra datos asociados a diferentes eventos que ocurren durante la sesión de un usuario, principalmente, durante la ejecución de las tareas evaluadas.

Si bien, este aspecto mantiene una relación de dependencia con la clase Task, ofrece su funcionalidad como un servicio que otros módulos (aspectos) pueden consumir para registrar datos contextuales sobre diferentes eventos.

El aspecto abstracto TaskLog, establece un conjunto de pointcuts concretos y métodos abstractos. Los pointcuts, están sujetos a la ejecución de la tarea e interceptan los siguientes sucesos: la creación de una tarea, la terminación de una tarea, el cambio de estado de una tarea, la actualización de las métricas de una tarea.

Los métodos abstractos, permiten definir mecanismos para registrar los datos en diferentes formatos, tales como XML o texto plano. Estos métodos son implementados por aspectos concretos tales como TaskLogXML (registra los datos en formato XML), y TaskLogTXT (registra los datos en formato de texto plano). Figura 3.

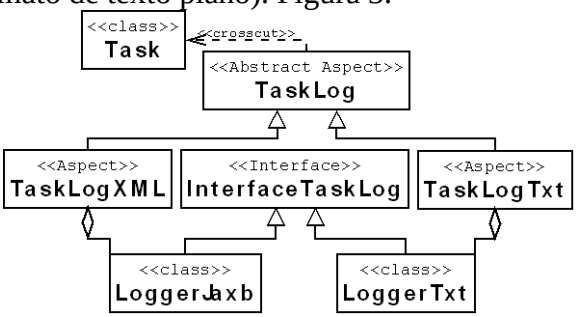


Figura 3 - Módulo encargado del registro de datos

A continuación, se describe la funcionalidad de estos métodos:

- initTask y endTask registran datos referidos al inicio y la finalización de la ejecución de la tarea.

- events, registra datos referidos a la situación actual de la tarea, cuando ocurre un cambio en su estado, ó una actualización de alguna de sus métricas.

- log, este método, registra datos contextuales acerca de un evento determinado.

```
abstract aspect TaskLog {
   abstract void events();
   abstract void initTask();
   abstract void endTask();
   abstract InterfaceTaskLog log(Task t);

   pointcut logStart(Task t): execution (Task.new(..))..
   pointcut logEnd(Task t): execution (Task.finalize(..))..
   pointcut logEvent(Task t) : call (*.Task.setQ*(..))…
   pointcut logNoEnd(Task t) : execution(void Task.noFinalize(..))&&this(t);

   after(Task t) : logStart(t){
         initTask(t);
         t.setState("Running");
   }
   after(Task t) : logEnd(t){
         endTask(t);
   }
   after(Task t) : logEvent(t){
         events(t);
   }
   after(Task t): logNoEnd(t){
         log().addLogPartial(..);
   }
}
```

4.1.3. Captura de Eventos

Una tarea de usuario, durante su ejecución, puede transitar por diferentes estados y sus métricas, ser actualizadas ante la ocurrencia de los siguientes eventos: excepciones, accesos a la documentación de ayuda, o eventos que indican que el usuario abandonó la tarea sin antes haberla finalizado.
Con la finalidad de capturar estos eventos, una jerarquía de aspectos fue diseñada (Figura 4).
Los dos primeros niveles de la jerarquía, son abstractos.
El aspecto abstracto de mayor nivel en la jerarquía es TaskEvent. En este aspecto, se definen un conjunto de pointcuts concretos que delimitan el flujo de control dentro del cual se espera capturar los diferentes tipos de sucesos antes mencionado.
Por ejemplo, en el pointcut aspectFlow, se define el flujo de control que ocurre dentro del aspecto con la finalidad de exceptuarlo en la definición de los pointcuts de los restantes aspectos y de esta manera evitar capturar eventos que ocurren dentro de esta jerarquía y que no corresponden al dominio dentro del cual se está ejecutando la tarea evaluada.

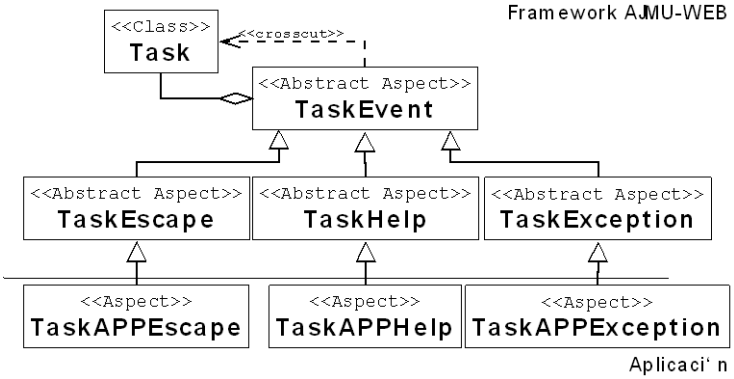


Figura 4 - Módulo de Eventos

Los pointcuts initFlow y endFlow, hacen referencia al flujo de control que ocurre durante el inicio y la finalización de cada una de las tareas que serán evaluadas.

Finalmente, isATask, permite verificar que existe una tarea en ejecución.

Además de estos pointcuts, el aspecto TaskEvent, ofrece un método abstracto denominado logEvent, cuya finalidad, es permitir al desarrollador, seleccionar entre los servicios de registro de datos disponibles en el framework, inicialmente Texto plano, ó, XML.

```
abstract aspect TaskEvent {
	private Task taskRef=null;
	abstract Interface logEvent();<- hot spot

	pointcut init(): initialization(Task.new(..));
	pointcut end(): execution (void Task.finalize(..));
	pointcut aspectFlow():cflow(adviceexecution);

	pointcut initFlow():cflow(init());
	pointcut endFlow(): cflow(end());
	pointcut isATask():if ((taskRef!=null)&&(!taskRef.isComplete()))
	pointcut saveRef(Task t): init()&&this(t);

	after(Task t) : saveRef (t){
		taskRef=t;
	}
}
```

En el segundo nivel de la jerarquía se encuentran los aspectos abstractos TaskException, TaskHelp, y TaskEscape.

En TaskException, se definen dos pointcuts concretos que capturan las excepciones que ocurren durante el inicio de la tarea, y durante la ejecución de la misma.

Asociados a estos pointcuts, existen advices que actualizan el contador de excepciones ocurridas durante la ejecución de la tarea (métrica totalExceptions), y registran los datos contextuales en el formato que luego deberá ser definido en el método abstracto logEvent.

Por otro lado, TaskHelp, ofrece un pointcut abstracto (accessDocumentation) que el desarrollador puede extender para definir un conjunto de joinpoints que permitan capturar los accesos del usuario a la documentación de ayuda del sistema.

Al igual que ocurre con TaskException, vinculado al pointcut accessDocumentation, existe un advice que además de actualizar el contador de accesos a la ayuda del sistema (accessDocumentation), registra datos contextuales en el formato definido en logEvent.

```
abstract aspect TaskException extends TaskEvent{
	pointcut initExceptions(Throwable e):
		initFlow()&&!aspectFlow()&&handler(Throwable+)&&args(e)
	pointcut executionExceptions(Throwable e):
		!initFlow()&& !aspectFlow()&&handler(Throwable+)&&args(e)&&isATask();
	before(Throwable e) : initExceptions (e){
		taskRef.setTotalExceptions();
		logEvent().addException(taskRef.getTotalExceptions(), reg);
	}
	before(Throwable e) : executionExceptions (e){
		taskRef.setTotalExceptions();
		logEvent().addException(taskRef.getTotalExceptions(), reg);
	}
}
```

Con respecto al aspecto abstracto TaskEscape, el mismo, analiza si la tarea fue abandonada voluntariamente por el usuario antes de su normal finalización. Posee un pointcut abstracto denominado notComplete, en el cual se deben definir los joinpoints necesarios que permitan capturar aquellas condiciones particulares que permitan reconocer cuándo un usuario abandona la tarea. Un ejemplo de estas condiciones particulares, puede ser, pulsar el botón Cancelar para interrumpir una operación. Asociado a este pointcut, existe un advice que invoca al método noFinalize de la clase Task.

```
abstract aspect TaskEscape extends TaskEvent{
	abstract pointcut notComplete();<- hot spot
	pointcut completeEscape():(call(void java.lang.System.exit(..))&&
	!endFlow()&&!aspectFlow()&& isATask()||(notComplete()&& isAtask());
	before(): completeEscape(){
		taskRef.noFinalize();
		taskRef = null;
	}
}
```

Los aspectos concretos citados previamente, y que heredan de TaskEvent, hacen uso de los pointcuts concretos de este último en la definición de sus respectivos pointcuts con la finalidad de capturar la ocurrencia de los diferentes eventos durante la ejecución de las tareas de usuario a ser evaluadas.

4.1.4. Adquisición de datos

Previamente, se ha mencionado que la definición de la clase Task, también comprende atributos vinculados al grado de satisfacción del usuario al realizar la tarea. Estos atributos son: el grado de dificultad que experimentó el usuario para realizar la tarea (difficultTask); el grado de satisfacción del usuario, luego de realizar la tarea (satisfiedTask); y el tiempo que empleó el usuario en realizar la tarea (timeTask).

La valoración de estos atributos, es asignada por el usuario de manera subjetiva, a través de un cuestionario lanzado, por el aspecto TaskSatisfaction, cada vez que una tarea finaliza. Esto es posible, ya que TaskSatisfaction cuenta con un pointcut denominado satisfaction, que captura las invocaciones al método finalize de la clase Task.

```
aspect TaskSatisfaction {
	pointcut satisfaction(Task t):execution(void Task.finalize(..))&& this(t);
	before(Task t): satisfaction (t) {
		// desplegar cuestionario, obtener atributos satisfacción de la tarea
		return;
	}
}
```

De acuerdo a la metodología de evaluación de usabilidad adoptada, también es necesario identificar el perfil de los usuarios que participan de la evaluación y realizan las tareas definidas para el experimento. El aspecto abstracto TaskUserProfile, ofrece un pointcut abstracto donde el desarrollador deberá definir (en un aspecto concreto) los jointpoints necesarios para capturar cada inicio de sesión de un usuario en la aplicación evaluada. Este pointcut, posee un advice asociado que, luego de iniciada la sesión, muestra al usuario un formulario a través del cual se le solicitan algunos datos tales como su edad, y el sexo. Una vez obtenidos estos datos, son registrados de acuerdo a alguno de los formatos disponibles en el framework (XML o Texto Plano) que el desarrollador deberá definir en el método abstracto logProfile.

```
abstract aspect TaskUserProfile implements ActionListener{

	abstract InterfaceTaskLog logProfile(); <- hot spot
```

```
	abstract pointcut initUserSession();	<- hot spot

	before():initUserSession() {
		// desplegar cuestionario y obtener datos del perfil
		logProfile();
	}
}
```

Finalmente, el aspecto abstracto UsabilityTaskConfiguration, a través de la definición de sus métodos abstractos, permiten al desarrollador, configurar las tareas que serán evaluadas durante una sesión particular, y así mismo, registrar datos de la aplicación tales como su nombre y versión. Este aspecto dispone de un sólo pointcut abstracto (configuration), en el cual se deben definir los joinpoints necesarios para capturar el inicio de la aplicación.

```
abstract aspect UsabilityTaskConfiguration {
	String appName, appVersion;
	HashTable<String, String> tasksTest;

	public void setApplicationTest(String name,String version){
		appName = name;
		appVersion = version;
	}
	public void addTask(String id, String desc){
		tasksTest.put(id, desc);
	}

	abstract void initializationLogger(); <- hot spot
	abstract InterfaceTaskLog logApp();	<- hot spot
	abstract pointcut configuration();	<- hot spot

	before(): configuration(){
		this.initializationLogger();
//Se registran los datos de la aplicación y las tareas a ser evaluadas con logApp
	}
}
```

5. PATRONES DE DISEÑO

Con la finalidad de lograr la funcionalidad deseada, se han implementado diferentes patrones de diseño que otorgan al framework la flexibilidad necesaria para adaptarse a diversas aplicaciones web permitiendo a los programadores configurarlo sin modificar su núcleo (frozen-spot). Por el contrario, solo necesitará extender determinados módulos (aspectos) que le permitirán conectar el framework con la aplicación que se desea evaluar (hot-spot).

El arquitecto Christopher Alexander, define a un patrón como una solución recurrente a un problema común en un contexto dado y sistema de fuerzas, dicho de otra manera, cada patrón describe un problema que se da continuamente en nuestro entorno así como también la solución base o nuclear a ese problema, de tal manera, que es posible usar la solución un millón de veces sin aplicarla del mismo modo" (Alexander C. et al. 1977). En la definición de este autor, el término contexto, hace referencia a un conjunto de condiciones ó situaciones en las cuáles un determinado patrón es aplicable, y el término sistema de fuerzas se refiere al conjunto de restricciones que ocurren en el contexto específico.

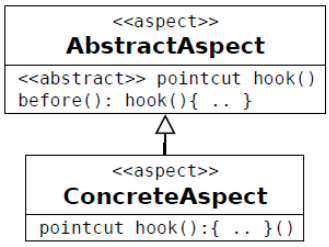
La aplicación de patrones de diseño permite ayudar a mejorar la calidad del software en términos de reusabilidad, mantenibilidad, y extensibilidad; reducir el tiempo de desarrollo; y aumentar la fiabilidad del software (Kuchana P. 2004).

Actualmente, se han descubierto y definido un número significativo de patrones orientados a aspectos (Hanenberg S. y Schmidmeier A. 2003),(Hanenberg S. et al. 2003), (Noble J. et al. 2007), (Menkyna R. et al. 2010). Estos patrones, permiten a los programadores, comprender y evaluar los diseños orientados a aspectos existentes, con la finalidad de mejorarlos, y hacer un mejor uso de los lenguajes de programación orientados a aspectos y servir de guía para quienes deseen implementar estos patrones en lenguajes no orientados a aspectos.

En esta Sección, se describen los patrones de diseño orientados a aspectos que se emplearon para satisfacer diferentes requerimientos de la implementación del framework AJMU-WEB, y se describe de qué manera fueron implementados en el framework.

5.1. Abstract Pointcut

Problema que intenta resolver: Si bien, el comportamiento de un aspecto debería poder ser reusado en diferentes aplicaciones, a veces se desconoce el momento durante el cual este comportamiento debería tener lugar (es decir dónde se hará el crosscutting, o lo que es lo mismo, donde ocurrirán los advices). Dicho de otra manera algunas partes del pointcut no se conocen y dependen de la aplicación concreta donde se utilizará el aspecto.
Solución: El aspecto se define como abstracto al igual que los pointcuts variables. Luego en un aspecto concreto, que lo extiende, se definen los pointcuts según sea necesario (Hanenberg S. et al. 2003). En el aspecto concreto, se pueden definir o sobreescribir los pointcuts abstractos (Figura 5).
La definición del pointcut completo, se deriva para que sea el programador quien realice la conexión del aspecto con la aplicación. Es decir que el programador, necesita conocer con exactitud el propósito para el cual el aspecto fue creado para poder definir correctamente los join points donde el comportamiento del aspecto tomará sentido.




```
public aspect AbstractAspect{
      abstract pointcut hook();
      before(): hook(){
            System.out.println("Before method was called");
      }
}
public aspect ConcretAspect extends AbstractAspect {
      pointcut hook(): call (void TargetClass.method());
}
```

FIGURA 5 - PATRÓN ABSTRACT POINTCUT

Aplicación en AJMU-WEB
Este patrón de diseño se ha aplicado a los siguientes módulos del framework AJMU-WEB:

- UsabilityTaskConfiguration. Este aspecto dispone de un pointcut abstracto (denominado configuration), en el cual se deben definir los joinpoints necesarios para capturar el inicio de la aplicación. Se espera que cada aplicación web disponga de un método de inicio diferente, por tal motivo, la definición de este pointcut deberá ser realizada por el programador.

```
abstract aspect UsabilityTaskConfiguration {
      // datos de la app
      String appName, appVersion;
      // listado de tareas a ser evaluadas
      HashTable<String, String> tasksTest;
      // Operaciones
      public void setApplicationTest(String name,String version){
            appName = name;
            appVersion = version;
      }
      public void addTask(String id, String desc){
            tasksTest.put(id, desc);
      }
      // Operaciones primitivas

      abstract void initializationLogger(); <- hot spot

      abstract InterfaceTaskLog logApp();   <- hot spot

      // Pointcuts Abstractos
      abstract pointcut configuration();    <- hot spot
```

El propósito de este pointcut, es permitir al programador definir el conjunto de joinpoints necesarios para capturar el inicio de la aplicación.

```
    before(): configuration(){
            this.initializationLogger();
            ...
                  //Se registran los datos de la aplicación y las tareas evaluadas
            con logApp
            ...
    }
}
```

- TaskUserProfile. Ofrece un pointcut abstracto donde se deben definir los jointpoints necesarios para capturar cada inicio de sesión de un usuario en la aplicación web evaluada. Este pointcut, posee un advice asociado que a través de un formulario solicita al usuario datos tales como el sexo y la edad. Dado que el evento que inicia una sesión de usuario puede variar para cada aplicación, la definición del pointcut debe ser realizada por el programador que utilice el framework.

```
abstract aspect TaskUserProfile implements ActionListener{

    // Operaciones primitivas
    abstract InterfaceTaskLog logProfile(); <- hot spot

    // Pointcuts abstractos
    abstract pointcut initUserSession(); <- hot spot

    before():initUserSession() {
            // desplegar cuestionario
            // obtener datos del perfil
            logProfile();
    }
}
```

En un aspecto concreto, el programador debe definir el conjunto de joinpoints necesarios para capturar el inicio sesión de los usuarios.

- TaskConnect. Este aspecto es el que permite conectar las tareas de la aplicación web, con el framework AJMU-WEB. Cada Tarea definida, deberá poseer un inicio y un fin que la identifiquen de manera única frente al resto de las tareas. El inicio y fin, de cada tarea, se debe definir en los pointcuts abstractos startTask y endTask, respectivamente. Ambos pointcuts, sólo podrán ser definidos una vez determinado el listado de tareas que se emplearán durante la etapa de evaluación de usabilidad de la aplicación web.

```
abstract aspect TaskConnect {
    Task taskAnalyzed;
    Hashtable<Integer, String> startedTasks;

    //Operaciones primitivas
    abstract void setIdTask();  <- hot spot

    //Pointcuts abstractos
    abstract pointcut starTask; <- hot spot
    abstract pointcut endTask;  <- hot spot

    before() : starTask(){
            this.setIdTask();
            taskAnalyzed = new Task(idTask);
            startedTasks.put(taskAnalyzed.hashCode(), taskAnalyzed.getId());
    }
    after() returning: endTask(){
            taskAnalyzed.finalize();
            startedTasks.remove(taskAnalyzed.hashCode());
            taskAnalyzed = null;
    }
}
```

El programador debe crear un aspecto concreto para cada tarea que será evaluada y definir cuándo inicia la tarea, y cuándo se

5.2. Template Advice

Problema que intenta resolver: Este patrón, se emplea cuando el código a ser ejecutado en un advice tiene una parte fija y una parte variable que depende de la aplicación sobre la cual se está realizando el crosscutting.
Solución: Usar la parte fija del advice como una plantilla y colocar la parte variable dentro de un método (operación primitiva) (Hanenberg S. et al. 2003). Ambos elementos deben definirse en un Aspecto Abstracto (Figura 6).
Luego, la operación primitiva podrá ser definida o sobreescrita, según se requiera, en un Aspecto Concreto.
Componentes de la solución (Hanenberg S. y Schmidmeier A. 2003):

1. El aspecto abstracto: contiene declaraciones de un número de operaciones primitivas (métodos abstractos).
2. El aspecto concreto: en este aspecto se definen o sobrescriben las operaciones primitivas del aspecto abstracto.

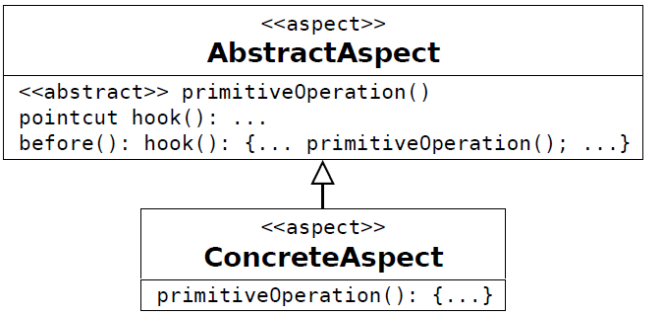


Figura 6 - Patrón de diseño Template Advice

```
public aspect AbstractAspect{
       abstract primitiveOperation();
       pointcut hook(): call(void TargetClass.method());
       before(): hook(){ primitiveOperation(); }
}
public aspect ConcretAspect extends AbstractAspect {
       void primitiveOperacion(){
              System.out.println("Before method was called in the application … ");
       }
}
```

Aplicación en AJMU-WEB

Este patrón de diseño se ha aplicado a los siguiente módulos del framework AJMU-WEB:

- TaskUserProfile, UsabilityTaskConfiguration, y TaskEvent. Cada uno de estos aspectos abstractos ofrece un método (logProfile, logApp, logEvent, respectivamente) cuya finalidad es permitir al desarrollador seleccionar entre los servicios de registro de datos disponibles en el framework, inicialmente Texto plano, y XML. Luego, este método es invocado desde diferentes advices para registrar información contextual durante la ejecución de las tareas.

```
abstract aspect TaskUserProfile implements ActionListener{
      // Operaciones primitivas
      abstract InterfaceTaskLog logProfile(); <- hot spot
      // Pointcuts abstractos
      abstract pointcut initUserSession();    <- hot spot
```

A través de la definición de este método, el programador determinará el formato que empleará para el registro de datos contextuales (txt ó xml).

```
        before():initUserSession() {
                // desplegar cuestionario Y obtener datos del perfil
                logProfile();
        }
}

abstract aspect TaskEvent {
        private Task taskRef=null;

        // Operaciones primitivas
        abstract Interface logEvent();<- hot spot

        //pointcuts componentes
        pointcut init(): initialization(Task.new(..));
        pointcut end(): execution (void Task.finalize(..));
        pointcut aspectFlow():cflow(adviceexecution);
        //composiciones de pointcuts
        pointcut initFlow():cflow(init());
        pointcut endFlow(): cflow(end());
        pointcut isATask():if ((taskRef!=null)&&(!taskRef.isComplete()))
        pointcut saveRef(Task t): init()&&this(t);
        after(Task t) : saveRef (t){
                taskRef=t;
        }
}
```

A través de la definición de este método, el programador determinará el formato que empleará para el registro de datos contextuales (txt ó xml)

- UsabilityTaskConfiguration. Ofrece un método abstracto (initializationLogger), el cual debe ser definido por el programador para especificar los datos de la aplicación web que será evaluada (nombre y versión), y el listado de tareas que se utilizarán durante la evaluación de usabilidad. Este método, es invocado dentro del advice asociado al pointcut configuration luego de iniciada la aplicación.

```
abstract aspect UsabilityTaskConfiguration {
        // datos de la app
        String appName, appVersion;
        // listado de tareas a ser evaluadas
        HashTable<String, String> tasksTest;
        // Operaciones
        public void setApplicationTest(String name,String version){
                appName = name; appVersion = version;
        }
        public void addTask(String id, String desc){
                tasksTest.put(id, desc);
        }
        // Operaciones primitivas
        abstract void initializationLogger(); <- hot spot

        abstract InterfaceTaskLog logApp();    <- hot spot

        // Pointcuts Abstractos
        abstract pointcut configuration();     <- hot spot
        before(): configuration(){
            this.initializationLogger();
            //Se registran los datos de la aplicación y las tareas evaluadas con
logApp
        }
}
```

En la definición de este método, el programador debe registrar datos de la aplicación evaluada, y el listado de tareas que se encomendarán a los participantes durante la evaluación de usabilidad.

A través de la definición de este método, el programador determinará el formato que empleará para el registro de datos contextuales (txt ó xml)

- TaskConnect. Durante la descripción del framework AJMU-WEB, se ha mencionado que el programador debe crear un aspecto concreto que extienda de TaskConnect por cada tarea utilizada en la evaluación. A los efectos, de identificar a qué tarea corresponden los registros de cada una de las ejecuciones, este aspecto ofrece un método abstracto (setIdTask) a través del cual el programador puede hacer referencia al Id de la tarea a la cual corresponde el aspecto concreto definido.

```
abstract aspect TaskConnect {
```

```
    Task taskAnalyzed;
    Hashtable<Integer, String> startedTasks;

    //Operaciones primitivas
    abstract void setIdTask();  <- hot spot

    //Pointcuts abstractos
    abstract pointcut starTask; <- hot spot
    abstract pointcut endTask;  <- hot spot
    before() : starTask(){
          this.setIdTask();
          taskAnalyzed = new Task(idTask);
          startedTasks.put(taskAnalyzed.hashCode(), taskAnalyzed.getId());
    }
    after() returning: endTask(){
          taskAnalyzed.finalize();
          startedTasks.remove(taskAnalyzed.hashCode());
          taskAnalyzed = null;
    }
}
```

Este método permite al programador referenciar el id de la tarea a la cual corresponde la definición del aspecto concreto.

5.3. Composite Pointcut

Problema que intenta resolver: Generalmente, los aspectos genéricos poseen pointcuts cuyas definiciones tienden a ser extensas debido al gran número join points donde el comportamiento del aspecto debe acontecer.
Una definición de pointcut compleja, reduce la comprensión del aspecto y su capacidad para ser reusado.
Otro inconveniente asociado a este problema, es que si una definición extensa de un pointcut necesita ser adaptada dentro de un aspecto concreto, la única solución posible es redefinir el pointcut completo.
Solución: Descomponer el pointcut en una composición de pointcuts (Hanenberg S. et al. 2003). Cada elemento de la composición representa un pointcut lógicamente independiente (Figura 7).
Algunas partes de una composición pointcut, pueden ser frecuentemente pointcuts de otros aspectos, sin embargo, los advices sólo harán referencia al pointcut compuesto (composición) y no a sus pointcuts componentes.
Frecuentemente este patrón ocurre en conjunción con ABSTRACT POINTCUT, donde uno de los pointcuts componentes es abstracto.

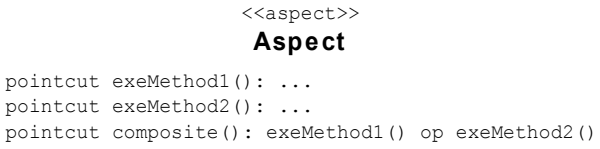


Figura 7 - Patrón de diseño Composite Pointcut

```
public aspect Aspect {
      pointcut exeMethod1(): execution(* TargetClass.method1());
      pointcut exeMethod2(): execution(* TargetClass.method2());
      pointcut composite(): exeMethod1() || exeMethod2();

      before(): composite(){ … }
}
```

Aplicación en AJMU-WEB

Este patrón de diseño se ha aplicado a la jerarquía de aspectos conformada por TaskEvent, TaskHelp, TaskException, y TaskEscape. En TaskEvent se definen un conjunto de pointcuts concretos que delimitan el flujo de control dentro del cual se espera capturar diferentes tipos

de sucesos. Estos pointcuts, son reusados en composiciones definidas en aspectos de nivel inferior dentro de la jerarquía de aspectos. Cabe aclarar, que TaskHelp y TaskEscape, ofrecen un aspecto abstracto (accessDocumentation y notComplete respectivamente) que es utilizado en la composición de pointcuts más complejos. A través de la definición de estos pointcuts abstractos (en un aspecto concreto), el programador puede capturar los accesos del usuario a la ayuda de la aplicación y las situaciones en las cuales abandona la tarea que se encuentra realizando.

Conjunto de pointcuts que luego serán utilizados para conformar composiciones más complejas en el resto de los aspectos de la jerarquía de aspectos.

```
abstract aspect TaskEvent {
	private Task taskRef=null;
	// Operaciones primitivas
	abstract Interface logEvent();<- hot spot

	//pointcuts componentes
	pointcut init(): initialization(Task.new(..));
	pointcut end(): execution (void Task.finalize(..));
	pointcut aspectFlow():cflow(adviceexecution);

	//composiciones de pointcuts
	pointcut initFlow():cflow(init());
	pointcut endFlow(): cflow(end());
	pointcut isATask():if ((taskRef!=null)&&(!taskRef.isComplete()))
	pointcut saveRef(Task t): init()&&this(t);

	after(Task t) : saveRef (t){
		taskRef=t;
	}
}
abstract aspect TaskEscape extends TaskEvent{

	//Pointcut componente
	abstract pointcut notComplete();<- hot spot

	//Composición de pointcuts
		pointcut completeEscape():(call(void java.lang.System.exit(..))&&
	!endFlow()&&!aspectFlow()&& isATask()||(notComplete()&& isAtask());

	before(): completeEscape(){
		taskRef.noFinalize();
		taskRef = null;
	}
}
```

La mayor parte de los pointcuts que conforman las composiciones, fueron definidos en el aspecto abstracto TaskEvent

La mayor parte de los pointcuts que conforman las composiciones, fueron definidos en el aspecto abstracto TaskEvent

```
abstract aspect TaskHelp extends TaskEvent{

	//Pointcut componente
	abstract pointcut accessDocumentation();<- hot spot

	//Composición de pointcuts
	pointcut completeAccessDocumentation():accessDocumentation() && isATask();

	before(): completeAccessDocumentation(){
		taskRef.setTotalAccessDocumentation();
	}
}

abstract aspect TaskException extends TaskEvent{

	//Composiciones de pointcuts
	pointcut initExceptions(Throwable e):
		initFlow()&&!aspectFlow()&&handler(Throwable+)&&args(e)
	pointcut executionExceptions(Throwable e):
		!initFlow()&& !aspectFlow()&&handler(Throwable+)&&args(e)&&isATask();
```

La mayor parte de los pointcuts que conforman las composiciones, fueron definidos en el aspecto abstracto TaskEvent

```
        before(Throwable e) : initExceptions (e){
                taskRef.setTotalExceptions();
                logEvent().addException(taskRef.getTotalExceptions(), reg);
        }
        before(Throwable e) : executionExceptions (e){
                taskRef.setTotalExceptions();
                logEvent().addException(taskRef.getTotalExceptions(), reg);
        }
}
```

6. EXPERIENCIAS

Inicialmente, el framework AJMU-WEB, extiende sistemas cuya arquitectura responde al patrón de diseño Fachada (Gamma E. et al. 1994). El patrón Fachada, simplifica el acceso a un conjunto de clases, o servicios, proporcionando una única clase que los clientes utilizan para consumirlos. Dicho de otra manera, una Fachada, es un objeto "Front-End" que es el único punto de entrada para los servicios de un sistema; la implementación y otros componentes del sistema son privados y no son visibles por los componentes externos. Mediante este patrón, la complejidad de un sistema es ocultada detrás de una clase con la finalidad de facilitar su utilización. La clase Fachada, por lo tanto, conoce las clases responsables de una determinada operación y delega las peticiones de los clientes a los objetos apropiados del sistema.
Las experiencias realizadas, tuvieron por finalidad evaluar la capacidad del framework de extender una aplicación web para registrar, de manera automática, datos asociados a la usabilidad de una aplicación web determinada.

6.1. Aplicación seleccionada para la experiencia

Para la experiencia de AJMU-WEB, se ha empleado una aplicación web, desarrollada en JSP, cuya arquitectura implementa el patrón de diseño Fachada. La aplicación seleccionada, se denomina JForum. Se trata de un poderoso y robusto sistema de foros escrito en java. En la siguiente tabla, se detallan algunos datos de la aplicación.

Ambiente de producción	Leguaje de desarrollo	Java Server Pages (JSP)
	Servidor web	Tomcat
	Motor de base de datos	PostgreSQL
Métricas estáticas de la complejidad del software	Número de clases	381 clases
	Número de paquetes	53 paquetes
	Número de interfaces	42 interfaces
	Número de métodos	2.676 métodos
	Líneas de código	31.660 líneas de código

Tabla 3 – Características del jforum.

6.2. Listado de Tareas

Las tareas que fueron configuradas, en el framework, son tres y se describen en la siguiente Tabla:

T 1	**Tarea 1:** Creación de un tema en el foro	
	En esta tarea, el usuario deberá ingresar al sistema de foros JForum y crear un nuevo tema en el foro titulado "Evaluación de Usabilidad". Luego deberá asignar un título significativo al tema y una breve descripción sobre el tema central de discusión que se desea desarrollar. En la descripción, se solicitará al menos: insertar una imagen y dar formato al texto aplicando alineación y colores a las frases o palabras destacadas.	
	Inicio:	La tarea se considerará iniciada luego de que el usuario haya seleccionado la opción "New Post".
	Fin:	La tarea se considerará finalizada luego de que el usuario envíe el formulario y aparezca la publicación en el listado de temas del foro "Evaluación de Usabilidad".
T	**Tarea 2:** Crear un grupo de usuarios y ajustar sus permisos	

2	Para realizar esta tarea, el usuario deberá ingresar al sistema de foros JForum con privilegios de administrador. Luego de que se haya registrado exitósamente, desde el Panel de Administrador, deberá crear un nuevo Grupo denominado Moderadores. Una vez creado el nuevo Grupo, deberá ajustar sus permisos para permitir a sus miembros administrar los temas y respuestas publicados en el foro "Evaluación de Usabilidad".	
	Inicio:	La tarea se considerará iniciada luego de que el usuario haya seleccionado la opción "Insert", en la Interfaz de Administración de Grupos.
	Fin:	La tarea se considerará finalizada luego de que se hayan ajustado los permisos sobre el grupo Moderadores, y guardado los cambios.
T 3	**Tarea 3:** Crear un usuario y agregarlo al grupo de Administradores	
	Para realizar esta tarea, el usuario deberá ingresar al sistema de foros JForum con privilegios de administrador. Luego de que se haya registrado exitosamente, desde el Panel de Administrador, deberá crear un nuevo usuario. Una vez creado el nuevo usuario, deberá incorporarlo al grupo de Administradores.	
	Inicio:	La tarea se considerará iniciada luego de que ingrese a la interfaz de Administración de Usuarios.
	Fin:	La tarea se considerará finalizada luego de que haya sido incorporado el nuevo usuario al grupo de Administradores.

Tabla 4 – Tareas definidas en Ajmu-web para la experiencia.

6.3. Descripción de la experiencia

La evaluación de la usabilidad de las tareas se realizó por el periodo de 1 (un) día. Luego se analizó el archivo del log, generado a partir de los datos recolectados durante la ejecución de las tareas, y se obtuvieron los siguientes resultados.

En la tabla 5, se presentan los datos obtenidos del perfil de cada usuario.

Usuarios	**Perfil**		**Tareas**		
	Sexo	**Edad**	**T 1**	**T 2**	**T3**
U1	Femenino	38	x	x	
U2	Masculino	33	x		x
U3	Femenino	24		x	x

Tabla 5 – Asignación de tareas por participante.

En la tabla 6, se presenta un resumen de la actividad realizada por los usuarios con respecto a las tareas definidas para la evaluación. La misma ha sido obtenida a partir del análisis de los datos extraídos del log. Con respecto a las métricas de Satisfacción (M4, M5 y M6), las mismas fueron ponderadas por los usuarios empleando un rango de valores discretos comprendidos entre 1 y 5, donde 1 expresa el máximo grado de insatisfacción, y 5 el mayor grado de satisfacción.

Métrica		**U1**		**U2**		**U3**	
		T1	**T2**	**T1**	**T3**	**T2**	**T3**
		No Finalizado	Finalizado	Finalizado	Finalizado	No Finalizado	Finalizado
M1	Cantidad de Excepciones producidas durante la ejecución de la tarea	3	0	0	2	1	2
M2	Cantidad de Accesos a la ayuda producidos durante la ejecución de la tarea	-	-	-	-	-	-
M3	Tiempo en realizar la tarea (ms)	-	234971	148727	51058	-	154538
M4	Satisfacción por la complejidad de la tarea realizada (sat1)	-	2	4	2	-	2
M5	Satisfacción por el tiempo requerido en realizar la tarea (sat3)	-	2	3	2	-	3
M6	Satisfacción por el uso de la aplicación para realizar la tarea (sat 2)	-	4	2	3	-	2
M7	Cantidad de Tareas finalizadas	1		2		1	
M8	Cantidad de Tareas no finalizadas	1		0		1	

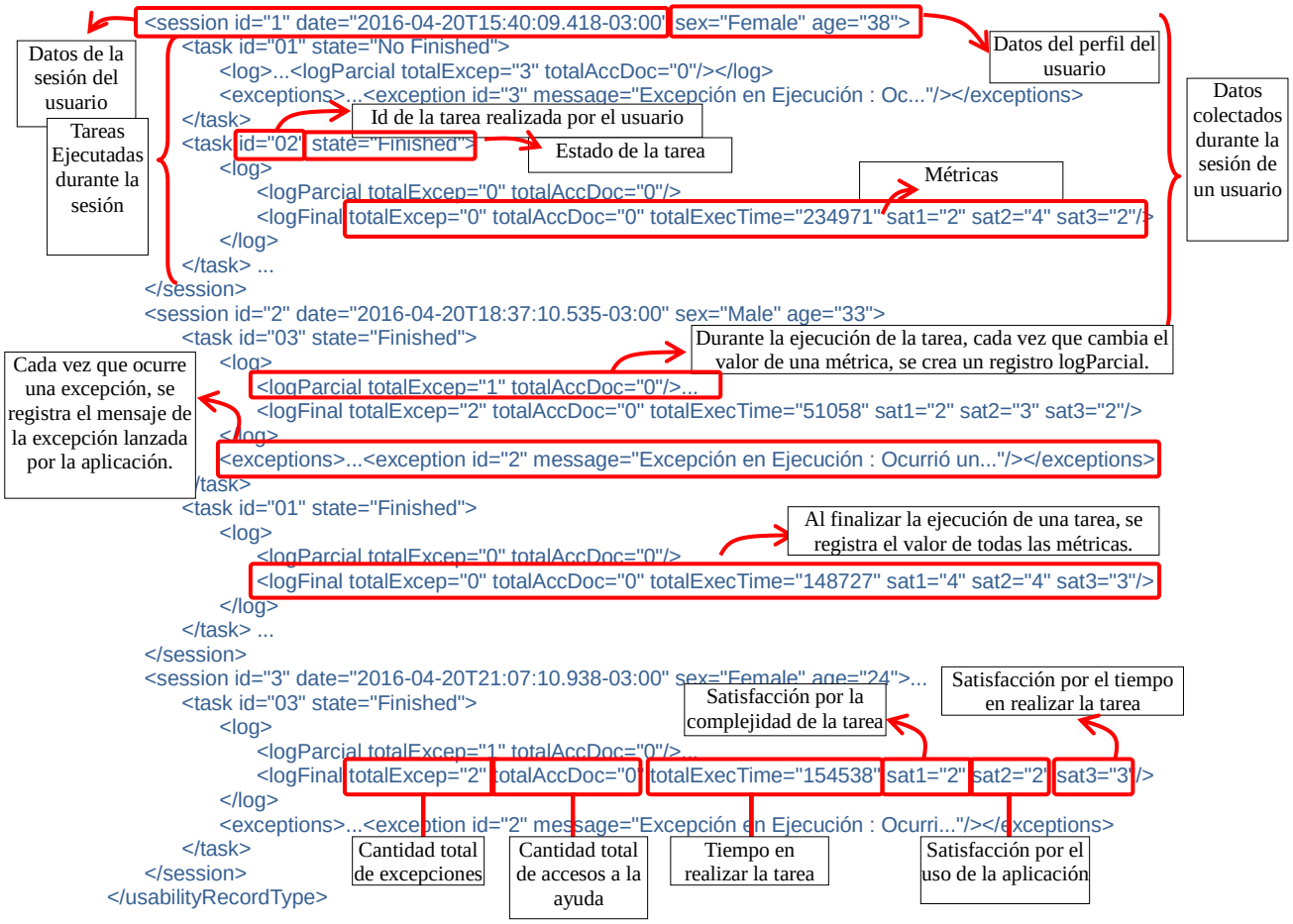
Tabla 6 - Resultados obtenidos de la experimentación

A continuación, se presentan fragmentos del log generado por el framework, en el cual se han resaltado aspectos relevantes de su estructura y contenido extraído.

```
<usabilityRecordType xmlns="http://www.example.org/usabilityrecord" appName="jforum" version="2.1.8">
<tasksAnalized id="03" name="Tarea 3: Crear un usuario y agregarlo al grupo de administradores"/>
<tasksAnalized id="02" name="Tarea 2: Crear un grupo de usuarios y ajustar sus permisos"/>
<tasksAnalized id="01" name="Tarea 1: Creación de un tema en el foro"/>
```

Datos de la aplicación web evaluada

Tareas definidas por el evaluador. Para cada tarea se indica un id, y un nombre



7. CONCLUSIONES

A través de la AOP, ha sido posible el desarrollo de un framework de dominio capaz de extender aplicaciones web, de manera no invasiva, con la finalidad de registrar datos específicos y relevantes para el proceso de evaluación de usabilidad. Estos datos, que corresponden a diferentes métricas que han sido definidas para cada una de las dimensiones de usabilidad propuestas por la ISO 9241 (efectividad, eficiencia y satisfacción), son recopilados de manera automática en el contexto de ejecución de una tarea de usuario.

El framework AJMU-Web, posee un nivel de abstracción suficiente para permitir, a los programadores/evaluadores, definir el conjunto de tareas de usuario que se emplearán durante el proceso de evaluación de la usabilidad, lo cual se logra extendiendo alguno de sus módulos. De la misma manera, a través de un proceso similar, es posible redefinir aquellos eventos que se desean capturar y que dependen de la aplicación que será evaluada. En principio estos eventos son: situaciones de abandono de una tarea, accesos a la documentación o ayuda, y ocurrencia de excepciones lanzadas por la aplicación. Sin embargo, esta flexibilidad para conectar el framework con aplicaciones web, se encuentra limitada a aquellas aplicaciones que aplican el patrón de diseño fachada para la gestión de request o solicitudes.

En relación a los eventos, debido a la naturaleza inherente a las aplicaciones web, basadas en una arquitectura cliente - servidor, y al lenguaje utilizado para el desarrollo del framework (Aspectj), los eventos que ocurren en el cliente, y que son gestionados por el lenguaje de programación javascript, no pueden ser capturados por AJMU-Web. Esta situación implica una limitación a considerar, dado que es en el cliente donde la interfaz de la aplicación se despliega y, por ende, la interacción con el usuario también tiene lugar.

Por otro lado, la estructura jerárquica de sus módulos, fundamentada por la aplicación de patrones de diseño, le otorgan al framework un alto grado de calidad en términos de reusabilidad, mantenibilidad, y escalabilidad. Tal es así, que es posible definir nuevos tipos de eventos a ser capturados durante la ejecución de una tarea, extendiendo los aspectos de mayor nivel en la jerarquía de aspectos; o incluso definir nuevos formatos para el registro de log (actualmente se ofrecen los formatos de texto plano y xml).

Con respecto a las precondiciones impuestas por el framework, requiere de un mínimo conocimiento sobre el lenguaje de programación orientado a aspectos por parte de los desarrolladores/evaluadores que deberán definir los eventos que se van a capturar, las tareas de usuario, y la configuración mínima requerida para el correcto funcionamiento de AJMU-Web. Sin embargo, dado que la utilización del framework implica, entre otras tareas, la definición de un conjunto de pointcuts, los desarrolladores deberán tener un conocimiento profundo de la estructura interna de la aplicación web que será evaluada.

En un futuro inmediato, se espera ampliar el espectro de aplicaciones web susceptibles de ser evaluadas con la asistencia del framework AJMU-Web que, inicialmente, sólo es aplicable a aquellas aplicaciones que gestionan sus solicitudes o request siguiendo el patrón de diseño fachada. Así mismo, se desea mejorar la usabilidad propia del framework de tal manera de permitir su utilización directamente por parte de los Ingenieros de Usabilidad sin depender de la disponibilidad de un desarrollador para la configuración de las pruebas de usabilidad.

8. BIBLIOGRAFÍA

ARPAIA P., BERNARDI M., DI LUCCA G., INGLESE V., SPIEZIA G. (2008) Aspect oriented based software synchronization in automatic measurement systems. In Proceedings of Instrumentation and Measurement Technology Conference, IMTC 2008, IEEE, 1718–1721, 12–15.

ALEXANDER C., ISHIKAWA S., SILVERSTEIN M., JACOBSON M., FIKSDAHL-KING I., ANGEL S. (1977) A Pattern Language. Oxford University Press, NewYork.

ASPECT-Oriented Software Development, 2004. http://aosd.net

BYNENS M., JOOSEN W. (2009) Towards a pattern language for Aspect-Based Design. In Proceedings of the 1st Workshop on Linking Aspect Technology and Evolution (PLATE ‘09), USA. ACM, p. 13–15.

DEMARCO T. (1986) Controlling Software Projects: Management, Measurement and Estimation. USA.

GAMMA E., JOHNSON R., HELM R., VLISSIDES J. (1994) Design Patterns. Elements of Reusable Object-Oriented Software – Addison Wesley

DEUTSCH L. (1989) Design reuse and framework in the Smalltalk-80 system. Software Reusability, vol II: Applications and Experience, pp 57-71. Addison-Wesley, Reading MA.

FERNANDEZ A. (2009) WUEP: Un Proceso de Evaluación de Usabilidad Web Integrado en el Desarrollo de Software Dirigido por Modelos. Tesina de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información (ISMFSI)

FLEURY M., REVERBEL F. (2003) The JBoss extensible server. In Proceedings of the 4th ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware’03). Vol. 2672 of Lecture Notes in Computer Science. Springer-Verlag, p. 344–373.

GREENWOOD P., LOUGHRAN N., SURAJBALI B., COULSON G., RASHID A., PESSEMIER N., SEINTURIER L., PAWLAK R., TRUYEN E., SANEN F., LAGAISSE B., GREGOIRE J., BYNENS M., JOOSEN W., JACKSON A., CLARKE S., HATTON N.,PINTO M., FUENTES L., AMOR M., COHEN T., COLYER A., SCHWANNINGER C., (2007) Validation of the Reference Architecture. Lancaster University: Lancaster. p. 1-38.

HOLZINGER A., BRUGGER M., SLANY W. (2011) Applying Aspect Oriented Programming in Usability Engineering Processes - On the Example of Tracking Usage Information for Remote Usability Testing. Proceeding of the 8th. International Conference on electronic Business and Telecommunications. España 53-56

HÜRSCH W., LOPES C. (1995) Separation of Concerns. Northeastern University, TR NU-CCS-95-03, USA.

HANENBERG S., SCHMIDMEIER A. (2003) Idioms for building software frameworks in AspectJ. In Proceedings of the Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS) at AOSD. Boston, Massachusetts, USA, March 17 - 21, 2003. ACM.

HANENBERG S., UNLAND R., SCHMIDMEIER A. (2003) AspectJ Idioms for Aspect-Oriented Software Construction. In Proceedings of 8th European Conference on Pattern Languages of Programs (EuroPLoP), Germany, 25th–29th June, p. 617–644.

IVORY M., HEARST M. (2001) The state of the art in automating usability evaluation of user interfaces. Journal ACM Computing Surveys (CSUR), vol. 33, no. 4, pp. 470−516

ISO (1998) International Standard - ISO 9241-11:1998 - Ergonomic requirements for office work with visual display terminals (VDTs). Guidance on usability

ISO (2000) International Standard - ISO / IEC 9126-1:2000 - Information Technology - Software product quality.

JOHNSON R., FOOTE B. (1988) Design a reusable classes. Journal of Object-Oriented Programming, 1(2):22-35.

KULESZA U., ALVES V., GARCIA A., DE LUCENA C., BORBA P. (2006) Improving Extensibility of Object-Oriented Frameworks with Aspect-Oriented Programming. In Proceedings of Intl Conference on Software Reuse (ICSR), Italy, p. 231–245.

KICZALES G., HILSDALE E., HUGUNIN J., KERSTEN M., PALM L., GRISWOLD, W. (2001) An overview of AspectJ. Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP).

KICZALES G., LAMPING J., MENDHEKAR A., MAEDA C., LOPES C., LOINGTIER J., IRWIN J. (1997) Aspect-oriented Programming, In Proceedings of the European Conference on Object-Oriented Programming (ECOOP), LNCS 1241, Springer-Verlag

KRONBAUER A., SANTOS C. (2011) Um modelo de avaliação da usabilidade baseado na captura automática de dados de interação do usuário em ambientes reais, In Proceedings of the 10th Brazilian Symposium on on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction - Pages 114-123

KRONBAUER A., SANTOS C., VIEIRA V. (2012) Um Estudo Experimental de Avaliação da Experiência dos Usuários de Aplicativos Móveis a partir da Captura Automática dos Dados Contextuais e de Interação. IHC´12, Braziliam Symposium on Human Factors in Computing Systems. Brazil

LADDAD R. (2003) AspectJ in Action: practical aspect-oriented programming. Manning Publications Co.

LADDAD R. (2010) AspectJ in Action. Second Edition: Enterprise AOP with Spring Applications. Manning Publications Co.

LOUGHRAN N., COULSON G., SEINTURIER L., PAWLAK R., TRUYEN E., SANEN F., BYNENS M., JOOSEN W., JACKSON A., CLARKE S., HATTON N., PINTO M., FUENTES L., AMOR M., COHEN T., COLYER A., SCHWANNINGER C. (2005) "Requirements and Definition of AO Middleware Reference Architecture", AOSD-Europe Report, Deliverable D21.

LAGAISSE B., GREENWOOD P., TRUYEN E., RASHID A., COULSON G., JOOSEN W. (2006) D67: Design of Frameworks for Aspects addressing of the Key Concerns. AOSD-Europe-KUL-10 – AOSD-Europe.

LETTNER F., HOLZMANN C. (2012) Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. The 10th International Conference on Advances in Mobile Computing & Multimedia, MoMM '12, Indonesia 118-127

MILES R. (2004). AspectJ Cookbook. O'Reilly Media.

MATERA M., RIZZO F, TOFFETTI G. (2006) Web engineering - Web Usability: Principles. Evaluation Methods. by Springer Berlin Heidelberg p 143-180

MENKYNA R., VRANIC V., POLÁŠek I. (2010) Composition and categorization of aspect-oriented design patterns. In Applied Machine Intelligence and Informatics (SAMI), 2010 IEEE 8th International Symposium on. Herlany, Janury 28-30, 2010. IEEE, pp. 129–134.

NIELSEN J. (1993), Usability Engineering. AP Professional.

NOBLE J., SCHMIEDMEIER A., PEARCE D., BLACK, A. (2007) Patterns of Aspect-Oriented Design. In L. B. Hvatum and T. Schümmer (Eds.), EuroPLoP. Irsee, Germany, July 4-8, 2007. UVK - Universitaetsverlag Konstanz, pp. 769–796.

PIVETA E., ZANCANELA L. (2003) Aspect Weaving Strategies. Journal of Universal Computer Science, vol.9, no. 8.

RAUSCH A., RUMPE B., HOOGENDOORN L. (2003) Aspect-Oriented Framework Modeling. In Proceedings of the 4th AOSD Modeling with UML Workshop, UML Conference.

SANTOS A., LOPES A., KOSKIMIES K. (2007) Framework specialization aspects. In Proceedings of AOSD '07 the 6th international conference on Aspect-oriented software development, ACM USA, p. 14–24.

KUCHANA P. (2004) Software Architecture Design Patterns in Java – Partha Kuchana – Auercach Publications – by CRC Press LLC – Boca Raton London New York Washington, D.C. ISBN 0-8493-2142-5

SANEN F., TRUYEN E., DE WIN B., LOUGHRAN N., RASHID A., CHITCHYAN R., LEIDENFROST N., FABRY J., CACHO N., GARCIA A., JOOSEN W., BOUCKÉ N., HOLVOET T., JACKSON A., NEDOS A., HATTON N., MUNNELLY J., FRITSCH S., CLARKE S., AMOR M., FUENTES L., PINTO M., CANAL C. (2006) "A Domain Analysis of Key Concerns – Known and New Candidates", AOSD-Europe Report, Deliverable D43.

TAHIR A., AHMAD R. (2010) An AOP-Based Approach for Collecting Software Maintainability Dynamic Metrics. IEEE Second International Conference on Computer Research and Development, pp. 168–172.

TARBY J., EZZEDINE H., ROUILLARD J., TRAN C., LAPORTE P., KOLSKI C. (2007) Traces Using Aspect Oriented Programming and Interactive Agent-Based Architecture for Early Usability Evaluation: Basic Principles and Comparison by: In HCI (1), Vol. 4550, pp. 632-g41, doi:10.1007/978-3-540-73105-4_70

TARTA A., MOLDOVAN G. (2006) Automatic Usability Evaluation Using AOP, 2006 Automation, Quality and Testing, Robotics, IEEE International Conference on (Volume:2)

VAIRA Ž., ČAPLINSKAS A. (2011) Paradigm independent design problems, GoF 23 design patterns and aspect design. Informatica, 22(2)

VOCES M. (2011) Rich Internet Applications (RIA) y Accesibilidad Web [on line]. "Hipertext.net", núm. 9, 2011. http://www.upf.edu/hipertextnet/numero-9/ria-accesibilidad-web.html

WCAG (2008) Web Content Accessibility Guidelines 2.0 - W3C Recommendation - http://www.w3.org/TR/WCAG20/

YONGLEI T. (2008) Automated Data Collection for Usability Evaluation in Early Stages of Application Development 7th WSEAS in ACACOS 08, China.

YONGLEI T. (2012) Aspect-Oriented Instrumentation for Capturing Task-Based Event Traces Int. J. on Control System and Instrumentation, Vol. 03, N0. 01.