I. INTRODUCCIÓN

El reciente aumento de aplicaciones en donde se utiliza la computadora ha sido posible debido a un hardware de bajo costo, por lo cual la demanda de software ha crecido de forma exponencial. Esto implica que son necesarias técnicas y tecnología eficientes de Ingeniería de Software para resolver los múltiples problemas que se derivan de las aplicaciones en donde se desarrollan sistemas software de gran tamaño.

La Ingeniería de Software implica seguir en cualquier proyecto de software una metodología de desarrollo y la utilización de distintas técnicas y herramientas. Los diferentes procedimientos a seguir en cualquier proyecto de Ingeniería de software son: Definición de requerimientos, Análisis, Diseño, Verificación y Validación (Pruebas de Calidad del Software), Pruebas y Mantenimiento.

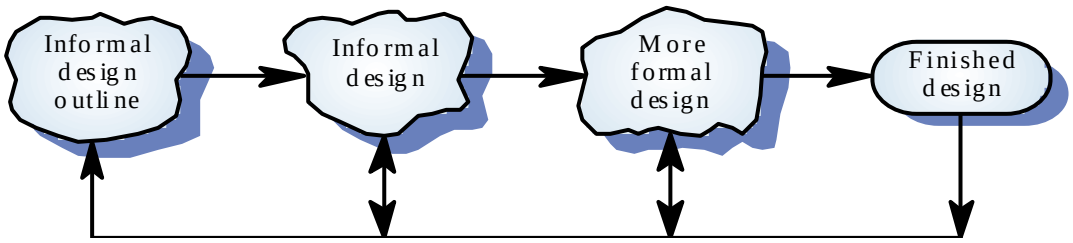
El presente documento intenta dar a conocer y describir los conceptos y aspectos fundamentales del diseño orientado a objetos (DOO) dentro del desarrollo de un producto software, así como las técnicas, metodologías y herramientas actuales de dicho paradigma en la Ingeniería de software.

Así pues, definimos **Diseño de Software** como la acción de construir soluciones que satisfagan los requerimientos del cliente. Existen varias etapas en el proceso de diseño de software, a saber son:

- Entendimiento del problema
- Identificar una o mas soluciones
- Describir abstracciones de la solución
- Repetir el proceso para cada abstracción identificada hasta que el diseño este expresado en términos sencillos

Cualquier diseño debe ser modelado como una gráfica dirigida hecha de entidades con atributos los cuales participan en relaciones. El sistema debe estar descrito a distintos niveles de abstracción y el diseño ocurre en etapas que se traslapan.

La primera idea que se tiene al construir una solución de un determinado problema es un modelo mental que constituye el primer intento de diseño llamado comúnmente **diseño informal**. Este diseño a medida que se va describiendo en papel utilizando técnicas y procedimientos esquemáticos y metódicos va adquiriendo forma hasta constituirse en un **diseño formal** equivalente. La siguiente figura ejemplifica este hecho:



Pues bien, dentro del paradigma de la orientación a objetos, el diseño OO es con mucho; más complejo que el diseño estructurado clásico, ya que lo que se busca es crear un diseño genérico y abierto y no cerrado y concreto.

El **Diseño Orientado a Objetos** se define como un diseño de sistemas que utiliza objetos auto-contenidos y clases de objetos.

Características principales del Diseño Orientado a Objetos:

u Los objetos son abstracciones del mundo real o entidades del sistema que se administran entre ellas mismas

uLos objetos son independientes y encapsulan el estado y la representación de información

uLa funcionalidad del sistema se expresa en términos de servicios de los objetos

uLas áreas de datos compartidas son eliminadas. Los objetos se comunican mediante paso de parámetros

uLos objetos pueden estar distribuidos y pueden ejecutarse en forma secuencial o en paralelo

Ventajas del Diseño Orientado a Objetos:

uFácil de mantener, los objetos representan entidades auto-contenidas

uLos objetos son componentes reutilizables

uPara algunos sistemas, puede haber un mapeo obvio entre las entidades del mundo real y los objetos del sistema

Desarrollo Orientado a Objetos:

uEl análisis, diseño y programación orientada a objetos están relacionados pero son diferentes

uEl análisis orientado a objetos concierne al desarrollo del modelo de objetos del dominio de la aplicación

uEl Diseño Orientado a Objetos trata del desarrollo del modelo del sistema orientado a objetos para implementar los requerimientos

uLa programación orientada a objetos trata de la realización del Diseño Orientado a Objetos utilizando algún lenguaje de programación orientada a objetos como C++

Métodos de Diseño Orientado a Objetos

uAlgunos métodos que fueron originalmente basados en funciones (método de Yourdon) han sido adaptadas al diseño orientado a objetos. Otros métodos como el método de Booch han sido específicamente desarrolladas específicamente para el Diseño Orientado a Objetos

uEl Diseño Orientado a Objetos es un método de diseño desarrollado para soportar la programación en Ada.

uJSD (Jackson system development) tiene una cierta orientación a objetos pero no contiene información sobre estados entidad

Componentes del Diseño Orientado a Objetos

uLa identificación de objetos, sus atributos y servicios

uLa organización de objetos dentro de una jerarquía

uLa construcción de descripciones dinámicas de objetos que muestran como se usan los servicios

uLa especificación de interfaces de objetos

Objetos, Clases y Herencia:

uLos objetos son entidades en un sistema de software que representan instancias de entidades del mundo real

uLas clases objetos son templates para objetos. Pueden usarse para crear objetos

uLas clases objetos pueden heredar atributos y servicios de otras clases objetos

Objetos:

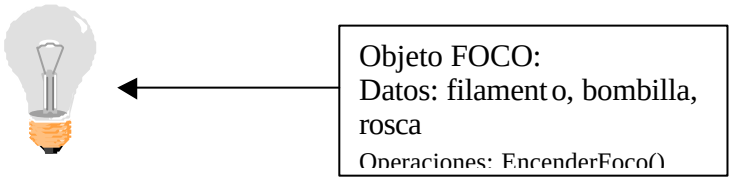
- Un objeto es una entidad que tiene un estado y un conjunto definido de operaciones que operan sobre este estado.

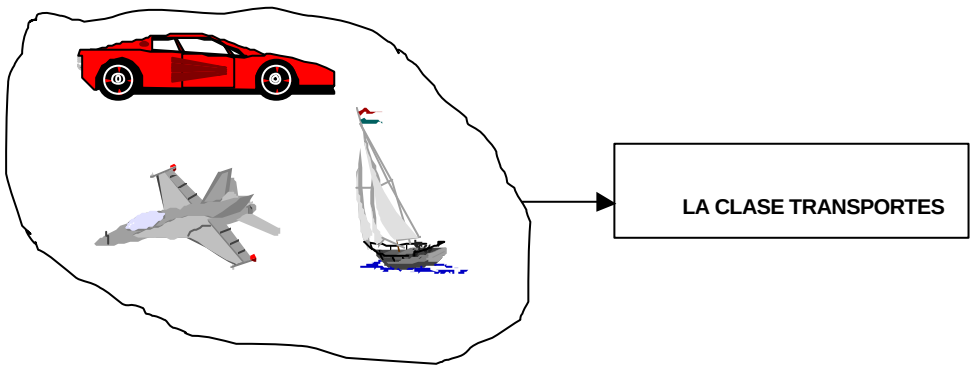
- El estado se representa mediante un conjunto de atributos del objeto.

- Las operaciones asociadas con el objeto proveen servicios a otros objetos (clientes) que requieren estos servicios cuando necesitan realizar alguna actividad de cómputo.

- Los objetos se crean de acuerdo a una definición de la clase objeto.

- La definición de la clase objeto sirve como template para los objetos. Esta incluye las declaraciones de todos los atributos y servicios los cuales deben estar asociados con un objeto de esta clase.





Comunicación entre Objetos:

uConceptualmente los objetos se comunican mediante paso de mensajes.

uMensajes

•El nombre del servicio requerido por algún objeto que llama.

•Copias de la información requerida para ejecutar el servicio y el nombre de quien tiene esta información.

uEn la practica, los mensajes se implementan a menudo mediante llamadas de procedimientos

•Nombre = nombre de procedimiento.

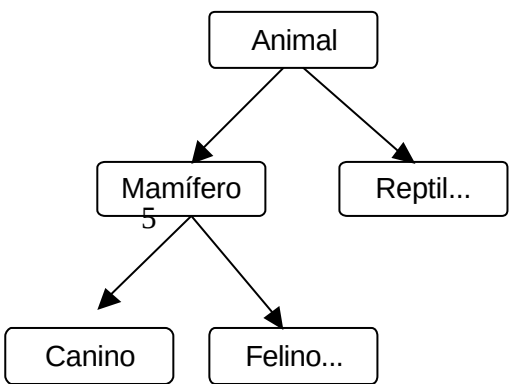
•Información = lista de parámetros.

Herencia:

uLos objetos son miembros de clases que define tipos de atributos y operaciones

uLas clases pueden organizarse en jerarquías de clases donde una clase se deriva de una clase existente (super-clase)

uUna sub-clase hereda los atributos y operaciones de su super-clases y puede añadir nuevos métodos o atributos propios



Herencia y Diseño Orientado a Objetos

- Existen distintos puntos de vista de la herencia respecto al Diseño Orientado a Objetos.

  - Punto de vista 1. Identificar la jerarquía de herencias es una parte fundamental del diseño orientado a objetos. Obviamente esto solo puede implementarse utilizando lenguajes de programación orientados a objetos.

  - Punto de vista 2. La herencia es un concepto útil para implementación que permite reusar los atributos y las operaciones. La identificación de la jerarquía de herencias en la fase de diseño pone muchas restricciones en la implementación.

Identificación de Objetos:

- La identificación de objetos es la parte más difícil del diseño orientado a objetos

- No hay una formula mágica para la identificación de objetos. Se base en la experiencia y el conocimiento del dominio de los diseñadores del sistema

- la identificación de objetos es un proceso iterativo. Es difícil que salga bien la primera vez

En Resumen:

- El diseño orientado a objetos es un diseño con ocultamiento de información. La representación puede cambiarse sin cambios muy extensos.

- Un objeto tiene un estado privado con un constructor asociado y operaciones de acceso. Los objetos proveen servicios (operaciones) a otros objetos.

- La identificación de objetos es un proceso difícil. La identificación de sustantivos y verbos en lenguaje natural es útil para identificar objetos

- Las interfaces de objetos deben ser precisamente definidas. Un lenguaje de programación como Ada, C++ o JAVA puede usarse para esto

- Documentación útil para el diseño orientado a objetos incluyen, gráficas de jerarquía de objetos y diagramas de interacción de objetos.

- Los objetos puede implementarse como entidades secuenciales o concurrentes.

II. DISEÑO DE SISTEMAS ORIENTADOS AOBJETOS:

El diseño Orientado a Objetos (DOO) difiere considerablemente del diseño estructurado ya que en DOO no se realiza un problema en términos de tareas (subrutinas) ni en términos de datos, sino (como ya se vio en la introducción) se analiza el problema como un sistema de objetos que interactúan entre sí.

Un problema desarrollado con técnicas orientadas a objetos requiere, en primer lugar saber cuales son los objetos del programa. Como tales objetos son instancias de clases, la primera etapa en el desarrollo orientado a objetos requiere de la identificación de dichas clases (atributos y comportamiento), así como las relaciones entre éstas y su posterior implementación en un lenguaje de programación.

Existen numerosos métodos de diseño orientado a objetos: Booch, Yourdon-Coad, Martín, Shlaer & Mellor, Rumbaugh, por citar algunos. Pero en general como ocurre en cualquier proyecto estructurado, un proyecto software OO se compone de las siguientes etapas:

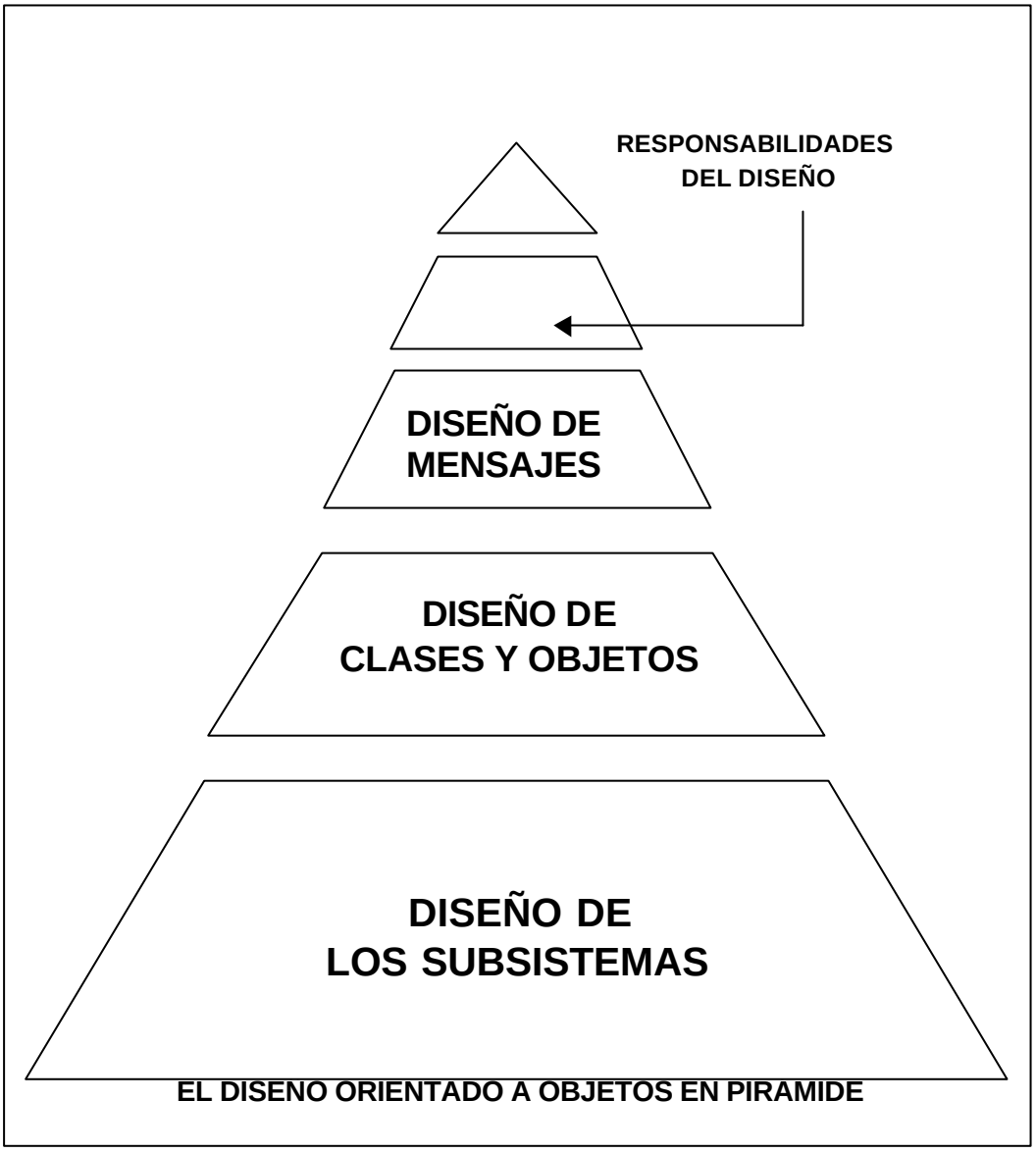
- Análisis Orientado a Objetos (AOO)
- Diseño Orientado a Objetos (DOO)
- Programación Orientada a Objetos (POO)

Aunque no siempre están bien delimitadas las etapas de análisis y diseño en la OO, se pueden sintetizar de alguna forma las ideas claves de las distintas tecnologías existentes dentro del desarrollo orientado a objetos al que denominaremos *diseño*.

El **método de Booch** considera que las etapas del proceso en un desarrollo orientado a objetos son:

1. Identificar las claves y objetos en un nivel dado de abstracción
2. Identificar la semántica de estas clases y objetos
3. Identificar las relaciones entre clases y objetos
4. Especificar la interfaz y la implementación de estas clases y objetos

Estas etapas suelen seguirse por la mayoría de los métodos de diseño OO existentes. De hecho, para los sistemas orientados a objetos se define el siguiente **diseño en pirámide** que contempla el método de Booch.



EL DISEÑO ORIENTADO A OBJETOS EN PIRAMIDE

La capa del subsistema.- Contiene una representación de cada uno de los subsistemas que le permiten al software conseguir los requisitos definidos por el cliente e implementar la infraestructura técnica que los soporta.

La capa de clases y Objetos.- Contiene las jerarquías de clase que permiten crear el sistema usando generalizaciones y especializaciones mejor definidas. Esta capa también contiene representaciones de diseño para cada objeto.

La capa de mensajes.- Contiene los detalles que le permiten a cada objeto comunicarse con sus colaboradores. Esta capa establece las interfaces externas e internas para el sistema.

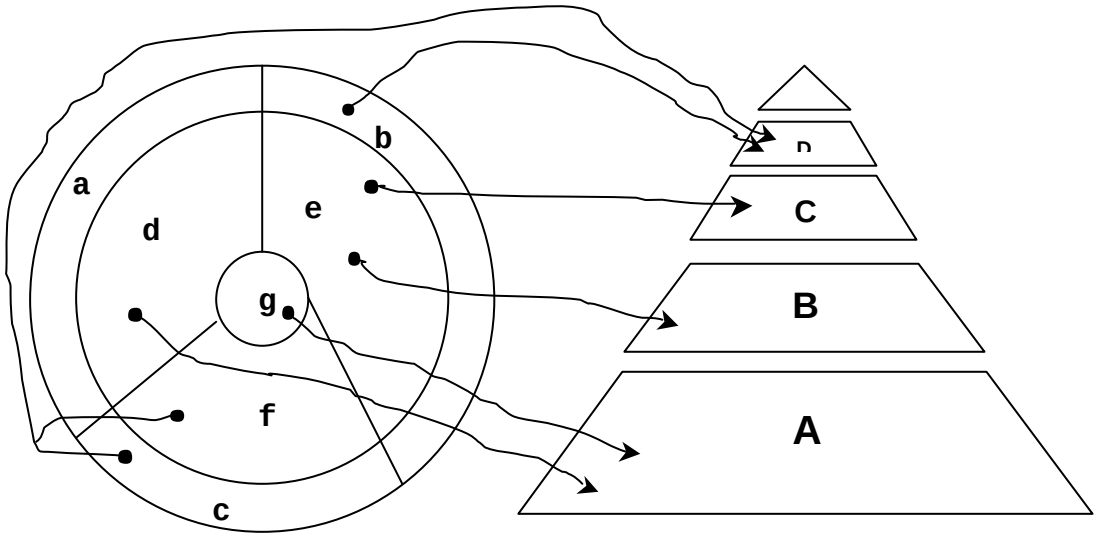
La capa de responsabilidades.- Contiene las estructuras de datos y el diseño algorítmico para todo los atributos y operaciones de cada objeto.

Esta pirámide de diseño se centra entonces en el diseño de un producto o sistema específico.

II.1. EL ENFOQUE CONVENCIONAL Y EL ENFOQUE OO

Los enfoques convencionales para el diseño del software aplican notaciones y heurísticas diferentes para establecer correspondencias entre el modelo de análisis y el diseño. Si recordamos la ingeniería de software clásica (imperativa), veremos que cada elemento del modelo de análisis convencional tiene correspondencia con una o más capas del modelo de diseño tal como lo ilustra la siguiente figura:

Transformación Análisis a Diseño en Ingeniería clásica



El modelo de Análisis **El modelo de Diseño**

a.- Descripción de los objetos de datos

b.- Especificación del proceso → **D.-** Diseño Procedimental

c.- Especificación de control → **D.-** Diseño Procedimental

d.- Diagrama Entidad-Relación → **A.-** Diseño de Datos

e.- Diagrama de flujo de datos → **B.-** Diseño Arquitectónico
→ **C.-** Diseño de Interfaz

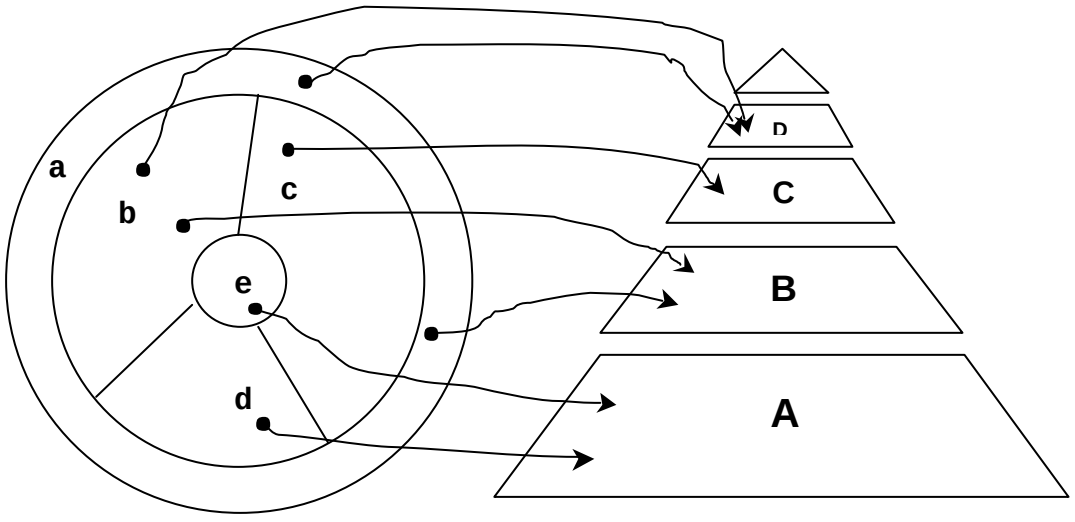
f.- Diagrama de transición de estado → **D.-** Diseño Procedimental

g.- Diccionario de datos → **A.-** Diseño de datos

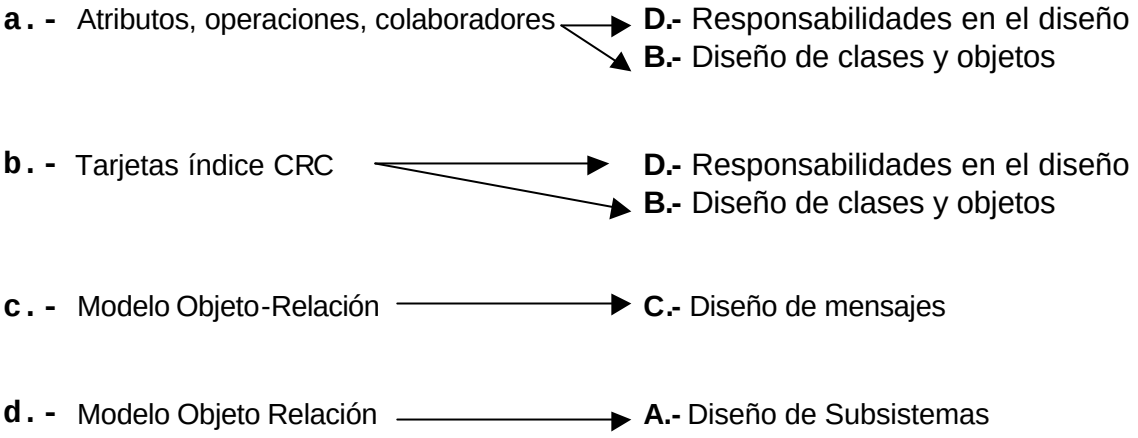
Al igual que el diseño de software convencional, el DOO aplica diseño de datos (cuando se representan atributos), diseño de interfaces (cuando se presenta el intercambio de mensajes) y diseño procedimental (en el diseño de operaciones), no obstante el diseño arquitectónico es diferente.

La arquitectura de diseño OO se centra más en las colaboraciones entre los objetos que con el flujo de control de datos. De esta manera las capas de la pirámide se renombran para reflejar de forma más exacta la naturaleza del DOO. La siguiente figura muestra ahora la correspondencia entre el AOO con las correspondientes capas de la pirámide de diseño OO.

Transformación Análisis a Diseño en Ingeniería OO



El modelo de Análisis OO **El modelo de Diseño OO**



II.2.- EL DISEÑO:

Bertrand Meyer sugiere los siguientes criterios para poder juzgar la capacidad que poseé un método de diseño en poder lograr ciertos elementos importantes tales como la modularidad:

Descomponibilidad.- Facilidad con la cual un método de diseño ayuda al diseñador a descomponer un gran problema en subproblemas más sencillos de resolver.

Componibilidad.- Grado con el cual un método de diseño asegura que los componentes de un programa (módulos), una vez diseñados y construidos, pueden reusarse para crear otros sistemas.

Comprensibilidad.- Facilidad de comprensión de un componente de programa sin referencia a otra información o módulos.

Continuidad.- Facilidad de hacer pequeños cambios en un programa y hacer que estos se manifiesten por sí mismos en cambios correspondientes solamente en no o unos pocos módulos más.

Protección.- Característica arquitectónica que reducirá la propagación de efectos colaterales si ocurre un error en un módulo dado.

Éstos criterios y principios de diseño presentados por Meyer pueden aplicarse a cualquier método de diseño (incluyendo diseño estructurado), no obstante el método de diseño orientado a objetos alcanza cada uno de los principios de manera más eficiente que otros enfoques y el resultado final es una arquitectura modular que permite cumplir con todos los principios de modularidad de una manera más eficiente.