ARTICULO DE INVESTIGACIÓN

PROYECTO DE AUTOESTUDIO

ESTRUCTURAS DE DATOS

GABRIEL PASTOR SANTIAGO PRIETO CHACÓN

OMAR SIMÓN FRANCISCO PRIETO CHACÓN

CARLOS EDUARDO TEJADA GUEVARA

BOGOTÁ D.C.

POLITÉCNICO GRANCOLOMBIANO

27 DE NOVIEMBRE DE 2009

UML (UNIFIED MODELING LANGUAGE)

El UML es un lenguaje diseñado para ilustrar los “planos” de un sistema o software, en los cuales se incluyen aspectos conceptuales los esquemas de bases de datos y los componentes reutilizables del sistema modelado. El UML está también respaldado por el OMG u Object Management Group.

Vale la pena resaltar que el UML es solo un lenguaje de modelado, es decir, no especifica en sí mismo qué metodología usar para la construcción del sistema representado, solo lo define. Por esto mismo, el UML no puede compararse con la programación estructurada, ya que este ni siquiera es programación, en este solo se diagrama la realidad de una utilización en un requerimiento.

El UML tiene cuenta con varios tipos de diagramas, los cuales muestran los diferentes aspectos de los entes representados; a su vez, cada uno de estos tipos, puede ser clasificado en uno de los tres grupos mayores

- Diagramas de estructura

  Donde se muestran los componentes que deben existir en el sistema modelado

  - Diagrama de clases

    Tipo de diagrama estático que describe la estructura de un sistema. Muestra las clases, sus atributos y las relaciones entre ellas; son comúnmente utilizados para el proceso diseño, análisis, y conceptualización de la información que se manejará en un sistema.

  - Diagrama de componentes

    Tipo de diagrama que muestra los componentes de un sistema, esto incluye las bibliotecas, archivos, paquetes y cabeceras entre otras, este representa como será dividido un sistema.

- Diagrama de objetos

  Este tipo de diagrama puede ser considerado como un sub diagrama de clases, ya que muestran a las clases instanciadas en un momento específico para un momento particular del sistema. Se utiliza para el proceso de diseño y análisis en el desarrollo de un sistema.

- Diagrama de estructura compuesta

  Este tipo de diagrama muestra la estructura interna de una clase en un sistema, esto incluye las partes internas y puertas mediante las cuales las otras clases interactuarán con ellas entre otras.

- Diagrama de despliegue

  Este diagrama muestra el hardware utilizado en las implementaciones de un sistema y sus componentes.

- Diagrama de paquetes

  Los diagramas de paquetes muestran como un sistema está jerarquizado en agrupaciones lógicas.

- Diagramas de comportamiento

  Donde se muestra lo que debe suceder en el sistema modelado:

  - Diagrama de actividades

    Representa los flujos de trabajo paso a paso del corrido de un sistema.

- Diagrama de casos de uso

  Este tipo de diagrama representa gráficamente casos de uso.

- Diagrama de estados

  En el UML se especifican y estandarizan las formas de representar los estados de un sistema y sus clases a través de círculos, flechas y otras formas geométricas.

- Diagramas de interacción

Donde se muestra el flujo de control y de datos entre los componentes del sistema

- Diagrama de secuencia

  Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo.

- Diagrama de comunicación

  Muestran tanto la estructura estática de un sistema como la interacción entre las diversas componentes.

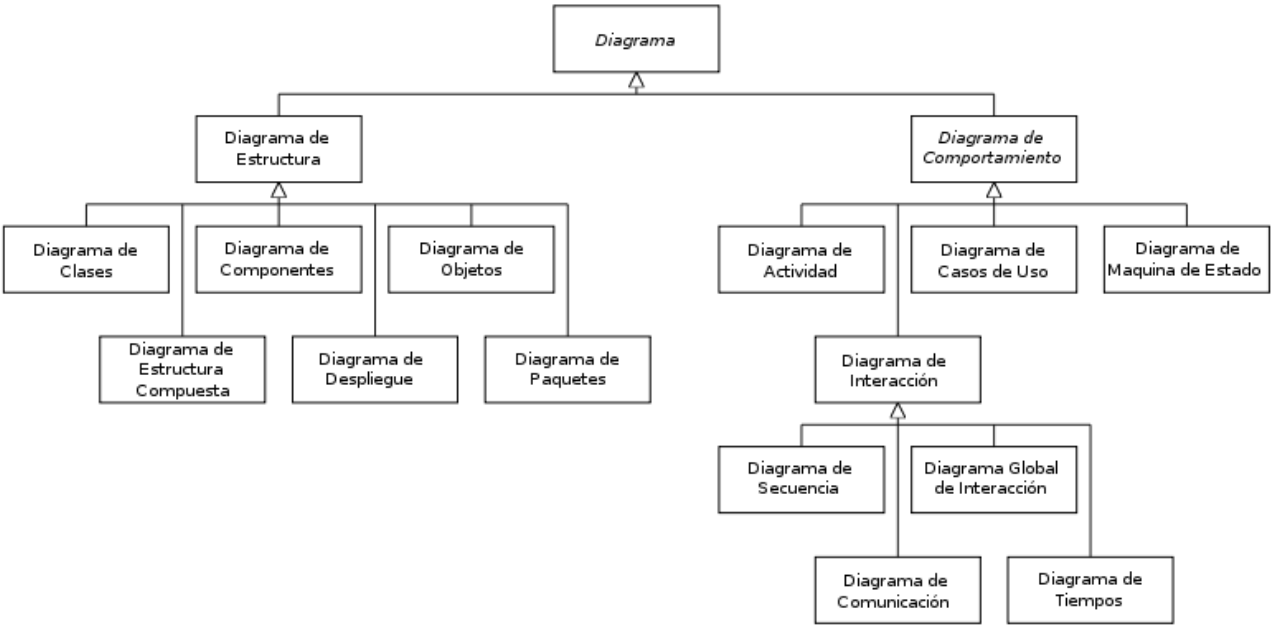
- Diagrama de tiempos

  Se enfoca en el tiempo que toman los mensajes enviados entre objetos.

- Diagrama de vista de interacciones

  Muestra una vista generalizada sobre los aspectos dinámicos de un sistema modelado.

En la siguiente gráfica (*) se muestra la jerarquía de todas las clases de diagramas UML explicadas anteriormente



MVC (MODEL VIEW CONTROLLER)

El patrón MVC es una forma para diseñar nuestros programas dándoles cierta estructura lógica a estos. Su objetivo primordial es separar lo gráfico de lo lógico.

Todas nuestras aplicaciones se construyen de tal forma que existen varias capas lógicas, cada una de estas capas está constituida por una o más clases que colaboran entre sí para solucionar una tarea o algún trabajo.

“Solo para recordar; en una aplicación con una arquitectura de capas, cada capa se encarga de una tarea determinada y una capa solo puede utilizar la capa inferior a ella y ni siquiera conocerá las capas superiores.”

El patrón de diseños MVC tiene como objetivo organizar el flujo de datos de una mejor manera, permitiendo manejar y crear sistemas y aplicaciones más robustos, rápidos y fáciles de mantener; MVC proviene de Model View Controller (Modelo Vista Controlador) y trabaja con tres componentes fundamentales que están muy relacionados entre sí: al momento de construir una aplicación por medio del patrón MVC en la parte de **vista** se ingresarán todas las partes de la aplicación que tengan que ver con la interfaz gráfica, entre otras cosas; en la parte de **modelo** estará toda la parte de la aplicación que trabaja nuestro programa. Por último está el **controlador**, que recibe los eventos de la interfaz del programa.

Algunas veces, en la implementación de este patrón, cuando ocurre un cambio en el diseño del mismo, este envía una notificación a la vista para que se actualice.

JUNIT

El JUnit es un simple framework usado para crear pruebas repetibles en el lenguaje de programación Java, esta herramienta es una instancia de la arquitectura XUnit para la prueba unitaria de las apliciones de ventana, originadas con SUnit.

Después de aparecer en el lenguaje Java, JUnit ha sido movida a otros lenguajes como C#, Ada, PHP, Python, C++ y JavaScript. Esta aparece como un JAR al tiempo de compilación y aparece bajo los paquetes JUnit.framework para los JUnit de la versión 3.8 o anteriores y bajo los paquetes org.JUnit en las versiones 4.0 o superiores.

Algunas de las funciones principales del JUnit son:

- Para probar el código

  Con esto se logra que un programador se asegure que la programación básica del código fuente es correcta.

- Para documentar interfaces

  Estas pruebas nos fuerzan a documentar el comportamiento de las interfaces de acuerdo a los resultados de las pruebas, no de acuerdo a como fueron diseñadas.

- Para localizar “Bugs”

  Frecuentemente, aunque los códigos estén bien escritos sintácticamente, presentan problemas al momento de ejecutarlos. Con las pruebas del JUnit, es más fácil localizarlos para poder repararlos.

- Para reparar “Bugs”

  Ya que en el JUnit solo se reproducen o ejecutan las secciones de código que presentan problema y son introducidas a las pruebas, es más rápida la verificación y corrección de los problemas encontrados.

JAVADOC

Javadoc es una utilidad integrada en el Java Development Kit (JDK) desarrollada por Sun Mycrosystems, que permite generar documentación sobre un proyecto hecho en JAVA y exportarla cómo HTML. Esto permite tener bien documentado un programa y poder ver todas sus características (tanto métodos cómo parámetros) cómo si se tratara del sitio mismo de documentación de Java (http://java.sun.com/).

AWT Y SWING

El kit de herramientas de ventana abstracta en español o AWT en Inglés, es un kit de herramientas para implementaciones gráficas e interfaces, así como también en sistemas de ventanas independientes de la de plataforma original de java, en nuestro caso eclipse.

AWT ahora se encuentra estandarizado y se puede encontrar en el API de java ya que ahora es parte de las Java Foundation Classes o JFC.

"Algunos desarrolladores de aplicaciones prefieren este modelo porque suministra un alto grado de fidelidad al kit de herramientas nativo subyacente y mejor integración con las aplicaciones nativas. En otras palabras, un programa GUI escrito usando AWT parece como una aplicación nativa Microsoft Windows cuando se ejecuta en Windows, pero el mismo programa parece una aplicación nativa Apple Macintosh cuando se ejecuta en un Mac, etc. Sin embargo, algunos desarrolladores de aplicaciones desprecian este modelo porque prefieren que sus aplicaciones se vean exactamente igual en todas las plataformas."

Desde el comienzo de AWT que fue lanzado en 1995 AWT solo suministraba un nivel de abstracción muy fino y que a veces suministraba errores, esto ha ido siendo mejorado con el tiempo y el desarrollo del mismo.

La siguiente tabla muestra las componentes y los eventos generados así como el significado de cada uno de ellos en un sistema AWT

Component	**Eventos generados**	**Significado**
Button	ActionEvent	Hacer click en el botón
Checkbox	ItemEvent	Seleccionar o deseleccionar un item.
CheckboxMenuItem	ItemEvent	Seleccionar o deseleccionar un item.
Choice	ItemEvent	Seleccionar o deseleccionar un item.
Component	ComponentEvent	Mover, cambiar tamaño, mostrar u ocultar un componente.
	FocusEvent	Obtener o perder el focus.
	KeyEvent	Pulsar o soltar una tecla.
	MouseEvent	Pulsar o soltar un botón del ratón; entrar o salir de un componente; mover o arrastrar el ratón (tener en cuenta que este evento tiene dos Listener).
Container	ContainerEvent	Añadir o eliminar un componente de un container.
List	ActionEvent	Hacer doble click sobre un item de la lista.
	ItemEvent	Seleccionar o deseleccionar un item de la lista.
MenuItem	ActionEvent	Seleccionar un item de un menú.
Scrollbar	AdjustementEvent	Cambiar el valor de la scrollbar.

TextComponent	TextEvent	Cambiar el texto.
TextField	ActionEvent	Terminar de editar un texto pulsando intro.
Window	WindowEvent	Acciones sobre una ventana: abrir, cerrar, restablecer e iniciar el cierre.

Swing es una biblioteca gráfica que incluye en ella widgets para interfaz grafica de usuarios externos al sistema, tales como cajas de texto, botones, desplegables, tablas, entre otras.

A diferencia de los componentes AWT, el IFC era mostrado y controlado por el mismo código Java independiente de la plataforma. Estos componentes son ligeros, dado que no requieren reservar recursos del sistema de ventanas del sistema operativo y al estar desarrollado en java por completo, se puede confiar más en él para la portabilidad del sistema y el uso en diferentes plataformas.

Las ventajas de SWING son que el diseño en Java puro posee menos limitaciones de plataforma, el desarrollo de componentes Swing es más activo así como también los componentes de Swing soportan más características.

La estructura básica de un swing es la siguiente:

- En las primeras líneas se importan los paquetes necesarios.
- Se declara la clase *a usar* y en el método *main* se setea el top level container.
- Debe existir, al menos, un contenedor de alto nivel (Top-Level Container), que provee el soporte que las componentes Swing necesitan para el pintado y el manejo de eventos.
- Otras componentes colgando del contenedor de alto nivel.

“Excepto los contenedores de alto nivel, todos los componentes que empiezan con J descienden de la clase JComponent. Obtienen muchas características de esta clase, como la posibilidad de tener bordes, tooltips, y Aspecto y Comportamiento configurable. También heredan muchos métodos de conveniencia.”

BIBLIOGRAFÍA

- http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado
- http://es.wikipedia.org/wiki/Diagrama_de_clases
- http://es.wikipedia.org/wiki/Diagrama_de_componentes
- http://es.wikipedia.org/wiki/Diagrama_de_objetos
- http://es.wikipedia.org/wiki/Abstract_Window_Toolkit
- http://es.wikipedia.org/wiki/Diagrama_de_estructura_compuesta
- http://es.wikipedia.org/wiki/Diagrama_de_despliegue
- http://es.wikipedia.org/wiki/Diagrama_de_paquetes
- http://es.wikipedia.org/wiki/Diagrama_de_actividades
- http://es.wikipedia.org/wiki/Swing_(biblioteca_gr%C3%A1fica)
- http://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso
- http://es.wikipedia.org/wiki/Diagrama_de_estados
- http://es.wikipedia.org/wiki/Diagrama_de_secuencia
- http://es.debugmodeon.com/articulo/el-patron-mvc
- http://www.programacion.com/java/tutorial/swing/11/
- http://es.wikipedia.org/wiki/Diagrama_de_comunicaci%C3%B3n
- http://www.dcc.uchile.cl/~lmateu/CC60H/Trabajos/edavis/swing.html
- http://es.wikipedia.org/wiki/Diagrama_de_tiempos
- http://es.wikipedia.org/wiki/Diagrama_global_de_interacciones
- http://junit.sourceforge.net/
- http://en.wikipedia.org/wiki/JUnit
- http://usuarios.lycos.es/java_2000/pagina_nueva_1.htm
- http://www.linux.ie/articles/tutorials/junit.php
- (*) imagen tomada de wikipedia

  http://es.wikipedia.org/wiki/Archivo:Uml_diagram-es.svg