

***DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN***

**CARRERA: INGENIERÍA EN SISTEMAS E INFORMATICA**

**SEXTO SEMESTRE**

DESARROLLO DE APLICACIONES WEB

**NRC:** 3871

**Programacion por componentes**

**INTEGRANTES:**

HENRY CARRERA

JOHANA ROSERO

WILLIAM MAILA  
RICARDO MUÑOZ

**PROFESOR:**

Ing. Raura Ruiz, Jorge Geovanny

2017-11-25

SANGOLQUI – ECUADOR

**Programación Orientada a Objetos**

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación. Esta forma de programar es mucho más simple que otras y nos da la ventaja de ser más rápido ya que se manejan en forma de "bloque"

**Características:**

**Abstracción**

Cada objeto sirve como modelo abstracto que puede realizar trabajos. Informar, cambiar su estado y  comunicarse con otros objetos en el sistema sin revelar como se implementan estas características

**Encapsulación**

También llamado ocultación de información asegura que no se pueda cambiar el estado de otros objetos, cada objeto tiene una interfaz diferente de los otros

**Polimorfismo**

Pueden tener objetos de diferentes tipos y la invocación  de un comportamiento correcto para el tipo real de referente esto ocurre durante  el tiempo de ejecución.

**Herencia**

Organiza y facilita el polimorfismo y el encapsulamiento permitiendo que los objetos sean definidos y creados especialmente.

**Ventajas**

* Fomenta la reutilización y extensión del código.
* Permite crear sistemas más complejos.
* Relacionar el sistema al mundo real.
* Facilita la creación de programas visuales.
* Construcción de prototipos
* Agiliza el desarrollo de software
* Facilita el trabajo en equipo
* Facilita el mantenimiento del software
* Todo el código se encuentra en un solo lugar
* Los objetos pueden tener varios atributos por ejemplo, que lea un sensor y a la vez encienda.
* Son más fáciles de entender los códigos.
* Le ejecución del programa es rápida y sencilla, todo se encuentra en una sola ventana.

**Desventajas**

* Los programas no pueden ser moldeados enteramente por la programación orientada a objetos.
* Para leer, modificar, o hacerles algo simplemente; en algunos programas debes realizar un paso extra para realizar estas acciones.
* Si se fuerza el lenguaje puede perder algunos objetos y características.
* Los objetos requieren una extensa documentación.
* Los objetos al ser abstracto pueden no coincidir la visión de un programador a otro.

**Programación Orientada a Componentes**

Extensión natural de la programación orientada a objetos (POO) en los sistemas de aplicación abiertos, que tiene como propósito contrarrestar las limitaciones de esta, como la falta de una unidad concreta de composición independiente en las aplicaciones, y la definición de interfaces a bajo nivel, que dificultan la reutilización comercial de objetos.

La programación orientada a componentes se centra en módulos intercambiables que trabajan de forma independiente y de los cuales no es necesario saber nada acerca de sus implementación interna.

La programación orientada a componentes, lo que predica es la llamada reutilización de caja negra, es decir, permite usar un componente existente sin necesidad de conocer los detalles internos de su implementación. Esto se logra a través de las interfaces. En vez de diseñar complejas jerarquías de clases, se diseñan interfaces usadas como contratos entre componentes, clientes y servidores. O sea, que hay que tender a diseñar jerarquías de clases simples y centrarse en la programación orientada a interfaces.

**Características**

La ingeniería de software basada en componentes es una rama de la ingeniería de software que enfatiza la separación de asuntos por lo que se refiere a la funcionalidad de amplio rango disponible a través de un sistema de software dado.

Es un acercamiento basado en la reutilización para definir, implementar, y componer componentes débilmente acoplados en sistemas. Esta práctica persigue un amplio grado de beneficios tanto en el corto como el largo plazo, para el software en sí mismo y para las organizaciones que patrocinan tal software.

**Ventajas**

* Una aplicación orientada a componentes es más fácil de extender (cumplir con nuevos requisitos a través de nuevos componentes sin tocar los ya existentes).
* Reduce los costes de mantenimiento de las aplicaciones y normalmente sus desarrollos suelen ser más cortos en términos de tiempo.
* Facilidad de instalación, se pude remplazar sin perjudicar la funcionalidad de otro componente.
* Reusable, se puede utilizar para otras aplicaciones.

**Desventajas**

* **Clarividencia**: se refiere a la dificultad con la que se encuentra el diseñador de un componente al realizar su diseño, pues no conoce ni quién lo utilizará, ni cómo, ni en qué entorno, ni para qué aplicación. Este problema está muy ligado a la composición tardía y reusabilidad de los componentes.
* **Evolución de los componentes**: la gestión de la evolución es un problema serio, pues en los sistemas grandes han de poder coexistir varias versiones de un mismo componente.
* **Percepción del entorno**: posibilidad de un componente de descubrir tanto el tipo de entorno en donde se está ejecutando (de diseño o de ejecución), como los servicios y recursos disponibles en él.
* **Particularización**: cómo particularizar los servicios que ofrece un componente para adaptarlo a las necesidades y requisitos concretos de la aplicación, sin poder manipular su implementación.
* **Falta de soporte formal**: la dificultad que encuentran los métodos formales para trabajar con sus peculiaridades, como puede ser la composición tardía, el polimorfismo o la evolución de los componentes.
* **El problema de la clase base frágil (FBCP):** este problema ocurre cuando la superclase de una clase sufre modificaciones. El FBCP existe a dos niveles, sintáctico y semántico. A nivel sintáctico ocurre cuando las modificaciones de la superclase son puramente a este nivel. A nivel semántico ocurre cuando lo que se altera es la implementación de los métodos de la superclase.
* **Asincronía y carreras de eventos**: problema que se presenta por los tiempos de comunicación en los sistemas abiertos (no se pueden despreciar retrasos). Es muy difícil garantizar el orden relativo en el que se distribuyen los eventos. El proceso de difusión de eventos es complicado cuando los emisores y receptores pueden cambiar con el tiempo.
* **Interoperabilidad**: los contratos de los componentes se reducen a la definición de sus interfaces a nivel sintáctico, y la interoperabilidad se reduce a la comprobación de los nombres y perfiles de los métodos. Sin embargo, es necesario ser capaces de buscar los servicios que se necesitan por algo más que sus nombres, y poder utilizar los métodos ofrecidos en una interfaz en el orden adecuado.

**Programación Orientada a Objetos vs. Programación Orientada a Componentes**

La programación orientada a objetos se focaliza en las relaciones que hay entre las clases combinadas dentro de un gran ejecutable binario, mientras que la programación orientada a componentes se centra en módulos intercambiables que trabajan de forma independiente y de los cuales no es necesario saber nada acerca de sus implementación interna.

La diferencia entre ambas técnicas es la manera que tienen ellas de ver a la aplicación final. En la programación orientada a objetos, el resultado es un código binario monolítico, todas las clases se localizan en él, mientras que la programación orientada a componentes se puede imaginar como las piezas del lego, un cambio en la implementación de una de las piezas está disponible inmediatamente para todo cliente que la use, sin necesidad de recompilar.

BIBLIOGRAFIA

<http://poao4.blogspot.com/2015/11/poo-que-es-ventajas-y-desventajas.html#!/2015/11/poo-que-es-ventajas-y-desventajas.html>

<http://www.ciberaula.com/articulo/tecnologia_orientada_objetos>

<https://desarrolloweb.com/articulos/499.php>

<https://www.ecured.cu/Programaci%C3%B3n_orientada_a_componentes>

<https://prezi.com/ir3_wycbqsrw/programacion-orientada-a-componentes/>