**Índice**

[**Metodologías Orientadas a Objetos** 2](#_Toc5812306)

[**Estos conceptos se explicarán en los siguientes puntos.** 2](#_Toc5812307)

[**Los objetivos clave del diseño orientado al objeto son:** 3](#_Toc5812308)

[**Las técnicas orientadas a objetos resuelven este problema, de manera que:** 4](#_Toc5812309)

[**El paradigma objetual** 4](#_Toc5812310)

[**HISTORIA Y MITOS** 5](#_Toc5812311)

[**MITOS DE LA POO. ETIQUETAS QUE NO SON CIERTAS** 6](#_Toc5812312)

[**Beneficios de la M.O.O.** 7](#_Toc5812313)

[**Conclusión** 7](#_Toc5812314)

[**Bibliografía** 8](#_Toc5812315)

# **Metodologías Orientadas a Objetos**

El diseño orientado al objeto, al igual que otras metodologías de diseño orientadas a

la información, crea una representación del campo del problema del mundo real y lo

hace corresponder con el ámbito de la solución, que es el software.

El diseño orientado al objeto produce un diseño que interconecta objetos de datos

(elementos dato) y operaciones de una forma que modulariza la información y el

procesamiento; por el contrario, otros métodos dejan aparte el procesamiento.

La naturaleza única del diseño orientado al objeto queda reflejada en su capacidad

de construir sobre tres pilares conceptuales importantes del diseño de software:

* Abstracción
* Encapsulamiento
* Modularidad

## **Estos conceptos se explicarán en los siguientes puntos.**

El análisis orientado al objeto (AOO), el diseño orientado al objeto (DOO) y la

producción orientada al objeto comprenden un conjunto de actividades de Ingeniería del Software para la construcción del sistema orientado a objetos.

Utilizando el diseño orientado a objetos el diseñador puede crear sus propios tipos

abstractos de datos y abstracciones funcionales y hacer corresponder el campo del

mundo real con esas abstracciones creadas por el propio programador. Esta

correspondencia será la mayoría de las veces mucho más natural, ya que el rango de

tipos abstractos de datos que puede inventar el diseñador es virtualmente ilimitado. Más aún, el diseño del software se desliga de los detalles de representación, sin que ello

afecte al sistema de software global.

## **Los objetivos clave del diseño orientado al objeto son:**

1. Aumentar la productividad: Según algunos estudios, el diseño orientado al

objeto logra aumentar la productividad de un desarrollo en un 20 %, lo cual

no es mucho. Sin embargo, sabemos que entre el 75% y el 80% del coste de

un sistema se produce después del desarrollo inicial. Pues bien, es

precisamente en esta fase donde el diseño orientado al objeto puede

ayudarnos a aumentar de forma espectacular la productividad.

1. Incrementar calidad: Cuando hacemos referencia al término calidad no solo

nos estamos refiriendo a la ausencia de errores, sino también a otros

aspectos quizás no tan fáciles de medir como son la facilidad de uso, la

portabilidad o la facilidad de modificación.

1. Facilidad de mantenimiento: Debemos partir de la base que es imposible

prever cambios que se producirán en meses o quizás años posteriores, pero

lo que si podemos hacer es separar las partes del sistema que son

intrínsecamente volátiles de aquellas que pueden ser estables. Los aspectos

más volátiles podrían ser la interface externa, los atributos que describen

ítems en el dominio del problema; ... Mientras que los más estables deben ser

las clases.

1. Continuidad en la representación: Uno de los problemas más importantes

que presentan las metodologías estructuradas clásicas es el problema de

comunicación existente entre las fases de Análisis y Diseño, e incluso

dentro de la primera, entre los DFD (Diagramas de Flujo de Datos) y los

DER (Diagramas Entidad-Relación).

## **Las técnicas orientadas a objetos resuelven este problema, de manera que:**

* No hay diferencias entre la notación empleada en el análisis y la que

se usa en el diseño.

* No hay etapa de transición al diseño.
* Es posible intercalar tareas del Análisis Orientado a Objetos y del

Diseño Orientado a Objetos en el ciclo de desarrollo de la aplicación.

Sin embargo, análisis y diseño usan técnicas diferentes. Mientras que

el AOO utiliza técnicas que ayudan a identificar y definir clases y

objetos del dominio del problema, el diseño orientado al objeto

emplea técnicas que ayudan a identificar y definir clases y objetos

que reflejen la implementación de requerimientos.

* La representación es uniforme desde el AOO hasta el DOO y la

POO, constituyendo un marco de trabajo para la comprensibilidad,

reusabilidad y extensibilidad.

## **El paradigma objetual**

Orientado a Objetos (OO) se ha convertido en un sinónimo de algo bueno,

moderno, etc., se aplica a múltiples cosas hoy en día, entre ellas a la que a nosotros nos

interesa.

Los beneficios prometidos por la OO son básicamente dos:

* Reusabilidad: Los nuevos sistemas OO pueden ser creados utilizando

otros Sistemas Orientados a Objetos ya existentes.

* Extensibilidad: Los nuevos Sistemas Orientados a Objetos así obtenidos son

fácilmente ampliables sin tener que retocar los módulos, empleados en su

construcción. El término OO debe entenderse como algo general, y que está

íntimamente ligado a las etapas de Análisis y Diseño.

## **HISTORIA Y MITOS**

En los Sistemas Orientados a Objetos tradicionalmente, primero, se hablaba sólo

del tema de Programación y posteriormente fueron tomando importancia los de Análisis

y Diseño.

La Programación orientada a objetos (POO) comienza en 1967 con Simula'67, y

continua en los 70's con el desarrollo de Smalltalk.

La clave del comienzo de la POO la tuvo el hecho de querer simular modelos de la

realidad, lo cual era, y es, bastante complicado utilizando lenguajes típicos de 3ª

generación, básicamente porque no permite un fácil y buen modelado de los objetos del

mundo real, así como de la manera de interactuar entre ellos.

En POO los mensajes reemplazan al concepto típico de función, y constituyen el

diálogo entre objetos, lo que, a fin de cuentas, es la ejecución de un programa.

Los mensajes constituyen la comunicación entre los objetos y el término OO lo

introdujo Smalltalk, el cual estaba influido por Simula y el trabajo de tesis doctoral de

Alan Kay (máquina Flex). En Xerox esta máquina se llamó DynaBook y estaba influida

por los conceptos de clase y herencia de Simula además de por determinadas

características del lenguaje funcional Lisp.

Los primeros Lenguajes orientados a objeto (LOO) se caracterizan por tener el

problema típico de la eficiencia, es por eso, que muchos de los LOO de hoy en día son

evoluciones Orientadas a Objetos de lenguajes de tercera generación, p.e.: Object-

Pascal, Objetive-C, C-Talk, C++, Turbo-Pascal, etc., o simplemente nuevos lenguajes,

pero muy parecidos a los "normales" de 3ª generación.

Es interesante destacar, que conforme pasa el tiempo, y más se asienta el uso de la

POO el foco de atención sobre temas relacionados con lo "Orientado a Objetos" se ha

ido trasladando del concepto de Programación a los de Diseño (DOO) y a los de Análisis (AOO).

## **MITOS DE LA POO. ETIQUETAS QUE NO SON CIERTAS**

Acompañando al concepto de POO siempre vamos a encontrar asociados lo que

vamos a denominar una serie de mitos o etiquetas que no son ciertas.

Vamos a ver a continuación algunos de ellos:

El primero de estos mitos es el de que "todos los programas construidos

utilizando técnicas de POO son automáticamente reutilizables y extensibles".

Esto es falso, si no programamos correctamente caeremos en los mismos errores

que utilizando programación procedural normal. Sí que es cierto que un buen

Análisis y Diseño previos facilitan enormemente la labor del programador, ya

que la POO ofrece una forma más natural de expresar o modelar los problemas

del mundo real en un computador.

Otro mito que tampoco es cierto completamente es el de que la POO rompe

completamente con la programación procedural, no sólo no es cierto, sino que

además de valerse de ella la extiende con los nuevos conceptos que añade.

Otra idea a eliminar es la de que todo LOO necesita de una Librería de Clases a

partir de la cual derivar cualquier clase nueva para trabajar con él, Smalltalk.

También es falso lo que alguna vez se ha dicho como que un LOO no puede

obtenerse como evolución de uno no OO, sólo puede ser LOO uno nuevo

diseñado para ello. La mayoría de LOO de hoy en día son evoluciones de

lenguajes de 3ª generación corrientes, tales como Pascal, C, etc.

Otro aspecto a desmitificar en programas realizados con técnicas de POO es el

de que estos no presentan una ejecución ordenada y clara. La ejecución de un

programa realizado utilizando POO consiste en el diálogo que se establece entre

los objetos que conforman el problema que se trata de resolver. Este diálogo se

realiza mediante el paso de mensajes entre ellos.

## **Beneficios de la M.O.O.**

* Mejora la seguridad de los sistemas al ocultar la información de los objetos.
* El análisis orientado a objetos abarca los modelos centrados en datos.
* Métodos formales más claros en MOO
* OO es una herramienta para manejar la complejidad
* Evolución y mantenimiento de los sistemas más fácil.
* MOO tiene mayor potencial para captar mayor significado en sus aplicaciones GUI, WFS, DSD

## **Conclusión**

Con la llegada del paradigma de programación orientada a objetos se empezó a adaptarlo en muchos ámbitos como en el diseño pasando hacer una metodología que pasa todos los conceptos de programación orientada a objetos a una metodología.

La metodología orientada a objetos es un aporte grande ya que se puede entender muy fácilmente, trayéndonos muchos beneficios.

## **Bibliografía**

* <https://www.academia.edu/15943563/Que_es_la_Metodolog%C3%ADa_orientada_a_objetos_-_Introducci%C3%B3n_a_UML>
* <https://slideplayer.es/slide/94222/>
* <https://prezi.com/pz3tbghssygd/13-metodos-de-desarrollo-de-software-orientado-a-objetos/>
* <https://es.wikipedia.org/wiki/An%C3%A1lisis_y_dise%C3%B1o_orientado_a_objetos>