Convertir aplicaciones de escritorio en aplicaciones móviles con Java

Beatriz Alexandra Arbeláez H.

Recibido el 30 de noviembre de 2009. Aprobado el 15 de marzo de 2010

Resumen
El presente artículo introduce un proceso básico para convertir aplicaciones J2SE en aplicaciones JavaFX, centrando su atención en programas locales que manejen persistencia sobre archivos texto y que deseen ser migrados a teléfonos inteligentes. Igualmente, entrega pautas para generar desde sus inicios en pasos ordenados una aplicación JavaFX sin tener que contar con la experiencia previa del programa en J2SE o en ambientes de escritorio. Para esta nueva modalidad de programación donde se pretende como mínimo la misma calidad, presentación y funcionalidad de las aplicaciones de escritorio pero con el aprovechamiento máximo de los recursos de un dispositivo móvil, se presentó a un grupo de estudiantes en un curso de programación avanzado de Tecnología en Informática de la Corporación Universitaria Minuto de Dios (UNIMINUTO) una aplicación ya diseñada e implementada en J2SE para ser convertida a JavaFX. Al finalizar el trabajo se conservó la integridad y lógica del desarrollo original a su nueva versión sin necesidad de tener el conocimiento y la agilidad en ambos lenguajes de programación.

Palabras clave
Java, Aplicaciones Móviles, Aplicaciones de Escritorio, POO, RIA, J2SE, J2ME, JavaFX, Patrones, UML

Abstract
This paper introduces a basic process for converting JavaFX applications to J2SE applications, over local programs that handle text files and persistence to be migrated to smart phones, also gives patterns to generate from its beginnings with orderly steps a JavaFX application, without to have prior experience in J2SE program or desktop environments. For this new type of programming that seeks the same quality, presentation and functionality of desktop applications with the maximum utilization of the resources of a mobile device, was presented to a group of students in an advanced programming course Computer Technology in the university, an application already designed and implemented in J2SE to be converted to JavaFX. At the end of job was preserved the integrity and logic of the original development in the new version without having the knowledge and agility in both programming languages.

Keywords
Java, Mobile Applications, Desktops Applications, OOP, J2SE, J2ME, JavaFX, Patterns, UML

I. Introducción

Después de programar en la plataforma J2SE uno de los caminos a seguir es dejar los ambientes para computadores de escritorio y portátiles e ingresar a los ambientes de dispositivos móviles. Al día de hoy, las aplicaciones deben adaptarse a los diferentes dispositivos que se encuentran en el mercado, con mayor razón si estos dispositivos tienen una alta difusión en personas cuya relación con la tecnología es mínima o en algunos casos nula. Esta ventaja debe aprovecharse con rapidez y eficiencia para reescribir de forma ágil y correcta aplicaciones de éxito desde entornos de escritorio o portátiles a entornos de dispositivos móviles. JavaFX es uno de los últimos lenguajes lanzados por Sun Microsystems (Javafx 2009) que une sus anteriores productos J2ME, Java 2D y fortalece las RIA (Rich Internet Aplications) para obtener beneficios en diseño con efectos visuales dentro de la lógica requerida (Javafx.com, 2009). Aquí Sun vuelve a apostar por su premisa 'write once – run everywhere', tema de artículos (Portal.acm, 2009) donde se presentan la conversión de aplicaciones J2SE a aplicaciones J2ME. El lenguaje J2ME (Developers 2009) difiere del lenguaje JavaFX por el paradigma que ambos utilizan, los dos pueden implementar el mismo, orientado a objetos pero JavaFX tiene la flexibilidad de incluir la implementación de los paradigmas para los lenguajes script (Frt.utn, 2009).

Las necesidades de conversión de una plataforma a otra, en este caso de escritorio a móvil, pueden resumirse (Burigat.Chittaro, 2007) desde la presentación de datos y navegación entre ellos; por ejemplo en móviles, los usuarios son forzados a realizar tareas de acercamiento-alejamiento para obtener una visión global de la información con altas posibilidades de desorientación y aumento de complejidad, pasando por la velocidad en el cálculo de instrucciones, de este modo, en escritorio, se tienen equipos con alta capacidad de procesamiento y almacenamiento para representar gráficos 3D o mapas detallados de posición geográfica, hasta llegar a periféricos específicos de trabajo. Esto influye en el diseño de aplicaciones, los paradigmas de programación deben adaptarse a las necesidades y características de su entorno, por ejemplo, la generación de menús en teléfonos celulares puede realizarse entre los modelos (Amant.Horton.Ritter, 2007) Fitts' law, GOMS o ACT-R que permiten un patrón de calidad para medir comportamiento, niveles de detalle y análisis en los usuarios.

Este artículo presenta una guía para convertir aplicaciones de escritorio en aplicaciones móviles con Java, mediante pasos ordenados, así como un listado de apuntes sobre buenas prácticas al momento de la implementación sobre JavaFX. Así, se describe al inicio los antecedentes, que actualmente se encuentran sobre el tema, para luego explicitar la metodología del proceso de conversión y exponer finalmente los resultados.

II. Antecedentes

Las aplicaciones móviles deben estar en capacidad de manejar características como trabajo desconectado y utilización adecuada de los recursos, en arquitecturas de componentes que puedan correr sobre diferentes capas (Nori, 2007). Se busca entonces, que su desarrollo tenga en cuenta: baja carga de archivos en compilación, eficiencia en invocaciones, baja redundancia de datos, acceso directo a la información, incluyendo programación por componentes, interfaces gráficas simples y ágiles, recargas puntuales de datos, código exacto. Estas características a hoy se evidencian, siendo reconocidas como requerimientos no funcionales sobre plataformas móviles, por lo tanto al ir de una plataforma de escritorio, la nueva aplicación móvil, debe reflejarlas en el proceso de conversión. Aparecen también, los lenguajes scripts que permiten naturalidad y flexibilidad en la programación, resultando una transición del paradigma de programación orientado a objetos. Sin embargo, los trabajos de investigación consultados hacen referencia (Turunen, 2005) a cambios en aplicaciones ya realizadas, que se modifican para aumentar su alcance en portabilidad u otros requerimientos adicionales, por ejemplo, una aplicación móvil inicial que aplica posteriormente otra arquitectura para manejar diálogos multimodales distribuidos.

Una de las principales razones para tomar una aplicación de escritorio ya existente y convertirla a una aplicación móvil, es la rapidez con la cual debe liberarse el código para evitar que el tiempo de vida útil de los dispositivos móviles las convierta en obsoletas y sean necesarias adecuaciones sin ser liberadas aún (Balan, 2007). Adicionalmente, se comparte en este artículo la posición de cambio parcial, del artículo anteriormente mencionado; no es necesario modificar todo el código original para obtener el código sobre el dispositivo móvil, pero no se comparte la idea de una plataforma neutral de desarrollo. Aquí se presenta una serie de pasos ordenados para convertir el desarrollo inicial, como forma manual y aproximación inicial de procesos más complejos, en los cuales si debiese automatizarse el resultado final, conclusiones que entrega el artículo consultado. En la metodología, resultados y conclusiones, del presente artículo, se entregarán preguntas puntuales a responder que indican las características a encontrarse en la aplicación de escritorio para obtener su equivalencia en la aplica-

ción móvil, así como la recurrencia a la experiencia del programador en el desarrollo e implementación de patrones, permitiendo los cambios apropiados, sin perder los objetivos primarios del proceso, fidelidad en los resultados, en el ¿qué de aplicación?, entregando la libertad en "el ¿cómo hace la aplicación?" que depende en un alto porcentaje de las características de la plataforma en que se ejecute. Como se mencionaba al inicio, dichas características forman una política de ejecución, asegurando que si se cumple permita la ejecución y conversión de la aplicación, también se conoce con el nombre de política de adaptación (Balan, 2007), que son definidas, adecuadas e implementadas.

III. Metodología

Se tomó como aplicación ejemplo, un juego llamado "El Aprendiz", desarrollado en J2SE versión 1.6 update 14 (Java.sun, 2009) para definir y concluir los cambios que una aplicación en esa plataforma contemplaría para acercarse a la promesa de Sun. El Aprendiz es un juego cuyo objetivo es completar satisfactoriamente, ejercitando la memoria, una ruta aleatoria de luces. Se puede seleccionar el nivel de dificultad que depende del número de luces que componen la ruta y del tiempo de visualización de cada luz ante el usuario para recordarlas. El usuario pierde al momento en que la ruta de luces ingresada no es correcta, el usuario puede reiniciar el juego cada vez que lo desee. El juego interactúa con el jugador y almacena al finalizar, el nivel, el tiempo invertido y el nombre del ganador para mantener una tabla de posiciones con los 5 mejores jugadores en tiempo. La aplicación tiene la presentación gráfica de las figuras 1 a la 3, esta visualización no cambió entre un lenguaje y otro:

Figura 1. *Menú principal de la aplicación. Figura generada en NetBeans 6.5. Fuente: el autor*

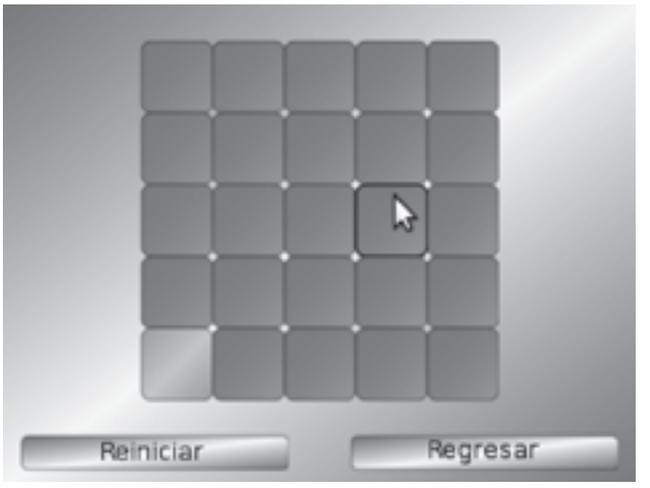

Figura 2. *Tablero de la aplicación, al seleccionar la opción Comenzar. Figura generada en NetBeans 6.5. Fuente: el autor.*

Figura 3. Configuración de la aplicación, al seleccionar la opción Niveles. Figura generada en NetBeans 6.5. Fuente: el autor.

La aplicación en J2SE tuvo el diseño orientado a objetos presente en la figura 4:

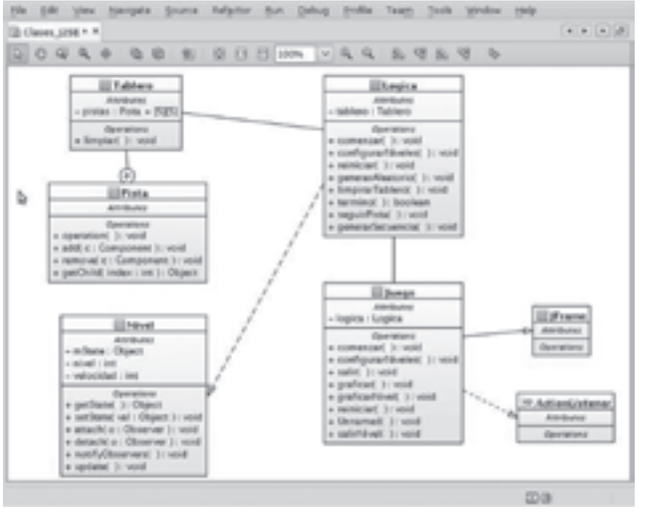

Figura 4. *Diagrama de clases para la aplicación de escritorio. Fuente: el autor.*

Al programar en J2SE se aplicaron las buenas prácticas del diseño orientado a objetos (Faculty, 2009) presentes en los patrones de software (quienes capturan las mejores soluciones para problemas que se presentan repetidamente) manteniendo la reutilización, extensibilidad, bajo acoplamiento y alta cohesión (Web.cs, 2009). Algunos de los más representativos son los GRASP, patrones de asignación de responsabilidades (General Responsibility Assignment Software Patterns), entre ellos, patrón creador, patrón controlador, patrón experto y patrón fachada (Cs.wcupa, 2009). JavaFX actualmente tiene libertad de implementar las soluciones desde la óptica orientada a objetos o desde la óptica script, la pregunta que definió el camino fue de la mano de la principal fortaleza que promete JavaFX, el diseño gráfico, si la aplicación tiene un alto número de pantallas y una navegación compleja la respuesta es aplicar diseño orientado a objetos, si por el contrario la aplicación tiene un alto número de animaciones pero bajo número de pantallas y navegación de complejidad simple puede optar por el modelo script. Sin embargo, en este punto aunque la respuesta parecía obvia lo más recomendado fue considerar una mezcla equilibrada de los dos paradigmas en el diseño de la aplicación JavaFX ya sea nueva o en migración de una aplicación J2SE. La aplicación El Aprendiz tomó esta decisión.

La siguiente pregunta consideró la permanencia del mismo diseño de clases y relaciones de la aplicación J2SE en la nueva aplicación JavaFX?, Para este caso, la respuesta fue no. Los patrones de software que se aplicaron en el diseño se centraron en solucionar problemas que llegaron de los requerimientos funcionales y no funcionales, pero para JavaFX la lista de requerimientos funcionales se ve opacada por la lista de requerimientos no funcionales donde la aplicación se torna más exigente, específica y mínima. Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, desempeño, disponibilidad, escalabilidad, usabilidad, flexibilidad, instalación, mantenibilidad, operatividad, seguridad, integración, conectividad.

Las aplicaciones JavaFX sobre dispositivos móviles se diseñan teniendo como prioridad los requerimientos no funcionales de desempeño, usabilidad, persistencia y conectividad, aunque existen patrones que hacen referencia a estos requerimientos, se debe concentrar la atención en las condiciones de memoria y espacio, particionando las aplicaciones en clases que permitan intercambiar su manejo en memoria, implementando un diseño separado de lógica, disminuir el nombre de los atributos, métodos y clases para disminuir el tamaño de la aplicación binaria.

IV. Resultados

Adicional a las condiciones de memoria y espacio presentes en los dispositivos móviles, fue necesario poner especial cuidado en el diseño gráfico, en la reproducción de sonido, video, manejo de tamaños y navegación, convirtiendo esta preocupación en otro punto a favor de la respuesta negativa a la pregunta anterior. Hay una diferencia entre los APIs de las dos plataformas que componen la parte gráfica, por ejemplo, JavaFX (Java, 2009) no considera necesarias las clases de Security Manager, RMI, Corba, las clases que manejan la interfaz gráfica varían desde la pareja SWING-AWT en J2SE hasta la línea principal de visualización de JavaFX formada por las clases Stage, Scene y Content que se visualizan en la figura 5.

Figura 5. Stage, Scene y Content en JavaFX. Figura generada en NetBeans 6.5. Fuente: el autor.

El borde de la gráfica representa la ventana (Stage), dentro se encuentra el espacio de trabajo (Scene) donde se ubican en forma jerárquica los objetos gráficos. Los objetos gráficos propios de una aplicación y diferentes a los presentes en el API gráfico de JavaFX son creados por herencia de la clase CustomNode que permite personalizar e incluir las características y comportamientos propios exigidos. Así mismo, la aplicación en J2SE puede trabajar con un tamaño fijo sin importar las dimensiones de la pantalla donde se ejecute, situación que no es cómoda de adoptar en los dispositivos móviles, las aplicaciones deben adecuar su visualización a la variedad de tamaños y funcionalidades presentes, este cambio entre aplicaciones se evidencia más en la implementación que en la etapa de diseño.

Así pues, las aplicaciones realizadas sobre J2SE no tienen que preocuparse en su gran mayoría por el espacio de procesamiento, la velocidad de lectura o el manejo de recursos pues se presentan como ilimitados por toda la facilidad y velocidad con que aumentan dentro de la infraestructura de los computadores de escritorio y portátiles que se ofrecen de fácil actualización. Otro requerimiento no funcional es la conectividad, los computadores de escritorios cuentan siempre con acceso a Internet, pero los dispositivos móviles como los teléfonos celulares, por su naturaleza, están en constante búsqueda de conexión. El juego no utilizó dichas características por lo que el artículo sólo lo menciona por su relevancia.

La persistencia sobre base de datos es algo común en las aplicaciones J2SE de la cual se conocen y se han implementado patrones y soluciones para mejorar desempeño. En los dispositivos móviles el espacio es un recurso crítico y el uso de base de datos puede implementarse en otros tipos de aplicaciones para dispositivos móviles como son cliente-servidor o consumo de servicios WEB que no son el alcance de este artículo. En cuanto al desarrollo local que convirtió El Aprendiz de lenguaje J2SE a lenguaje JavaFX usando archivos textos como persistencia, que debieron convertirse en recursos de almacenamiento (espacios de memoria identificados con un nombre cualquiera) permitiendo el manejo del espacio principal y secundario en los dispositivos móviles.

El empleo de las aplicaciones en J2SE está representado en el manejo del ratón y el teclado como periféricos de entrada, sin embargo, en las aplicaciones JavaFX (Javapassion, 2009) sobre dispositivos móviles el ingreso de información se realiza con las teclas de navegación arriba, abajo, izquierda, derecha, confirmar, aceptar y rechazar estas pueden simularse desde una aplicación J2SE con las flechas de posicionamiento presentes en el teclado de los computadores de escritorio y portátiles, esto garantiza que una pantalla táctil recibirá sin mayores contratiempos una aplicación realizada pensando en un dispositivo móvil de pantalla estándar.

Las preguntas y puntos mencionados anteriormente se aplicaron para pasar de J2SE a JavaFX, siguiéndolos esta guía presenta el diagrama de clases del ejemplo El Aprendiz para JavaFX en la figura 6.

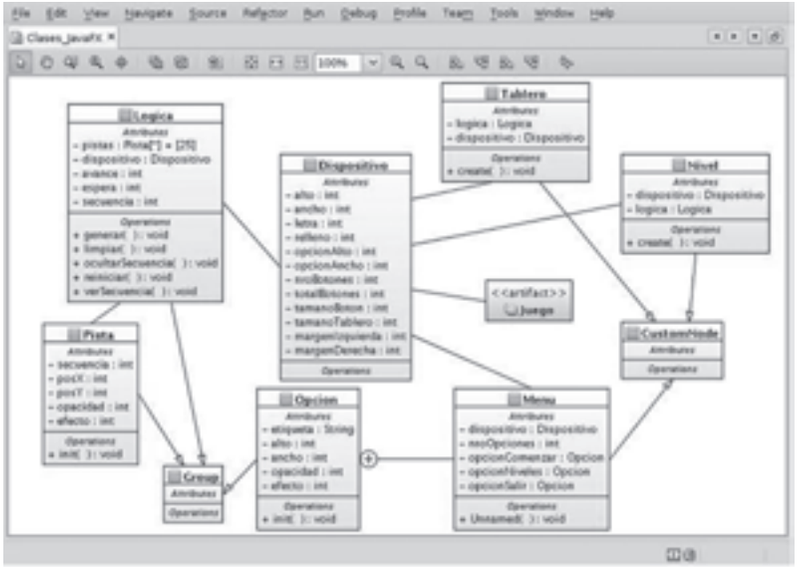

Figura 6. *Diagrama de clases para la aplicación móvil. Fuente: el autor.*

En la implementación y las pruebas, JavaFX disminuyó los tiempos, aplicando las ventajas de los lenguajes script programando con instrucciones básicas, fáciles de entender, naturales, en el juego durante la implementación en J2SE se invirtieron 24 horas y para JavaFX 16 horas aproximadamente.

V. Conclusiones

- Las ventanas emergentes que presentan datos de confirmación deben cambiarse a visualizaciones de escenas diferentes dentro del escenario que simulan el comportamiento de las anteriores.

- Los dispositivos móviles utilizan una máquina virtual KVM llamada así por la referencia a pocos kilobytes de tamaño, demostrando que JavaFX se enfoca en ciertos APIs específicos porque no requiere de los otros, por lo tanto una buena práctica de programación es importar sólo las clases exactas dentro de los paquetes en la aplicación para evitar cargas de código que no sean necesarias en la ejecución.

- Una ventaja que tiene Java en sus plataformas es el Garbage Collector que permite concentrar esfuerzos en la lógica, no en la administración de los recursos permitiendo liberar en forma desatendida el espacio en memoria.

- En la etapa de pruebas las aplicaciones JavaFX necesitan de un emulador genérico de dispositivo móvil o del propio del teléfono para realizar sus pruebas, mientras que las aplicaciones de escritorio no. La flexibilidad de las pruebas llega en seleccionar el dispositivo móvil más acorde a las características de la aplicación, por ejemplo que tenga pantalla táctil o que posea teclado QWERTY.

- Sin importar el diseño implementado en la aplicación J2SE, si se utilizó MVC o se presentan patrones de diseño para manejo de recursos, es posible pasar la aplicación a JavaFX conservando la integridad y lógica independiente del análisis, diseño e implementación original.

VI. Referencias

[1] Álvarez, I. (2006), "Lenguajes y Paradigmas de Programación", Cali, s.e., disponible en: http://www.dis.eafit.edu.co/ depto/ colegios/actas/Leng_Progr.ppt, recuperado el 15 de Septiembre de 2009.

[2] Amant, R, St.; Horton, T. & Ritter, F. (2007, May), Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Inte-*

raction, Vol. 14 No. 1, disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi= 10.1.1.64.842, recuperado el 11 de Mayo de 2010
[3] Balan, R.; Sousa, J. & Satyanarayanan, M. (2003), "Meeting the Software Engineering Challenges of Adaptive Mobile Applications", disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.728, recuperado el 15 de Mayo de 2010.
[4] Burigat, Stefano & Chittaro, Luca (2007), Geographic Data Visualization on Mobile Devices for User's Navigation and Decision Support Activities (2007), disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.104.9491, recuperado el 13 de Mayo de 2010.
[5] Java (s.f.), disponible en: http://java.sun.com/javafx/1.2/docs/api/index.html, recuperado el 9 de Septiembre de 2009
[6] Java (s.f.), "Java™ Platform, Standard Edition 6 API Specification", Java, disponible en: http://java.sun.com/javase/6/docs/api/, recuperado el 9 de Septiembre de 2009
[7] Java (s.f), "Getting Started With JavaFX Technology", Java, disponible en: http://javafx.com/docs/gettingstarted/javafx/index.jsp, recuperado el 10 de Septiembre de 2009
[8] Java (s.f.), "Develop Expressive Content with the JavaFX platform", Java, disponible en: http://javafx.com/about/overview/, Recuperado el 10 de Septiembre de 2009
[9] Javapassion.Com (s.f.), disponible en: http://www.javapassion.com/javafx/, recuperado el 11 de Septiembre de 2009
[10] Universidad Tecnológica Nacional (s.f.), disponible en: http://www.frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm, recuperado el 15 de Septiembre de 2009
[11] Jiang, Z. (s.f.), "GRASP Pattern", West Chester University, disponible en: http://www.cs.wcupa.edu/~zjiang/GRASP_pattern.ppt, recuperado el 16 de Septiembre de 2009.
[12] Levitt, D. (s.f.), Introduction to OO Design-GRASP PatternsInver Hills, Community College disponible en: http://faculty.inverhills.edu/dlevitt/CS%202000%20(FP)/GRASP%20Patterns.pdf, recuperado el 16 de Septiembre de 2009.
[13] Muchow, j. (2002), *Core J2ME Technology and MIDP*, San Antonio, California, Sun Mycrosystems Press A Prentice Hall Title, disponible en: http://developers.sun.com/mobility/midp/chapters/muchowcore/ch1.pdf, recuperado el 10 de Septiembre de 2009
[14] Nogueira, T. et al. (2008), "An adaptation of the collections framework, reflection and object cloning from J2SE to J2ME", 2008 ACM Symposium on Applied Computing, Fortaleza, Brazil, disponible en: http://portal.acm.org/citation.cfm?id=1363686.1363748, recuperado el 9 de Septiembre de 2009
[15] Nori, Anil K. (2007), Mobile and embedded databases, SIGMOD 2007, Proceedings of the 2007 ACM SIGMOD international conference on Management of data, Beijing, China, disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.105.9596, recuperado el 11 de Mayo de 2010
[16] Turunen, Markku et al. (2005), Mobile Architecture for Distributed Multimodal Dialogues. Proc. ASIDE., Universiti of Tampere, Finland, disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.1081, Recuperado el 13 de Mayo de 2010
[17] Worcester Polytechnic Institute (s.f.), disponible en: http://web.cs.wpi.edu/~gpollice/cs4233-a05/CourseNotes/maps/class4/GRASPpatterns.html, recuperado el 16 de Septiembre de 2009.

Beatriz Alexandra Arbeláez H. *Ingeniera de Sistemas de la Universidad Autónoma de Manizales, 1999. Master en Ingeniería de Sistemas y Computación de la Universidad de los Andes (UNIANDES), 2001. En la actualidad es docente tiempo completo del programa de Tecnología en Informática, Corporación Universitaria Minuto de Dios (UNIMINUTO). barbelaez@uniminuto.edu*