UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS



**METODOLOGÍA ORIENTADA A OBJETOS**

**Y LENGUAJE UNIFICADO DE MODELADO**

Profesor: Integrantes:

Víctor Mujica Díaz Nicolás C.I. 20347421

Caraballo José C.I. 21013733

Núñez Luis C.I. 20874644

Rojas Elhymar C.I. 21174128

Barcelona, Mayo de 2015

**1.** **Metodología Orientada a Objetos (OOM)** 4

**2.** **Fases de OMT:** 4

2.1. Análisis. 5

2.2. Diseño del sistema. 5

2.3. Diseño de objetos. 5

2.4. Implementación. 5

**3.** **Modelos de OMT** 5

3.1. Modelo de objetos. 5

3.2. Modelo dinámico. 5

3.3. Modelo funcional. 6

**4.** **Ventajas de la Metodología Orientada A Objetos** 6

4.1. Reutilización. 6

4.2. Estabilidad. 6

4.3. Calidad. 6

4.4. Integridad. 6

**5.** **Lenguaje Unificado de Modelado (UML)** 7

**6.** **Para UML se considera que el modelo de proceso de software sea** 7

6.1. Basados en casos de uso 7

6.2. Centrado en la arquitectura 7

6.3. Iterativo 7

6.4. Incremental 7

**7.** **Ciclo de Vida de UML** 7

7.1. Iniciación o concepción. 7

7.2. Elaboración. 8

7.3. Construcción, 8

7.4. Transición. 8

**8.** **Modelo de la Arquitectura** 8

**9.** **Las vistas para describir mejor el sistema son:** 9

9.1. La vista de casos de uso 9

9.1.1. Diagrama de casos de uso 9

9.1.2. Diagramas de interacción 9

9.2. La vista de diseño 9

9.2.1. Diagrama de clases 10

9.2.2. Diagrama de comunicación 10

9.2.3. Diagrama de secuencia 10

9.3. La vista de interacción 10

9.3.1. Diagrama de actividades 10

9.4. La vista de implementación 10

9.4.1. Diagrama de componentes 10

9.4.2. Diagrama de paquetes 10

9.5. La vista de despliegue 10

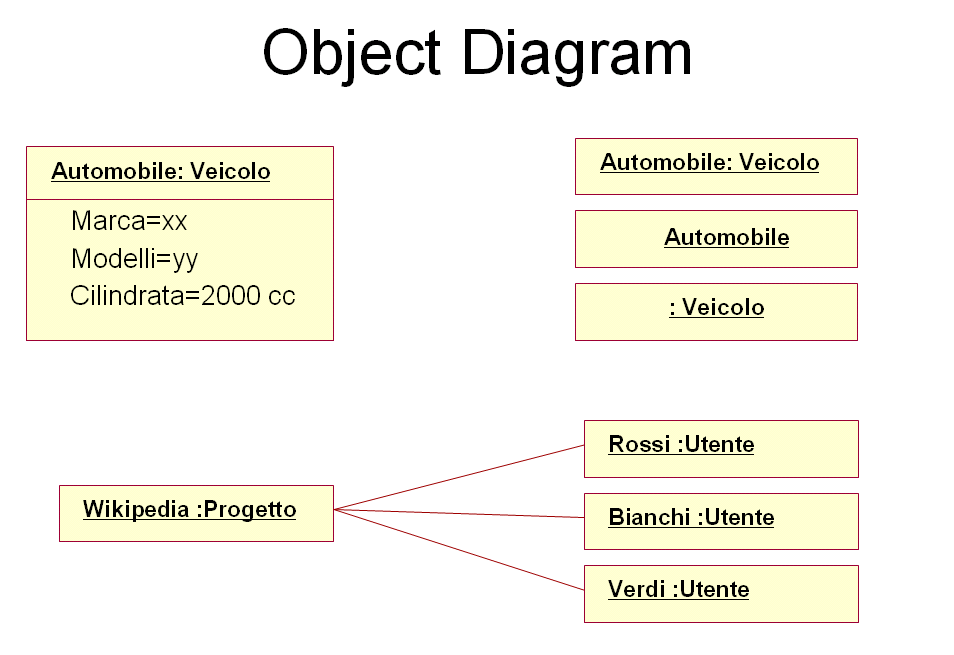
9.5.1. Diagrama de Despliegue 10

.

## **Metodología Orientada a Objetos (OOM)**

En [UML](https://es.wikipedia.org/wiki/UML), diagrama que muestra una vista completa o parcial de los [objetos](https://es.wikipedia.org/wiki/Objetos_%28programaci%C3%B3n_orientada_a_objetos%29) de un sistema en un instante de ejecución específico.

Actualmente el modelo de objetos ha sido influenciado por un número de factores no sólo de la Programación Orientada a Objetos, POO. Además, el modelo de objetos ha probado ser un concepto uniforme en las ciencias de la computación, aplicable no sólo a los lenguajes de programación sino también al diseño de interfaces de usuario, bases de datos y arquitectura de computadoras por completo. La razón de ello es, simplemente, que una orientación a objetos nos ayuda a hacer frente a la inherente complejidad de muchos tipos de sistemas.

La metodología OMT (Técnica de Modelado de Objetos) fue creada por James Rumbaugh y Michael Blaha en 1991, es una de las metodologías de análisis y diseño orientados a objetos, más maduros y eficientes que existen en la actualidad. 

## **Fases de OMT:**

### Análisis.

El Analista Construye un modelo del dominio del problema, mostrando sus propiedades más importantes. El modelo de análisis es una abstracción resumida y precisa de lo que debe de hacer el sistema deseado y no de la forma en que se hará. Los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos tales como estructuras de datos. Un buen modelo debe poder ser entendido y criticado por expertos en el dominio del problema que no tengan conocimientos informáticos.

### Diseño del sistema.

**El Diseñador del Sistema** toma decisiones de alto nivel sobre la Arquitectura del Software. Durante esta fase el sistema se organiza en subsistemas basándose tanto en la estructura del análisis como en la arquitectura propuesta. Se selecciona una estrategia para afrontar el problema.

### Diseño de objetos.

**El Diseñador de Objetos** construye un modelo de diseño basándose en el modelo de análisis, pero incorporando detalles de implementación. El diseño de objetos se centra en las estructuras de datos y algoritmos que son necesarios para implementar cada clase. OMT describe la forma en que el diseño puede ser implementado en distintos lenguajes (orientados y no orientados a objetos, bases de datos, etc.).

### Implementación.

Los Objetos de Clases y relaciones desarrollados durante el análisis de objetos se traducen finalmente a una implementación concreta. Durante la fase de implementación es importante tener en cuenta los principios de la ingeniería del software de forma que la correspondencia con el diseño sea directa y el sistema implementado sea flexible y extensible. No tiene sentido que utilicemos AOO y DOO de forma que potenciemos la reutilización de código y la correspondencia entre el dominio del problema y el sistema informático, si luego perdemos todas estas ventajas con una implementación de mala calidad.

## **Modelos de OMT**

* 1. Modelo de objetos.

Describe la estructura estática de los objetos del sistema. El modelo de objetos proporciona el entorno esencial en el cual se pueden situar el modelo dinámico y el modelo funcional. El objetivo es capturar aquellos conceptos del mundo real que sean importantes para la aplicación. Se representa mediante diagramas de objetos.

* 1. Modelo dinámico.

Describe los aspectos de un sistema que tratan de la temporización y secuencia de operaciones (sucesos que marcan los cambios, secuencias de sucesos, estados que definen el contexto para los sucesos) y la organización de sucesos y estados. Captura el control, aquel aspecto de un sistema que describe las secuencias de operaciones que se producen sin tener en cuenta lo que hagan las operaciones, aquello a lo que afecten o la forma en que están implementadas. Se representa gráficamente mediante diagramas de estado.

* 1. Modelo funcional.

Describe las transformaciones de valores de datos que ocurren dentro del sistema. Captura lo que hace el sistema, independientemente de cuándo se haga o de la forma en que se haga. Se representa mediante diagramas de flujo de datos

## **Ventajas de la Metodología Orientada A Objetos**

* 1. Reutilización.

Las clases están diseñadas para que se reutilicen en muchos sistemas. Para maximizar la reutilización, las clases se construyen de manera que se puedan adaptar a los otros sistemas. Un objetivo fundamental de las técnicas orientadas a objetos es lograr la reutilización masiva al construir el software.

* 1. Estabilidad.

Las clases diseñadas para una reutilización repetida se vuelven estables, de la misma manera que los microprocesadores y otros chips se hacen estables.

El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel. El encapsulamiento oculta los detalles y hace que las clases complejas sean fáciles de utilizar.

Se construyen clases cada vez más complejas. Se construyen clases a partir de otras clases, las cuales a su vez se integran mediante clases. Esto permite construir componentes de software complejos, que a su vez se convierten en bloques de construcción de software más complejo.

* 1. Calidad.

Los diseños suelen tener mayor calidad, puesto que se integran a partir de componentes probados, que han sido verificados y pulidos varias veces.

* 1. Integridad.

Las estructuras de datos (los objetos) sólo se pueden utilizar con métodos específicos. Esto tiene particular importancia en los sistemas cliente-servidor y los sistemas distribuidos, en los que usuarios desconocidos podrían intentar el acceso al sistema.

## **Lenguaje Unificado de Modelado (UML)**

Es un lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, serie de normas y estándares gráficos respecto a cómo se deben representar los esquemas relativos al software.

## **Para UML se considera que el modelo de proceso de software sea**

* 1. Basados en casos de uso

Que los casos de uso sean un artefacto básico para establecer el comportamiento deseado del sistema, para validar la arquitectura, para las pruebas y para la comunicación entre las personas involucradas en el proyecto.

* 1. Centrado en la arquitectura

Que sea el artefacto básico para conceptuar, construir, gestionar y hacer evolucionar el sistema.

* 1. Iterativo

Que involucre la gestión del flujo de ejecutables del sistema.

* 1. Incremental

Donde cada nueva versión corrige defectos de la anterior e incorpora nueva funcionalidad. Un proceso iterativo e incremental se denomina dirigido por el riesgo, lo que significa que cada nueva versión se ataca y reducen los riesgos más significativos para el éxito del proyecto.

## **Ciclo de Vida de UML**

Durante el ciclo de vida de un proyecto software existen cuatro fases por las cuales debe pasar el proceso de desarrollo iterativo, entre las cuales se encuentran las siguientes:

* 1. Iniciación o concepción.

En ésta fase se desarrollan las ideas para otorgar un sistema final, es decir, pensar en lo que realizará el sistema, como estará compuesto arquitectónicamente, cual es el plan a seguir para lograrlo y además el costo que tendrá el mismo sistema, tomando en cuenta los riesgos que se puedan presentar durante la elaboración de la aplicación. Un buen sistema debe de pensar en todo esto antes de realizarlo.

* 1. Elaboración.

Se deben determinar los requisitos del sistema y las pruebas sobre el mismo. En esta fase se establecen en detalle los casos de uso que se van a implementar para el desarrollo del programa y a la vez se diseña la estructura del sistema, teniendo al final de esta fase las actividades planeadas y los recursos necesarios para que se ejecute el proyecto.

* 1. Construcción,

Es cuando se pasa de la base arquitectónica ejecutable hasta su disponibilidad para los usuarios. Se reexaminan los requisitos y las pruebas que ha de soportar. Como su nombre lo dice, se comienza a construir (programar) la aplicación tomando en cuenta la estructura que se realizó del proyecto durante la etapa anterior. En esta fase se debe de tener un avance del sistema a crear, para que los usuarios otorguen opiniones de la aplicación y si existe algún error o deficiencia se puedan realizar los cambios necesarios para que este se desarrolle perfectamente.

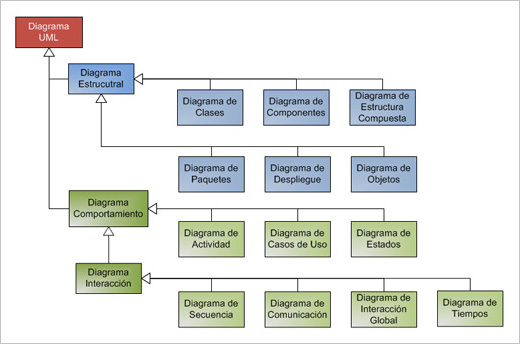
* 1. Transición.

El proceso del software termina en la etapa de transición, ya que es continuamente reexaminado y mejorado erradicando errores y añadiendo nuevas funcionalidades no contempladas. En esta parte de desarrollo se otorga el sistema a los usuarios para que se realicen pruebas del sistema, se pudiese llamar como si fuera una versión beta del programa, para si existiese fallos se puedan corregir con anterioridad y cumplir con el objetivo señalado en tiempo y forma.

La iteración afecta a estas cuatro fases. Una iteración es la repetición de una serie de instrucciones repetitivas que comprenden el proceso para lograr un sistema, como ejemplo de ello en programación se puede mencionar la recursividad, ya que esta permite repetir las instrucciones hasta que se cumpla una acción en específico. Este elemento está presente en UML como un conjunto bien definido de actividades, con un plan y unos criterios de evaluación, que acaban en una versión del producto.

## **Modelo de la Arquitectura**

La arquitectura es la que se encarga de describir la organización del sistema, selección de elementos estructurales y sus interfaces, el comportamiento, las colaboraciones entre los componentes del sistema, composición de los elementos estructurales



## **Las vistas para describir mejor el sistema son:**

* 1. La vista de casos de uso

Comprende la descripción del comportamiento del sistema tal y como es percibido por los usuarios finales, analistas y encargados de las pruebas y se utilizan los diagramas de casos de uso para capturar los aspectos estáticos mientras que los dinámicos son representados por diagramas de interacción, estados y actividades.

#### Diagrama de casos de uso

#### Diagramas de interacción

* 1. La vista de diseño

Comprende las clases, interfaces y colaboraciones que forman el vocabulario del problema y de la solución. Esta vista soporta principalmente los requisitos funcionales del sistema, o sea, los servicios que el sistema debe proporcionar. Los aspectos estáticos se representan mediante diagramas de clases y objetos y los aspectos dinámicos con diagramas de interacción, estados y actividades.

#### Diagrama de clases

#### Diagrama de comunicación

#### Diagrama de secuencia

* 1. La vista de interacción

Comprende los hilos y procesos que forman mecanismos de sincronización y concurrencia del sistema cubriendo el funcionamiento, capacidad de crecimiento y el rendimiento del sistema. Con UML, los aspectos estáticos y dinámicos se representan igual que en la vista de diseño, pero con el énfasis que aportan las clases activas, las cuales representan los procesos y los hilos.

#### Diagrama de actividades

* 1. La vista de implementación

Comprende los componentes y los archivos que un sistema utiliza para ensamblar y hacer disponible el sistema físico. Se ocupa principalmente de la gestión de configuraciones de las distintas versiones del sistema. Los aspectos estáticos se capturan con los diagramas de componentes y los aspectos dinámicos con los diagramas de interacción, estados y actividades.

#### Diagrama de componentes

#### Diagrama de paquetes

* 1. La vista de despliegue

De un sistema contiene los nodos que forman la topología hardware sobre la que se ejecuta el sistema. Se preocupa principalmente de la distribución, entrega e instalación de las partes que constituyen el sistema. Los aspectos estáticos de esta vista se representan mediante los diagramas de despliegue y los aspectos dinámicos con diagramas de interacción, estados y actividades.

#### Diagrama de Despliegue

