Framework para Implementar Pruebas de Usabilidad Flexibles para Aplicaciones Móviles

Enriquez Juan Gabriel y Casas Sandra Isabel

GISP - Instituto de Tecnología Aplicada - Unidad Académica Río Gallegos

Universidad Nacional de la Patagonia Austral
Río Gallegos, Santa Cruz, Argentina.

{jenriquez, scasas}@unpa.edu.ar

Resumen. Al realizar pruebas de usabilidad en aplicaciones móviles, dos de los factores a considerar son el contexto móvil de uso y la variabilidad del hardware de los dispositivos móviles. Este trabajo propone un framework para implementar pruebas de usabilidad que sean flexibles para los desarrolladores al encontrarse con diferente hardware, no intrusivas para la aplicación que se quiere probar y que recolecten, además de las métricas de usabilidad, información relacionada al contexto de uso. Para lograr los objetivos mencionados se plantea combinar la Programación Orientada a Características y la Programación Orientada a Aspectos como técnicas para realizar la recolección de datos de usabilidad (métricas, contexto y satisfacción) de manera no intrusiva y además para permitir administrar la variabilidad del hardware de los dispositivos móviles de un modo flexible.

Palabras Claves: Pruebas de Usabilidad, Programación Orientada a Características, Programación Orientada a Aspectos, Aplicaciones Móviles.

1 Introducción

Para que una aplicación o producto de software pueda ser considerado exitoso, una de las características principales que debería cumplir es que sea usable. Para los desarrolladores de software resulta importante poder medir esa usabilidad o realizar pruebas de usabilidad a las aplicaciones creadas.

Formalmente, la definición más utilizada o reconocida de usabilidad es la que se expone en la norma ISO 9241-11[1], en la cual usabilidad se describe como el grado con el que un producto puede ser usado por usuarios específicos para alcanzar objetivos específicos con efectividad, eficiencia y satisfacción, en un contexto de uso específico [1].

[1] ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) - part 11: Guidance on usability.

Particularmente en las aplicaciones móviles uno de los desafíos que existen al medir usabilidad consiste en identificar los factores adicionales relacionados al ambiente de uso (contexto móvil) que pueden impactar en la usabilidad de este tipo de aplicaciones [2]. Debido a la gran variedad de hardware de los dispositivos móviles, otro de los desafíos consiste en poder administrar de forma flexible la variabilidad para que las pruebas puedan ejecutarse en los diferentes dispositivos. Además es deseable que la implementación de las pruebas de usabilidad no sea intrusiva con respecto al código de la aplicación que se quiere probar, con esto se pretende que las mismas resulten prácticas para los evaluadores.

Para alcanzar los desafíos mencionados, en este artículo se propone integrar técnicas de la Programación Orientada a Características (FOP, Feature-Oriented Programming) [3] y de la Programación Orientada a Aspectos (AOP, Aspect-Oriented Programming) [4] para desarrollar un framework que permita implementar pruebas de usabilidad para aplicaciones móviles desarrolladas en la plataforma Android. La AOP ha sido propuesta como una técnica no intrusiva para incorporar funcionalidad a una aplicación base. Por otro lado la FOP en combinación con la AOP fue presentada como una forma efectiva de gestionar la variabilidad del software [5].

Las ventajas de estas pruebas son: a) no intrusivas para la aplicación que se está probando, b) flexibles para el desarrollador de las pruebas con respecto a las diferentes configuraciones de hardware de los dispositivos, c) transparentes para el usuario y d) además de la métricas clásicas recolectadas en estas prueba se incorpora información del contexto de uso en el momento de la interacción del usuario con la aplicación.

El artículo se organiza de la siguiente forma: La Sección 2 realiza un resumen general del estado actual de las pruebas de usabilidad en aplicaciones móviles. En la Sección 3 se detallan las ventajas de combinar FOP y AOP. En la Sección 4 se presenta el framework propuesto y por último en la Sección 5 se describen las conclusiones y aportes de este trabajo.

2 Pruebas de Usabilidad en Aplicaciones Móviles

En las plataformas móviles a diferencia de otras (Web, escritorio, TV digital), la usabilidad es un problema más significativo, esto es debido a que una gran mayoría de las aplicaciones móviles son difíciles de usar, poco flexibles y no son robustas.

Para medir usabilidad en estas aplicaciones y obtener resultados reales, es necesario considerar el contexto como parte integral de la aplicación. La información contextual que se pueda obtener al momento en que se está usando la aplicación permite realizar un análisis más preciso de usabilidad [6, 7, 8, 9], de lo contrario el análisis resulta sesgado en comparación con lo que sucede cuando un usuario usa la aplicación en el mundo real. Por ejemplo, la conectividad en un entorno real de uso puede ir cambiando según el lugar donde se encuentre el usuario, afectando la usabilidad de la aplicación.

La falta de incorporación de información contextual en las pruebas se debe a que resulta complicada la recolección de datos en los ambientes donde el usuario

interactúa con la aplicación; aplicar técnicas de evaluación en estos entornos reales está lejos de ser trivial [10].

Por otro lado, la carencia de herramientas informáticas específicas que faciliten la realización de estas pruebas y el análisis de los datos recolectados, hace que estas técnicas de evaluación no estén integradas en las actuales prácticas de la Ingeniería de Software. Las herramientas existentes como por ejemplo Flurry Analytics [11], Localytics [12], Mixpanel [13], EVA Helper Framework [14], VizWear [15], solo cubren algunos factores relacionados a la usabilidad de una aplicación móvil. Recolectan información referida al uso de la aplicación y no a como interactúa el usuario con ella, por ejemplo las métricas que proponen son el número de usuarios que accede a la aplicación por día, el promedio de tiempo que se utiliza la aplicación. Otras simplemente hacen un seguimiento de eventos producidos mediante logs, ninguna incorpora información contextual. Además estas herramientas emplean técnicas invasivas en la cual el desarrollador debe introducir código en su aplicación para incorporarla y poder realizar la prueba durante el uso de la aplicación. Otro inconveniente que presentan es que no tienen en cuenta la variabilidad con respecto al hardware de los dispositivos móviles.

Algunos trabajos como los que se proponen en [16], [17], [18] y [19] plantean pruebas específicas para aplicaciones móviles usando la AOP. Los primeros trabajos recolectan métricas cuantitativas pero no consideran el contexto móvil, mientras que el último solo captura ciertos datos contextuales mediante los sensores del dispositivo móvil, ninguno de los trabajos tiene en cuenta la flexibilidad o variabilidad cuando se desea configurar la prueba para diferente hardware.

3 Combinando FOP y AOP

La FOP es una técnica de programación que permite la representación de las similitudes y las variaciones de un sistema de software. El concepto principal de esta técnica es denominado “característica” (feature). Una característica es la unidad funcional en la que se descompone un sistema de software para satisfacer un requisito, representa una decisión de diseño, y proporciona una potencial opción de configuración [3]. El objetivo de la descomposición es construir software bien estructurado que se puede adaptar fácilmente a las necesidades del usuario y a diferentes escenarios de uso. Por lo general, a partir de un conjunto de características, muchos sistemas de software diferentes pueden ser generados compartiendo características comunes y difiriendo en otras [20].

Por otro lado, la AOP es un técnica para el desarrollo de software que proporciona mecanismos y abstracciones para la implementación de crosscutting concerns (preocupaciones transversales), de manera separada y aislada. Las herramientas AOP permiten la descomposición de estos crosscutting concerns en unidades o módulos denominados “aspectos”, tiene la ventaja de permitir agregar funcionalidad a una aplicación base de una manera no intrusiva. Una propiedad deseable de una prueba de usabilidad es que pueda acoplarse o ligarse de forma no invasiva a los componentes de la aplicación base que se quiere testear. En [21] se demostró que la AOP puede ser

utilizada con éxito para realizar pruebas de usabilidad automáticas y de fácil integración con la aplicación a probar.

El potencial de la integración sinérgica entre FOP y AOP ha sido estudiado en varios trabajos de investigación, por ejemplo [22, 23, 24]. Estos trabajos muestran que la integración mencionada mejora la modularidad de las características (FOP) y flexibiliza la composición entre ellas. Sin embargo, no hay un consenso acerca de cómo los conceptos claves de cada enfoque se relacionan entre sí [25].

4 Framework Propuesto.

FOP es una técnica que permite gestionar la variabilidad en una familia de sistemas de software [4]. La variabilidad que definimos en las pruebas de usabilidad está asociada con la selección de la información que se desea recolectar durante la prueba, en particular la información contextual está ligada a la disponibilidad o no de cierto hardware en el dispositivo móvil. En este sentido, se definen tres tipos de variabilidad: *métricas*, *contexto* y *satisfacción*. Las *métricas* están directamente relacionadas a la aplicación móvil que se pruebe, por ejemplos el tiempo que demora en realizarse una tarea. En cambio, la información del *contexto* está vinculada a capturar datos del entorno durante la realización de dicha tarea y depende directamente del hardware del dispositivo en el cual se va a ejecutar la misma. Por lo general esta captura de información se hace a través de sensores incorporados en los dispositivos. Por último *satisfacción* permite desplegar un cuestionario para ser completado por el usuario luego de finalizar una tarea.

Los distintos tipos de datos o información que recolecta una prueba de usabilidad (métricas, contexto, satisfacción) pueden ser representados como características opcionales al momento de implementar la prueba. El desarrollador podría seleccionar que información desea recolectar de la interacción entre el usuario y la aplicación, en base a sus requerimientos y teniendo en cuenta las especificaciones técnicas del dispositivo en el cual se va a ejecutar la misma. Por ejemplo una prueba podría ser configurada para recolectar el tiempo que se requiere para completar una tarea (métrica) y además la luminosidad y temperatura del ambiente (contexto).

El framework se implemento utilizando el entorno de desarrollo Eclipse integrado con FeatureIDE, AspectJ y ADT (Android Development Tool). FeatureIDE [26] es una herramienta que soporta el desarrollo de software orientado a características (FOSD), se empleo para definir el modelo de características y además da soporte a la generación de la familia de pruebas. Las características son implementadas y compuestas en AspectJ, mediante ADT se accede a las librerías de la plataforma Android para darle funcionalidad a las mismas.

4.1 Modelo de Características

El modelo de características definido se puede observar en la Fig. 1, las características opcionales en el modelo representan la funcionalidad relacionada con la información que se puede recolectar de la interacción del usuario con la aplicación móvil. La

característica abstracta **<Usabilidad>** se utiliza para estructurar el modelo y no tienen impacto a nivel de implementación.

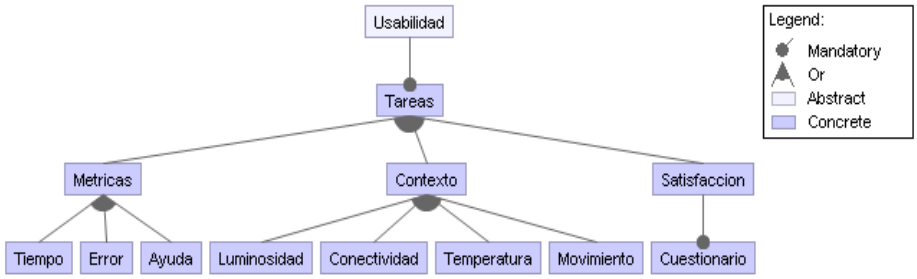

Fig. 1. Modelo de Características para Pruebas de Usabilidad Móvil.

La descripción de la características estan definidas en la Tabla1. **<Tareas>** es una característica de tipo obligatoria (mandatory), si no hay definidas tareas no se puede recolectar información de usabilidad. A partir de la definición de **<Tareas>** se puede configurar la recolección de información de **<Métricas>**, **<Contexto>** o **<Satisfaccion>** (composición de tipo Or). La característica **<Métricas>** está compuesta por las características **<Tiempo>**, **<Error>** y **<Ayuda>** mediante una composición de tipo Or; las características **<Luminosidad>**, **<Conectividad>**, **<Temperatura>** y **<Movimiento>** componen a **<Contexto>** también por una composición de tipo Or. Por último **<Cuestionario>** es una característica obligatoria para componer **<Satisfaccion>**.

Tabla 1. Descripción de las Características.

Nombre	Descripción	Tipo	Padre	Restricción
Usabilidad	Inicial (Abstract)		Ninguno	Ninguna
Tareas	Define el inicio y el fin de las tarea	Mandatory	Usabilidad	Ninguna
Métricas	Intercepta las tareas	Or	Tareas	Ninguna
Contexto	Intercepta las tareas	Or	Tareas	Ninguna
Satisfacción	Intercepta las tareas	Or	Tareas	Ninguna
Tiempo	Registra el tiempo	Or	Métricas	Ninguna
Error	Registra si se produce un error	Or	Métricas	Ninguna
Ayuda	Registra si se utilizo la ayuda	Or	Métricas	Ninguna
Luminosidad	Registra el nivel de luz exterior	Or	Contexto	Requiere sensor de luz
Conectividad	Registra el nivel de conectividad con una red	Or	Contexto	Ninguna

Temperatura	Registra la temperatura del entorno	Or	Contexto	Requiere sensor de temperatura
Movimiento	Registra los movimientos	Or	Contexto	Requiere giróscopo
Cuestionario	Realiza un cuestionario al finalizar una tarea	Mandatory	Satisfacción	Ninguna

4.2 Proceso de Generación de las Pruebas

El proceso de generación de las pruebas que van a ejecutarse en diferentes dispositivos se puede observar en la Fig. 2. A partir de la aplicación base (desarrollada en Android) que se quiere probar, el evaluador debe describir el método inicial y final de cada tarea que desea evaluar. Luego mediante la selección de las características, el generador teje las características seleccionadas representadas como aspectos (AspectJ) sobre la aplicación base, según los métodos iniciales y finales definidos. Por último, a partir del código compilado (tejido), correspondiente a la aplicación base integrada con la prueba, se genera el archivo apk (Android package archive) que es el que finalmente se va a ejecutar en el dispositivo.

El mecanismo de FeatureIDE permite que para una configuración dada (selección de características), se pueda generar la prueba de manera automática. Esto otorga flexibilidad al desarrollador permitiéndole configurar sus pruebas para diferentes dispositivos con solo seleccionar las características que desea disponer en la ejecución de la prueba e iniciar el proceso de generación. Este proceso se puede repetir seleccionando las características deseadas y volviendo a generar otra prueba o también se pueden generar automáticamente todas las posibles pruebas que se derivan de las diferentes composiciones posibles y válidas de las características definidas en el modelo.

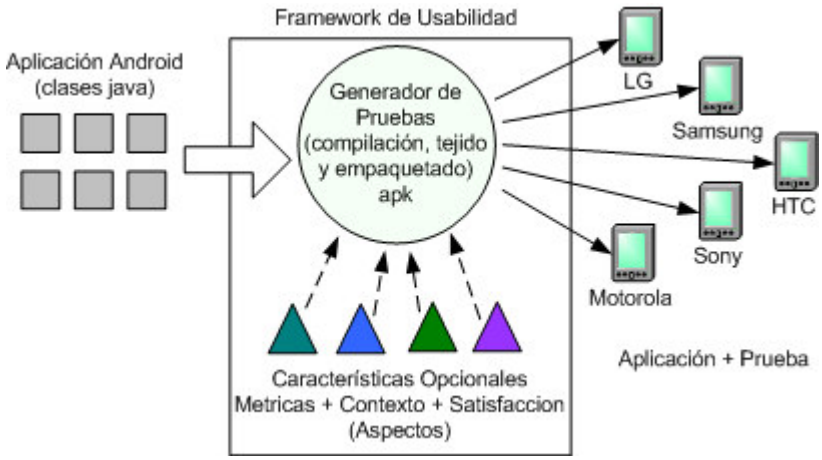

Fig. 2. Proceso de Generación de las Pruebas.

El cálculo del número de pruebas que se pueden derivar del modelo de características revela información acerca de la flexibilidad y complejidad de las pruebas. Mientras mayor sea el número de pruebas mayor será la flexibilidad. Para nuestro modelo de características FeatureIDE calcula que existen 255 configuraciones posibles, es decir, 255 combinaciones de pruebas de usabilidad.

4.3 Implementación mediante AspectJ

La AOP se ha empleado para dar soporte a la implementación de un sistema orientado a características, teniendo en cuenta la funcionalidad transversal y la variabilidad de los sistemas. Permite separar el código relacionado a una característica en un módulo independiente denominado aspecto, facilitando la composición de las mismas implementando lo que se denomina código de unión (glue code), que es el código con el cual se une un componente base a una extensión del sistema [25].

Utilizando AspectJ se implementan los aspectos que van a representar al modelo de características definido anteriormente (Fig. 1), esta tecnología utilizada para Android tiene la restricción que solo soporta tejido en tiempo de compilación.

En la Fig. 3 se puede observar un ejemplo de composición de las características **<Tareas>**, **<Contexto>** y **<Luminosidad>** con aspectos. En este ejemplo se utiliza como aplicación a probar NotePad[2], que es una aplicación desarrollada en Android para gestionar notas. El desarrollador de la prueba primero tiene que indicar en qué métodos de la aplicación a probar, empiezan y terminan las tareas que desea evaluar.

En el aspecto Tarea se definen los puntos de unión entre la prueba de usabilidad y la aplicación a probar. Estos puntos de unión están identificados con los nombres TareaInicio y TareaFin, donde cada tarea de la aplicación que se quiera evaluar debe tener definidos estos puntos. En base al ejemplo en TareaInicio se ha definido como método inicial de la tarea "...NoteList.onListItemClick()" y en TareaFin como método final de la tarea "...NoteEditor.saveNote()".

Estos dos puntos (TareaInicio y TareaFin) declarados son interceptados por el aspecto Contexto en TareaInicio y TareaFin respectivamente. El aspecto que realmente realiza la implementación de la funcionalidad de la característica es Luminosidad que extiende de Contexto. El aspecto Luminosidad es el encargado de utilizar la clase SensorLuz para registrar en un log los valores obtenidos en el inicio y en el final de la tarea. Estos valores obtenidos van a quedar asociado con la tarea realizada, así como también cualquier otra información, asociada con una característica, que fue seleccionada al momento de generar la prueba.

[2] Copyright 2007 The Android Open Source Project.

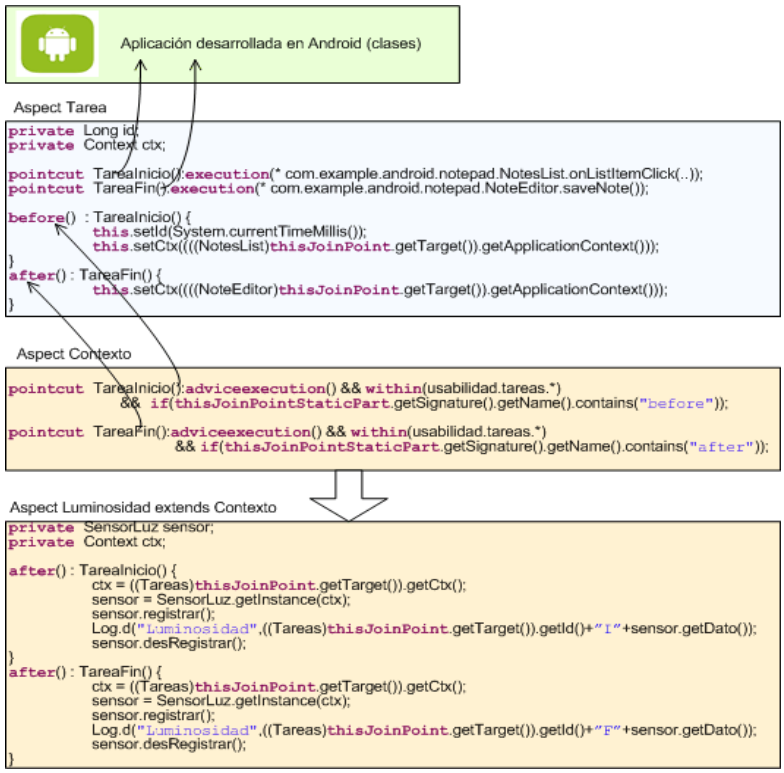

Fig. 3. Implementación con AspectJ de la Característica Luminosidad.

5 Conclusiones

Resulta importante contar con herramientas que faciliten a los desarrolladores la medición de usabilidad de sus aplicaciones móviles. Las herramientas actuales son intrusivas, necesitan que el desarrollador introduzca código en su aplicación para incorporarlas y poder realizar la prueba de usabilidad. El framework propuesto es no intrusivo y recolecta la información de usabilidad de forma automática en el ambiente real de uso, de esta manera facilita la implementación de dichas pruebas sobre una aplicación móvil.

Debido a la gran variedad de dispositivos móviles resulta complejo realizar distintas pruebas para diferentes dispositivos Este framework permite generar una familia de pruebas de manera automática a partir de la selección de distintas características predeterminadas, esto le brinda flexibilidad al desarrollador al momento de generar e incorporar las pruebas en la aplicación que se está probando.

Por otro lado el framework se puede extender para que recolecte nueva información de usabilidad, mediante la creación de nuevos aspectos que implementen

la recolección de la información deseada. Para incorporar esta nueva funcionalidad (aspecto) se debe seguir el esquema definido por el framework.

Como trabajo futuro resta realizar nuevos experimentos con aplicaciones más complejas, usuarios con diferentes perfiles y diferentes dispositivos móviles con el fin de validar el framework propuesto.

Referencias

1. ISO International Standard, ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs.) - Part 11: Guidance on usability (1998).
2. Zhang, D., Adipal, B.: Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. International Journal of Human-Computer Interaction, vol. 18, pp. 293–308 (2005).
3. Kang, K., Cohen, S., Hess, J., Novak, W. and Peterson, S.: Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990).
4. Kiczales, G.: Aspect-Oriented Programming. In Proceedings of European Conference on Object Oriented Programming (ECOOP) (1997).
5. Mezini, M. and Ostermann, K.: Variability Management with Feature Oriented Programming and Aspects. SIGSOFT, International Symposium on the Foundations of Software Engineering FSE (2004).
6. Hummel, K. A., Hess, A. and Grill, T.: Environmental Context Sensing for Usability Evaluation in Mobile HCI by Means of Small Wireless Sensor Networks. International Conference on Advances in Mobile Computing and Multimedia, pp. 302–306 (2008).
7. Coursaris, C. K. and Kim, D. J.: A Meta-Analytical Review of Empirical Mobile Usability Studies. Journal of Usability Studies, vol. 6, no. 3, pp. 117−171 (2011).
8. Maguire, M.: Context of Use within usability activities. International Journal of Human Computer Studies, vol. 55, no. 4, pp. 453−483 (2001).
9. Munesh C. T. and Mohammadi A. K.: Role of Context in Usability Evaluations: A review. Advanced Computing: An International Journal (ACIJ), vol. 3, no. 2, pp. 69−78 (2012).
10. Kjeldskov, J. and Skov, M. B.: Creating Realistic Laboratory Settings: Comparative Studies of Three Think-Aloud Usability Evaluations of a Mobile System. 9th IFIP TC13 International Conference on Human-Computer Interaction, pp. 663–670 (2003).
11. http://www.flurry.com.
12. http://www.localytics.com.
13. http://www.mixpanel.com.
14. Balagtas-Fernandez, F. and Hussmann, H.: A Methodology and Framework to Simplify Usability Analysis of Mobile Applications. 24th IEEE/ACM International Conference on Automated Software Engineering, pp. 520 (2009).
15. Lyons, K. and Starner, T.: Mobile Capture for Wearable Computer Usability Testing. ISWC '01, 5th IEEE International Symposium on Wearable Computers, pp. 69–76 (2001).
16. Tarta, A.M. and Moldovan, G.S.: Automatic Usability Evaluation Using AOP. 2006 IEEE International Conference on Automation, Quality and Testing, Robotics, vol. 2, pp. 84−89 (2006).
17. Kronbauer, A. H. and Santos, C.: 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, pp. 114–123. Porto Alegre, Brazil (2011).

18. Lettner, F. and Holzmann, C.: Automated and Unsupervised User Interaction Logging as Basis for Usability Evaluation of Mobile Applications. International Conference on Advances in Mobile Computing & Multimedia, pp. 118–127. New York, USA (2012).
19. Kronbauer, A., Santos, C. and Vieira, V.: 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, pp. 305–314. Porto Alegre, Brasil (2012).
20. Sven Apel Christian Kästner: An Overview of Feature-Oriented Software Development. Journal de object technology vol 8 no 5 (2009).
21. Tarja, A.M., Moldovan, G.S.: Automatic Usability Evaluation Using AOP. IEEE International Conference on Automation, Quality and Testing, Robotics, vol. 2, pp. 84–89 (2006).
22. U. Kulesza, V. Alves, A. Garcia, A. C. Neto, E. Cirilo, C. J. P. de Lucena, and P. Borba: Mapping features to aspects: A model-based generative approach. Early Aspects: Current Challenges and Future Directions, 4765/2007:155–174 (2007).
23. K. Lee, K. C. Kang, M. Kim, and S. Park: Combining feature-oriented analysis and aspect-oriented programming for product line asset development. IEEE Computer Society, In SPLC '06: Proc. of the 10th Intl. on Software Product Line Conf., pages 103–112, Washington, DC, USA (2006).
24. N. Loughran, A. Sampaio, and A. Rashid: From requirements documents to feature models for aspect oriented product line implementation. Satellite Events at the MoDELS Conf., Volume 3844/2006:262–271 (2006).
25. Tizzei, P. L., Lee, J. and Rubira, C.M.: An Aspect-oriented View to Support Feature-oriented Reengineering. Proceedings of the 15th International Workshop on Aspect-Oriented Modeling (2010).
26. KÄastner, C., ThÄum, T., Saake, G., Feigenspan, J., Leich, T., Wielgorz, F. and Apel, S.: FeatureIDE: A tool framework for feature-oriented software development. IEEE, In ICSE, pp. 611–614 (2009).