ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr

FUsaM: Framework, con base en una SPL, para la medición de usabilidad en aplicaciones móviles

FUsaM: Framework for Measuring Usability Mobile Applications Based on an SPL

Juan G. Enríquez
jenriquez@unpa.edu.ar
GISP – Instituto de Tecnología Aplicada
Universidad Nacional de la Patagonia Austral
Río Gallegos, Argentina

Sandra I. Casas
scasas@unpa.edu.ar
GISP – Instituto de Tecnología Aplicada
Universidad Nacional de la Patagonia Austral
Río Gallegos, Argentina

Recibido-Received: ***21/set/2015*** / Aceptado-Accepted: ***1/dic/2015*** / Publicado-Published: ***31/jul/2016***

Resumen
El uso masivo de los dispositivos móviles, asociado a la heterogeneidad de estos y también de los usuarios, implica un desafío al momento de evaluar la usabilidad de las aplicaciones móviles. Es importante disponer de metodologías y herramientas que permitan realizar estudios de usabilidad específicos para este tipo de aplicaciones en las cuales el contexto de uso cambia continuamente. Es necesario que las pruebas de usabilidad sean transparentes para el usuario, recolecten datos de usabilidad y del contexto de manera automática, no sean intrusivas para la aplicación que se prueba y que soporten en alguna medida la variabilidad de los dispositivos. Para lograr las características mencionadas, en este trabajo se propone un framework denominado FUsaM (Framework de usabilidad móvil) que es extensible y permite generar e integrar pruebas de usabilidad en aplicaciones móviles. Para su diseño e implementación se utiliza el enfoque de construcción de Línea de productos de software (SPL, Software Product Line) combinado con la Programación orientada a características (FOP, Feature-Oriented Programming) y la Programación orientada a aspectos (AOP, Aspect-Oriented Programming). También se presenta un caso de estudio para demostrar la funcionalidad de este.

Palabras claves: Usabilidad, aplicaciones móviles, pruebas, línea de productos de software.

Abstract
The widespread use of mobile devices and the heterogeneity of the users involves challenges when evaluating usability of mobile applications. It is important to have methodologies and tools that enable to perform specific usability studies for this type of applications in which the context of use is constantly changing. It is necessary that usability testing are transparent to the user, usability and context data are automatically collected, the test code are not intrusive in the tested application and the tests support the variability of the devices. To achieve the above features, this article presents a framework called FUsaM (Mobile Usability Framework) which is extensible and can generate and integrate usability testing in mobile applications. The design and implementation approach is based on Software Product Line (SPL) combined with Features Oriented Programming (FOP) and Aspect Oriented Programming (AOP). A case study is also presented to prove the functionality.

Keywords: Usability, mobile applications, testing, software product line.

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr
ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

La evaluación de la usabilidad (Hornbæk, 2006; ISO 9241-11, 1998) de una aplicación de software consiste en realizar pruebas para obtener medidas e información de la interacción del usuario con la aplicación y observar debilidades relacionadas con el uso de esta misma. El proceso de ingeniería de usabilidad resulta costoso y consume tiempo. El uso de pruebas automáticas y flexibles reduce el tiempo y los costos de las evaluaciones de usabilidad e impacta, directamente, en el éxito económico de los productos.

El proceso de evaluación de usabilidad consta de las siguientes actividades (Ivory y Hearst, 2001): La preparación de la prueba, que consiste en configurar la aplicación o el prototipo para habilitar el registro de los datos necesarios para la evaluación; la captura, que realiza la recolección de las métricas de usabilidad, tales como el tiempo de terminación de una tarea, errores, valores subjetivos, etc. y, por último, el análisis de las métricas capturadas para identificar problemas de usabilidad y sugerir soluciones o mejoras para mitigar los problemas encontrados.

Las pruebas y métricas actualmente utilizadas para medir usabilidad fueron creadas para aplicaciones de escritorio; sin embargo, estas pueden no ser directamente adecuadas o apropiadas a entornos móviles (Zhang y Adipat 2005). Uno de los desafíos consiste en identificar las variables adicionales relacionadas con el contexto de uso que pueden impactar en la usabilidad de una aplicación móvil. Además, es necesario que estas pruebas sean fáciles de realizar para que puedan ser incorporadas en las prácticas habituales de un desarrollador.

Por lo indicado, el propósito de este trabajo es proponer una solución para desarrollar pruebas flexibles, no intrusivas y automáticas, que permitan evaluar la usabilidad de aplicaciones móviles a partir de los diferentes requerimientos de los evaluadores y características de los dispositivos.

Se presentan, como contribuciones principales, una línea de productos de software (SPL) (Software Engineering Institute, Software Product Lines, http://www.sei.cmu.edu/productlines/) para generar pruebas de usabilidad para aplicaciones móviles, cuya representación teórica y abstracta se realiza por medio del modelo de características (Batory 2005); el framework FUsaM que implementa el modelo y permite generar automáticamente una familia de pruebas de usabilidad para aplicaciones móviles e integrarlas a estas para la posterior ejecución; y un caso de estudio sobre una aplicación móvil real y distintas configuraciones.

El resto de este trabajo se organiza de la siguiente manera: Primero se indica la estrategia de investigación y los aspectos conceptuales, metodológicos e instrumentales aplicados. Luego se presentan los resultados: el modelo conceptual definido para soportar la SPL, la estructura del framework FUsaM y un caso de estudio sencillo para demostrar su funcionalidad. Por último, se efectúa la discusión y conclusiones de la propuesta.

Materiales y métodos

La estrategia de investigación se encuadra, según los preceptos de Shaw (2003), en la combinación, pregunta: método o medio de desarrollo, resultado: herramienta, y validación: experiencias.

A continuación se describen los aspectos conceptuales, metodológicos e instrumentales que se han aplicado en este trabajo.

Conceptos, métodos e instrumentos

Una SPL es "un conjunto de sistemas de software (productos) que comparten un conjunto de características, las cuales satisfacen las necesidades especificas de un dominio o segmento particular del mercado, y que se desarrollan a partir de un sistema común (core) de una

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr

manera preestablecida" (Clements y Northrop, 2001, p. 608). Los productos de una SPL poseen un conjunto de características en común, pero cada producto difiere de otro en el conjunto de características opcionales que implementa. Esta diferencia entre productos de una SPL es conocida como la variabilidad de una SPL (Pohl, Böckle y Linden, 2005).

Una metodología para modelar este tipo de sistemas es la FOP. Esta permite la representación de las similitudes y las variaciones de un sistema de software. El concepto principal de esta técnica es denominado *feature*. Las *features* son características identificables unívocamente de un dominio de aplicación, según el punto de vista del usuario o desarrollador. Es la unidad funcional en la que se descompone un sistema de software para satisfacer un requisito y proporciona una potencial opción de configuración.

Esta metodología utiliza lo que se denomina un modelo de características (FM - feature model) que describe todas las posibles variantes o configuraciones de productos de software que se pueden obtener. Para ello, los FM organizan el conjunto de características jerárquicamente, mediante relaciones entre ellas. En la figura 1 se observa un ejemplo de FM en el dominio de los hogares inteligentes (SmartHome).

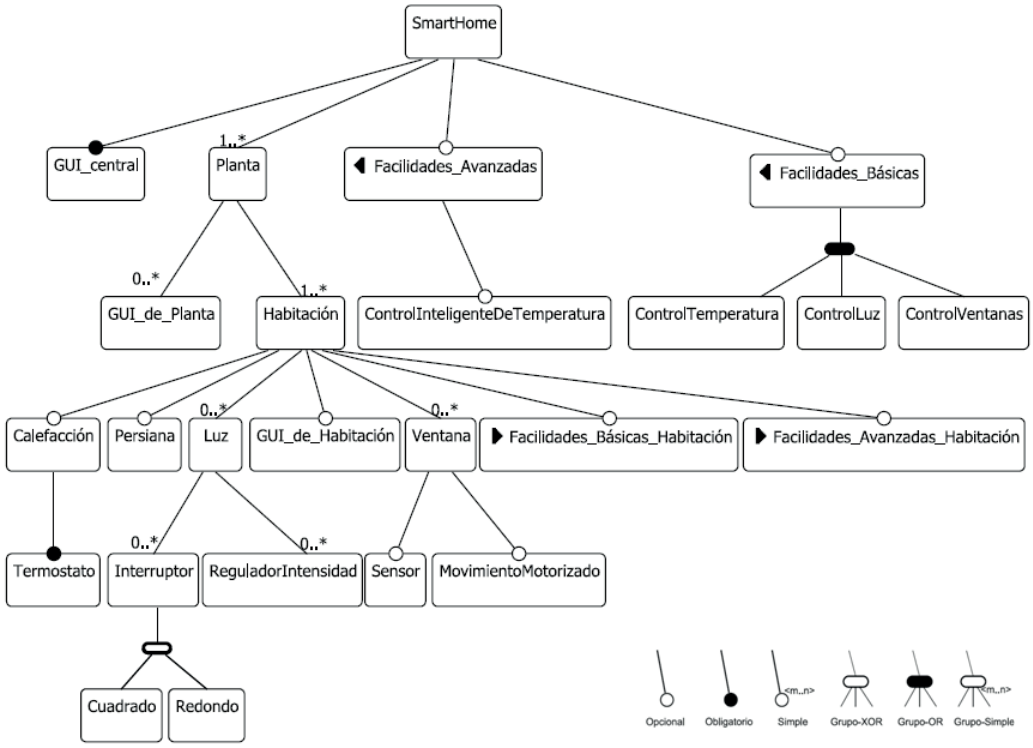

Figura 1. **Modelo de características (SmartHome),** En **Salazar (2009).**

El ejemplo corresponde a un SmartHome que tiene un número variable de plantas y habitaciones que ofrecen servicios categorizados en básicos y avanzados. Un hogar inteligente tiene al menos una planta, cada planta puede tener un número indeterminado de interfaces de usuario, así mismo, cada planta contiene al menos una habitación en la cual se puede controlar o no la calefacción, persiana, un número variable de luces y ventanas.

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr
ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

En FOP una configuración es un subconjunto de todas las características definidas en el modelo de características. Una configuración es válida, si la combinación de características es permitida por el modelo. De lo contrario, la configuración es llamada inválida.

Para implementar un FM existen varias técnicas, entre ellas la AOP. La AOP proporciona mecanismos y abstracciones para representar las preocupaciones transversales (crosscutting concerns) de una aplicación, de manera separada y aislada. Las herramientas AOP permiten la descomposición de estas preocupaciones transversales en unidades o módulos denominados aspectos; tiene la ventaja de permitir agregar, de manera no intrusiva, funcionalidad a una aplicación base. Los elementos más importantes de esta técnica son:

1. Un punto de unión (join point) indica un posible punto del sistema en el cual el código definido por un aspecto puede insertarse.
2. Un punto de corte (pointcuts) especifica un conjunto de puntos de unión.
3. Un aviso o consejo (advices) es una pieza de código que se inserta en un punto de corte.

Combinando FOP con AOP se pueden modularizar las características en aspectos y usar un tejedor (weaving) de AOP para integrar las características en diferentes instancias del producto de software.

El framework FUsaM fue desarrollado sobre el IDE Eclipse (The Eclipse Foundation, https://eclipse.org/) con soporte para FOP mediante FeatureIDE (Framework for Feature-Oriented Software Development) (Thüm, Kästner, Benduhn, Meinicke, Saake y Leich, 2014). FeatureIDE es un framework de código abierto, basado en Eclipse que soporta varias técnicas para el desarrollo de software orientado a características, una de ellas es la AOP. Como lenguaje de programación para AOP se empleó AspectJ (AspectJ, http://www.eclipse.org/aspectj/) y la herramienta AJDT (AspectJ Development Tool, http://www.eclipse.org/ajdt/).

Resultados

Modelo conceptual

El objetivo de una SPL es proporcionar una infraestructura adecuada para una rápida y fácil producción de sistemas de software similares. Las SPL se pueden ver análogas a las líneas de producción industriales, cuyos productos similares o idénticos se ensamblan y configuran a partir de piezas prefabricadas bien definidas, que son reutilizadas para la construcción de productos con características similares.

Una pieza clave en la creación y desarrollo de una línea de productos es el análisis y especificación de qué elementos son comunes y qué elementos son variables dentro del conjunto de productos similares producidos por la línea de productos de software. Para realizar dicha tarea, la FOP permite construir modelos de características, el modelo posibilita descomponer un sistema en características comunes y variables.

El modelo de características definido para el framework se puede observar en la figura 2. La característica denominada <Usabilidad> es abstracta y se define para estructurar el modelo y no tiene impacto a nivel de implementación. La característica <Tareas> tiene la especificación de las tareas de la aplicación a probar, que van a ser monitoreadas durante la prueba. Las otras características están asociadas con los datos de usabilidad que se necesitan recolectar en las pruebas. Estos datos pueden ser métricas (<Tiempo> <Error> <Batería> <Memoria>), o pueden

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr

ser datos relacionados al ambiente real de uso (<Luminosidad> <Conectividad> <Temperatura> <Movimiento> <Presión> <Humedad>).

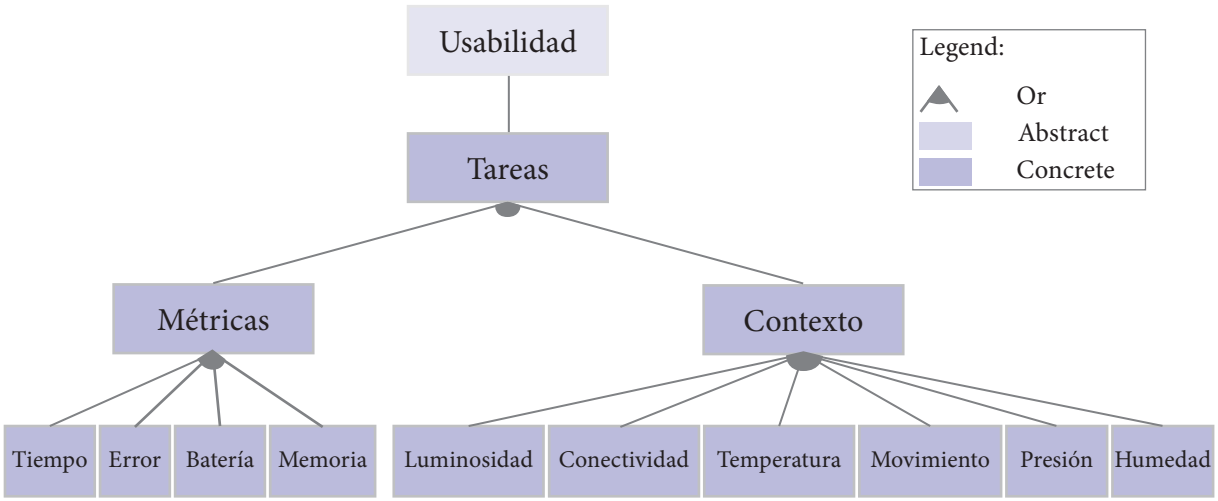

Figura 2. **Modelo de características** (propio del estudio).

A partir del modelo definido, la herramienta de soporte de FOP permite componer diferentes pruebas de usabilidad mediante la selección de características que van a formar parte de estas. Por ejemplo, una prueba derivada del modelo que incluya las características de <Tiempo> y <Presión>, implica que durante la prueba se va a registrar el tiempo en que inicia y finaliza cada tarea y, además, que el dispositivo en el cual se va a ejecutar la misma tiene que contar con un barómetro para que se pueda registrar la presión del ambiente durante el inicio y finalización de las tareas.

Las métricas de usabilidad y los datos del contexto considerados son descriptos en la tabla 1. La estructura del modelo posibilita que nuevas métricas y datos contextuales puedan ser fácilmente incorporados, garantizando, en este sentido, la extensibilidad de este.

Tabla 1
Datos de usabilidad

Dato	Descripción	Tipo
Tiempo	Tiempo empleado en realizar una tarea	M
Error	Error cometido al realizar una tarea	M
Batería	Nivel de batería al efectuar una tarea	M
Memoria	Memoria ocupada por la aplicación al hacer una tarea	M
Luminosidad	Luz ambiental al realizar una tarea	C
Conectividad	Estado de la conectividad a una red de datos al realizar una tarea	C
Temperatura	Temperatura ambiental al realizar una tarea	C
Movimiento	Registra la posición en 2 o 3 dimensiones del dispositivo al realizar una tarea	C
Presión	Presión ambiental del aire al realizar una tarea	C
Humedad	Humedad del ambiente al realizar una tarea	C

Nota: Fuente propia de la investigación.

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr
ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

Los datos de tipo M identifican a aquellos que están relacionados con una métricas de forma directa o indirectamente. Por ejemplo, el dato tiempo puede ser utilizado para obtener diferentes métricas de usabilidad como el tiempo medio en completar una tarea.

Los de tipo C reflejan características del contexto real en el momento que se realizó una tarea. La mayoría de estos se obtiene mediante los sensores de los dispositivos móviles, vale recordar que no todos los dispositivos poseen los mismos sensores, por tanto, se tiene que considerar la variabilidad del hardware.

Framework de soporte a la SPL

La estructura básica del framework se presenta en la figura 3. Se puede observar que existen dos proyectos en el IDE: uno es el proyecto App que contiene el código fuente de la aplicación a probar y el otro es el proyecto Usabilidad que contiene el código que implementa la funcionalidad de FUsaM. Por medio de una simple configuración sobre el IDE, se logra que ambos proyectos sean visibles entre sí para una posterior etapa de integración.

Con AspectJ se implementan las características del modelo, brindando la flexibilidad y transparencia requerida en la generación e integración de las pruebas. Cada característica representada en el modelo (figura 2) esta implementada por medio de un aspecto. Por ejemplo, la característica <Tiempo> representa la métrica que recolecta el tiempo empleado en realizar una tarea y tiene asociado un aspecto llamado Tiempo.aj.

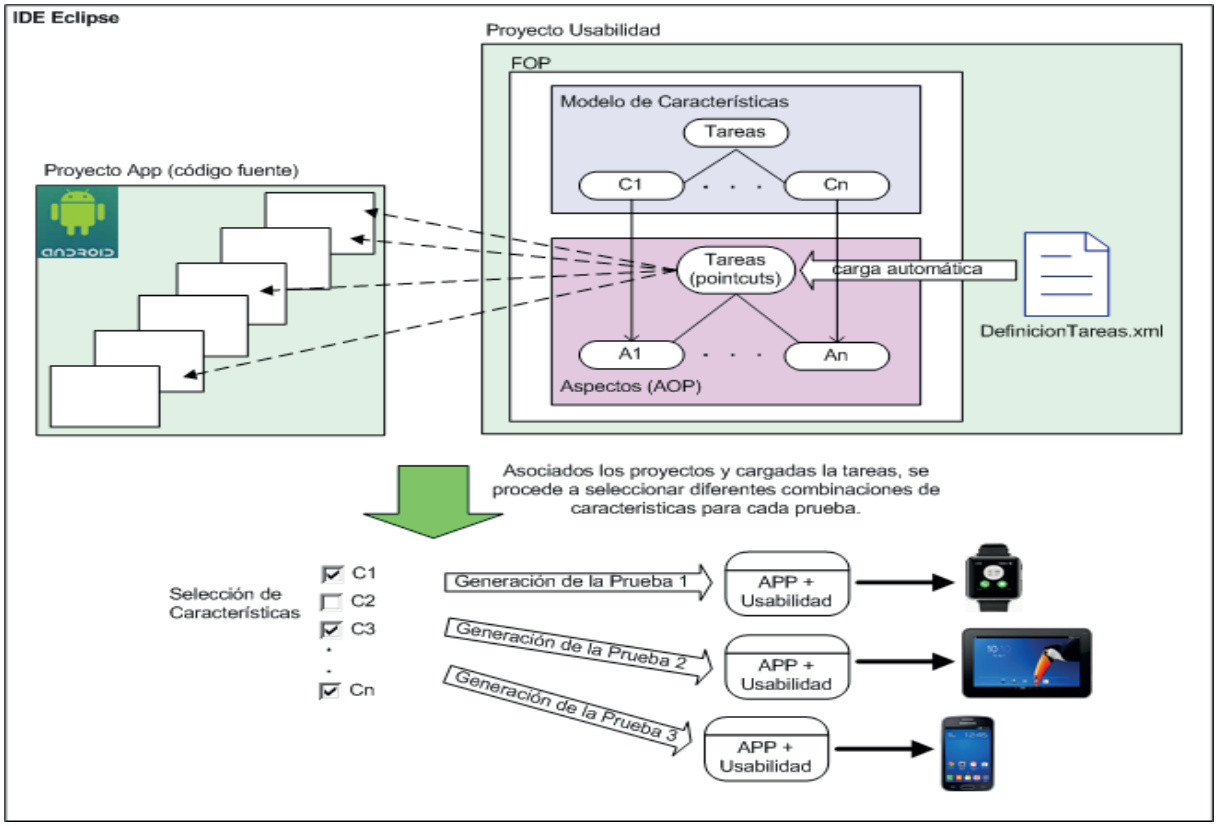

Figura 3. **Estructura del Framework** (propio del estudio).

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr

En el archivo ***DefinicionTareas.xml*** se especifican los nombres de los métodos iniciales y finales de cada tarea que se pretende monitorear. Estos métodos son definidos por el desarrollador de la prueba y se corresponden con la signatura de los métodos del código fuente de la aplicación a probar. A partir de este archivo se genera, automáticamente, el aspecto Tareas.aj que contiene como pointcuts los métodos asociados a las tareas (figura 4).

Una vez que se definieron las tareas y se creó el aspecto correspondiente, el próximo paso consiste en configurar y generar una prueba a partir de la selección de características (datos de usabilidad) que se necesitan formando parte de la misma. Mediante la interfaz que brinda FeatureIDE, representa el FM, se procede a seleccionar los datos de usabilidad. Después de seleccionadas las características, se debe realizar la compilación del proyecto Usabilidad para que se integren, mediante el tejedor de AspectJ, las características seleccionadas con el código fuente de la aplicación a probar en los puntos definidos en el archivo ***DefinicionTareas.xml.***

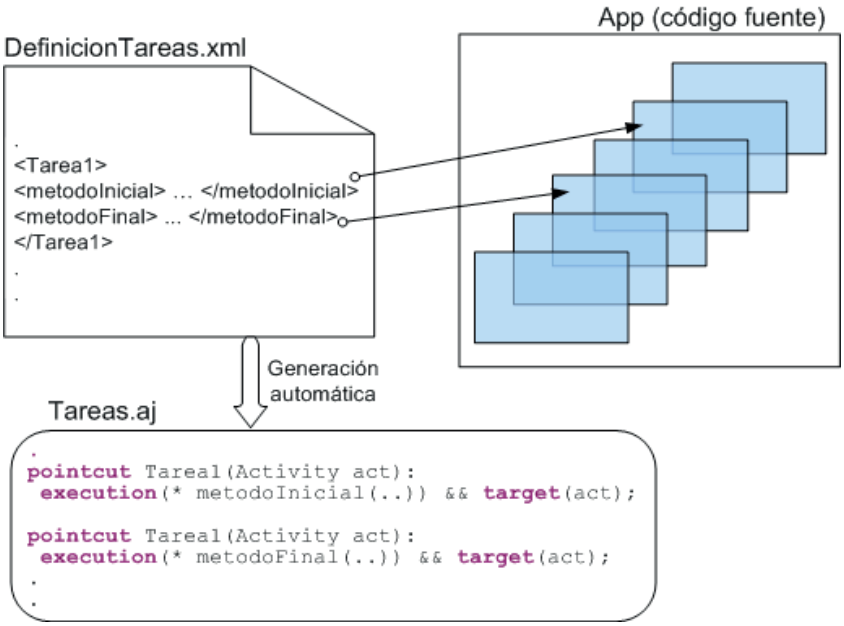

Figura 4. **Especificación de tareas** (propio del estudio).

Por último, se ejecuta el método generarAPK de FUsaM y, de esta forma, la aplicación más la prueba generada en el paso anterior son cargados a un archivo apk (Application Package File) que es un paquete para el sistema operativo Android. Para generar distintas configuraciones de pruebas simplemente se deben seleccionar las características, compilar el proyecto y ejecutar el método mencionado anteriormente.

A continuación se detallan brevemente aspectos instrumentales de FUsaM en cuanto a su preparación, especificación de tareas, configuración e integración y recolección de datos.

Preparación

Inicialmente se debe configurar el IDE Eclipse para incorporar las herramientas de soporte utilizadas por FUsaM y relacionarlo con el código fuente de la aplicación que se requiere probar. Para

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr
ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

esto, primero se debe instalar FeatureIDE y AJDT. Luego, en las Properties del proyecto Usabilidad indicar en AspectJ Build -> Inpath el nombre del proyecto de la aplicación a probar. Por último, en el archivo denominado configuracion.properties especificar los path de diferentes elementos necesarios para el funcionamiento del framework, entre ellos el SDK Android y el JDK Java.

Especificación de tareas

Una técnica al realizar pruebas de usabilidad consiste en monitorear la interacción del usuario con la aplicación en el entorno real de uso. Desde el punto de vista de un desarrollador de pruebas esto consiste en recolectar datos de usabilidad cuando el usuario ejecuta una determinada tarea de la aplicación que se está probando.

En el framework FUsaM la especificación de las tareas que se requiere monitorear se realiza en un archivo de configuración denominado DefinicionTareas.xml, en el cual cada tarea que se define tiene que tener el siguiente formato:

```
<tarea nombre=”TareaInicial-1” tipo=”TI”>
        <pointcut>metodoInicial</pointcut>
</tarea>
<tarea nombre=”TareaFinal-1” tipo=”TF”>
        <pointcut>metodoFinal</pointcut>
</tarea>
```

Esta especificación de tareas en el archivo XML consiste en declarar ciertos métodos, relacionados con las tareas, en el código fuente de la aplicación que se requiere probar. Por cada tarea se deben indicar el método inicial que se invoca cuando comienza la tarea y el método final que delimita la finalización de esta.

Estos parámetros son utilizados para configurar los puntos de corte (pointcut) en donde se va a realizar la integración del código necesario para recolectar los datos de usabilidad. Por medio de AOP, antes de ejecutar el método inicial definido, se va a ejecutar el código que realiza la recolección de los datos de usabilidad configurados en la prueba. Igualmente para el método final, pero en este caso la recolección se realiza después.

Generación e integración

Para la generación e integración de la prueba de usabilidad, se utiliza la herramienta de configuración de FeatureIDE. Esta herramienta permite seleccionar las características que se necesitan tener disponibles en la prueba, es decir, se seleccionan los datos de usabilidad que se van a recolectar durante la ejecución de esta (figura 3).

A esta selección de características, en la FOP se le denomina configuración de un producto, distintas selecciones de características van a generar diferentes configuraciones del producto, para este caso diferentes pruebas de usabilidad. Una vez seleccionada las características, para la generación e integración de la prueba con la aplicación (App+Usabilidad) se debe compilar el proyecto Usabilidad.

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr

La selección de qué datos recolectar puede depender del hardware del dispositivo o de los requerimientos de la prueba. Por ejemplo, se puede generar una prueba para un dispositivo que posee sensor de temperatura y recolectar datos de la temperatura ambiente, así como también se puede generar la misma prueba; pero sin los datos de la temperatura para un dispositivo que no posee dicho sensor.

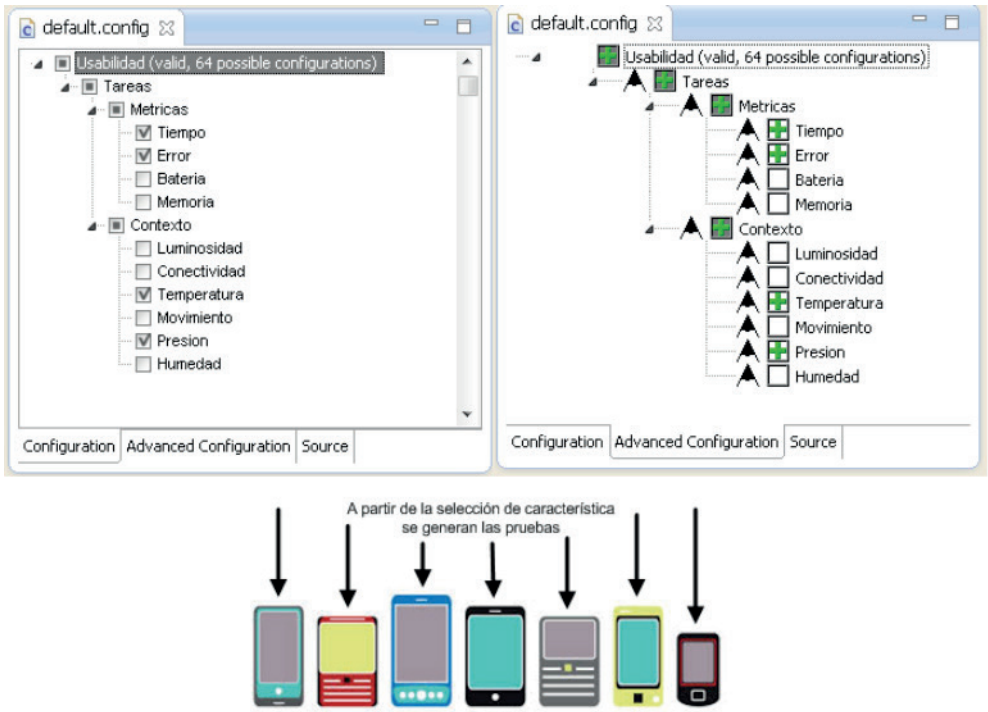

Figura 5. **Generación de la prueba** (propio del estudio).

En la figura 6 se puede observar la implementación de los diferentes aspectos que forman parte de la estructura de integración diseñada para FUsaM. El aspecto Tareas.aj se genera automáticamente a partir del archivo de definición de tareas definicionTareas.xml. Este aspecto se compone de todos los métodos (pointcuts) indicados por el usuario en los cuales se debe realizar la recolección de los datos de usabilidad configurados en la prueba.

Existen dos aspectos abstractos llamados Contexto.aj y Metricas.aj que permiten agrupar los datos de usabilidad en datos del contexto y datos relacionados a métricas, respectivamente; además, de estructurar el framework para desacoplar la definición de los pointcuts de los advices en los que se ejecuta el código de recolección.

Los aspectos asociados a las características encapsulan el código responsable de recolectar los datos de usabilidad. Por ejemplo, la característica relacionada con la métrica del tiempo que tarda en realizarse una tarea se denomina Tiempo.aj. El código de este aspecto registra el tiempo en el momento en que se inicia una tarea y el tiempo en el que finaliza la misma. En la figura 4 estos aspectos están representados de forma genérica con el nombre DatoUsabilidadX y DatoUsabilidadY.

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr
ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

```
public aspect Tareas {
   // aspecto auto-generado por CargarAspectoTareas.java a partir del archivo
definicionTareas.xml
   pointcut TareaInicial-1(Activity act): execution(* metodoInicial(..)) && target(act);
   before(Activity act):TareaInicial-1(act) { ... }
   pointcut TareaFinal-1(Activity act): execution(* metodoFinal(..)) && target(act);
   after(Activity act):TareaFinal-1(act) { ... }
   ........
}
public abstract aspect Contexto {
   pointcut TareaInicio():adviceexecution() && within(Tareas)
      && if(thisJoinPointStaticPart.getSignature().getName().contains("before"));
   pointcut TareaFin():adviceexecution() && within(Tareas)
      && if(thisJoinPointStaticPart.getSignature().getName().contains("after"));
}
public abstract aspect Metricas {
   pointcut TareaInicio():adviceexecution() && within(Tareas)
      && if(thisJoinPointStaticPart.getSignature().getName().contains("before"));
   pointcut TareaFin():adviceexecution() && within(Tareas)
      && if(thisJoinPointStaticPart.getSignature().getName().contains("after"));
}

public aspect DatoUsabilidadX extends Contexto {
   after() : TareaInicio() { // codigo para recolectar el dato de usabilidad X }
   before() : TareaFin() { // codigo para recolectar el dato de usabilidad X }
}

public aspect DatoUsabilidadY extends Metrica {
   after() : TareaInicio() { // codigo para recolectar el dato de usabilidad Y }
   before() : TareaFin() { // codigo para recolectar el dato de usabilidad Y }
}
```

Figura 6. **Estructura de integración de la prueba** (propia del estudio).

Como último paso, para que la integración App+Usabilidad pueda ser ejecutada en un dispositivo móvil o en un emulador, se debe invocar al método generarAPK de FUsaM para obtener el archivo apk. De esta manera, repitiendo los pasos mencionados, se puede generar una familia de pruebas de usabilidad; para cada una de las configuraciones seleccionadas por el desarrollador de las pruebas se produce un archivo apk. Al generar pruebas con esta estructura, el código final que va a ser instalado en el dispositivo solo contiene la aplicación más el código específico de los datos a recolectar elegidos.

Registro de datos recolectados

Una vez instalada la aplicación con la prueba integrada, durante la interacción del usuario con la aplicación en el ambiente real de uso, los datos recolectados, automáticamente, se almacenan localmente en un archivo XML. Este archivo tiene los datos de usabilidad capturados en el inicio y fin de las tareas que fueron definidas para monitorear y tiene la siguiente estructura:

```
<Tarea nombre=Add fecha=2015-02-02 10:20:33>
    <I nombre=conectividad>mobile</I>
    <I nombre=luminosidad>102</I>
    <I nombre=tiempo>1432995573372</I>
    <F nombre=tiempo>1432995597700</F>
```

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr

<F nombre=luminosidad>20</F>

<F nombre=conectividad>mobile</F>

</Tarea nombre=Save fecha=2015-02-02 10:21:00>

Durante la interacción, cada vez que se ejecuta una tarea monitoreada, se asientan los valores de los datos de usabilidad solicitados. En el ejemplo se ve que una tarea inicia con Agregar (Add) y finaliza con Guardar (Save); la etiqueta I indica que los datos recolectados son antes de ejecutar la tarea y la F que los datos fueron tomados luego de ejecutarla.

Con el apoyo automatizado, los datos capturados a nivel de tareas pueden ser identificados de acuerdo con las tareas definidas en la especificación. Los datos de usabilidad, a este nivel, proporcionan la base para mejorar las actividades posteriores de evaluación de usabilidad.

Caso de estudio

A continuación se describe un caso de estudio, el objetivo de este no fue obtener resultados rigurosos, sino validar la funcionalidad del FUsaM. La aplicación que se utilizó para realizar las pruebas de usabilidad se denomina NotePad y pertenece al Android Open Source Project (Android, https://source.android.com).

NotePad (figura 7) es un simple editor de texto que permite crear y guardar notas personales. Las tareas que se monitorearon son las de agregar nota, guardar nota, editar nota, borrar nota, editar título de nota, borra título de nota y descartar cambios. Los métodos relacionados con estas tareas son los que se definen en el archivo DefinicionTareas.xml, para que en la generación de la prueba sean interceptados por los aspectos que recolectan los datos de usabilidad configurados en la selección de características.

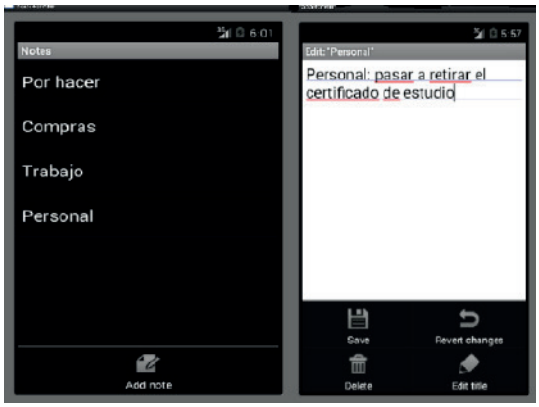

Figura 7. **Aplicación NOTEPAD** (propia del estudio).

Se generaron tres pruebas con diferentes características que se ejecutaron en diferentes dispositivos. Las características de las pruebas y el dispositivo sobre el que se ejecutaron se muestran en la tabla 2.

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr
ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

Tabla 2
Pruebas generadas

Prueba	Características	Dispositivo
1	Tiempo-Error-Batería-Memoria	Nexus One
2	Tiempo-Error-Batería-Memoria Movimiento-Conectividad	Galaxy S2
3	Tiempo-Error-Batería-Memoria Luminosidad-Conectividad Temperatura-Movimiento-Presión Humedad	Galaxy S4

Nota: Fuente propia de la investigación.

La aplicación instalada en un dispositivo móvil tiene un tamaño de 300KB, mientras que la aplicación más la prueba generada instalada tiene los siguientes tamaños: app+prueba1 (316KB), app+prueba2 (320KB) y app+prueba3 (336KB), con estos datos se verifica que el espacio utilizado en el dispositivo por la prueba incorporada a una aplicación no es de consideración, simplemente incrementa unos pocos bytes su tamaño.

La aplicación más la prueba se instaló en los diferentes dispositivos y se procedió a utilizar la aplicación, mientras en segundo plano se ejecutaba la prueba registrando y almacenando los datos de la interacción del usuario con la aplicación. Una vez finalizada la prueba en el archivo usabilidad.xml del dispositivo se encuentran los datos de usabilidad para su posterior análisis. El caso de estudio descrito permitió demostrar lo siguiente:

1. La integración de FUsaM con el proyecto de la aplicación a probar, implementada en Android sobre una plataforma de desarrollo Eclipse.
2. La flexibilidad que ofrece el framework para generar distintas pruebas para distintos dispositivos y diferentes requerimientos del desarrollador de las pruebas.
3. La funcionalidad y ejecución de la aplicación no se alteró por la integración de la prueba en esta misma.
4. Las pruebas almacenaron los datos de usabilidad definidos y necesarios para realizar un análisis de usabilidad, sin reducir el rendimiento del dispositivo utilizado.

Discusión y conclusiones

A continuación se plantea la discusión y conclusiones a partir de los siguientes ejes: comparación con trabajos relacionados, análisis de características sobresalientes de la propuesta y trabajos futuros.

Trabajos relacionados

Los enfoques y metodologías existentes para evaluar la usabilidad en aplicaciones móviles no ofrecen una guía consolidada precisa en definir una representación de usabilidad y de cómo medirla. Existen herramientas propuestas para dar soporte a las pruebas de usabilidad, pero no cubren todos los aspectos relacionados con la usabilidad en un contexto móvil, además de tener cierto grado de dificultad al momento de utilizarlas.

Las herramientas existentes, como por ejemplo Flurry Analytics (Flurry, http://www.flurry.com), Localytics (Localytics, http://www.localytics.com), Mixpanel (Mixpanel, http://www.

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

mixpanel.com), EVA Helper Framework (Balagtas-Fernandez y Hussmann, 2009), VizWear (Lyons y Starner, 2001), solo consideran algunos factores involucrados en la usabilidad de una aplicación móvil, y dejan de lado lo relacionado con el contexto de su uso. De esta forma, al no contar con información del entorno, se hace un análisis sesgado de la usabilidad. Además, aplican técnicas invasivas y, por ende, el desarrollador debe introducir código en la aplicación a probar para incorporar la herramienta que le dará soporte en la prueba. Otro inconveniente que presentan es que no soportan un análisis más enfocado en la interacción del usuario con la aplicación, no permiten hacer un monitoreo de las tareas realizadas por el usuario.

Para solucionar algunos inconvenientes mencionados, ciertos trabajos como los que se proponen en Kronbauer y Santos (2011); Lettner y Holzmann (2012); Kronbauer, Santos y Vieira (2012) plantean pruebas para aplicaciones móviles con el uso de la separación de concerns (AOP); los primeros trabajos no tienen en cuenta el contexto de uso, mientras que el último considera el entorno capturando solo algunos datos de este mediante los sensores del dispositivo móvil.

Otra desventaja que presentan estos trabajos es que son rígidos al no tener en cuenta los requerimientos de usabilidad, ni la variabilidad con respecto al hardware de los dispositivos móviles.

Comparando las herramientas y trabajos descriptos con el propuesto, se puede observar que el nuestro: a) tiene como base de recolección de datos las tareas de la aplicación a probar; b) permite gestionar la variabilidad del hardware mediante la selección de características al momento de generar la prueba; c) captura información contextual utilizando los sensores de los dispositivos móviles; d) es una técnica no intrusiva con respecto al código fuente de la aplicación a probar; e) la recolección de datos es automática lo que la hace transparente para el usuario de la aplicación.

Beneficios y limitaciones

Basado en la solución presentada, el enfoque propuesto brinda beneficios relacionados con la modularidad, captura automática, extensibilidad, transparencia y facilidad de uso.

Modularidad. La implementación del modelo de característica utilizando AOP permite que el código de cada característica esté confinado en su aspecto correspondiente.

Captura automática. La captura automática, directamente desde la aplicación, es una manera de poder recolectar tanto métricas como datos reales del contexto durante la interacción del usuario con la aplicación.

Extensibilidad. El framework da la posibilidad de incorporar la captura de nuevas métricas o datos contextuales mediante la incorporación de nuevas características al modelo de características. Permite, de esta forma, una evolución no costosa de este.

Transparencia. Las captura de métricas y datos del contexto se integran a la aplicación a probar de forma transparente, sin la necesidad de modificaciones en el código fuente de la misma.

Facilidad de uso. Efectivamente, el desarrollador de la pruebas puede modificar las tareas monitoreadas y los datos recolectados de las pruebas sin comprender detalles de la implementación. Además, repitiendo los pasos para generar una prueba se puede construir una familia de pruebas de usabilidad.

La principal limitación del trabajo radica, específicamente, en decisiones de implementación, puesto que el framework FUsaM se restringe a dispositivos móviles cuya plataforma es Android. Sin embargo, la SPL para la generación de pruebas de usabilidad móvil presentada es abstracta y puede ser implementada mediante otras herramientas de desarrollo en otras plataformas móviles.

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr
ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

Trabajo futuros

Uno de los objetivos futuros es desarrollar una herramienta que permita realizar un análisis de los datos recolectados y mostrar la información de manera gráfica, que resulte más amigable para los evaluadores de las pruebas de usabilidad.

Referencias

Balagtas-Fernández, F. y Hussmann, H. (2009). *A Methodology and Framework to Simplify Usability Analysis of Mobile Applications.* 24th IEEE/ACM International Conference on Automated Software Engineering, 520-524.

Batory, D. (2005). *Feature models, grammars, and propositional formulas. Software Product Lines,* 9th Int. Conference, LNCS, SPLC 2005, Rennes, France, Proceedings, vol. 3714, Springer (2005), pp. 7-20.

Clements, P. y Northrop, L. (2001). Software Product Lines: Practices and Patterns. Addison-Wesley Longman Publishing Co.. Boston, USA.

Hornbæk, K. (2006). Current practice in measuring usability: Challenges to usability studies and research. *International Journal of Human-Computer Studies,* 79-102.

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

Ivory, M. y Hearst, M. (2001). The state of the art in automating usability evaluation of user interfaces. *Journal ACM Computing Surveys (CSUR),* 470-516.

Kang, K., Cohen, S., Hess, J., Novak, W. y Peterson, S. (1990). Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University.

Kronbauer, A. H. y Santos, C. (2011). *Um modelo de avaliação da usabilidade baseado na captura automática de dados de interação do usuário em ambientes reais.* Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction. Porto Alegre, Brazil, 114-123.

Kronbauer, A., Santos, C. y 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.* Brazilian Symposium on Human Factors in Computing Systems. Porto Alegre, Brazil, 305-314.

Lettner, F. y Holzmann, C. (2012). Automated and Unsupervised User Interaction Logging as Basis for Usability Evaluation of Mobile Applications. International Conference on Advances in Mobile Computing & Multimedia. New York, USA, 118-127.

Loughran, N., Sampaio, A. y Rashid, A. (2006). From requirements documents to feature models for aspect oriented product line implementation. Satellite Events at the MoDELS Conf., 262-271.

Lyons, K. y Starner, T. (2001). Mobile Capture for Wearable Computer Usability Testing. ISWC '01, 5th IEEE International Symposium on Wearable Computers, 69-76.

Pohl, K., Böckle, G. y Linden, F. (2005). Linden Software Product Line Engineering: Foundations, Principles and Techniques. Springer.

Salazar, J. (2009). *Herramienta para el modelado y configuración de modelos de características*. Malaga, España.

Shaw, M. (2003). Writing Good Software Engineering Research Papers. Minitutorial. Proceedings of the 25th International Conference on Software Engineering. IEEE Computer Society, 726-736.

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.30-2.3

UNICIENCIA Vol. 30, No. 2, pp. 31-45. Julio-diciembre, 2016.
URL: www.revistas.una.ac.cr/uniciencia
Email: revistauniciencia@una.cr

Thüm, T., Kästner, C., Benduhn, F., Meinicke, J., Saake, G. y Leich, T. (2014). FeatureIDE: An Extensible Framework for Feature-Oriented Software Development. Science of Computer Programming, 79(0):70-85. http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide

Zhang, D. y Adipat, B. (2005). Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. *International Journal of Human-Computer Interaction,* 293-308.