***Metodologías de Gestión de Proyectos de Desarrollo de Software***

*Salgado Landa Cristian Gael*

*Edith Nancy Báez Perez*

*Macroeconomía*

*Secuencia 2GM2*

*Licenciatura en Negocios Digitales*

*Instituto Politécnico Nacional*

Una metodología consiste en la sistematización de los pasos a través de los cuales se ejecutará un proyecto.

Es un sistema de métodos que utilizados en conjunto buscan obtener un resultado. Es decir, a un proyecto se le puede aplicar más de un método o una empresa puede definir su metodología o paso a paso para ejecutar sus proyectos utilizando diversos métodos.

Una metodología ofrece un lenguaje común que permite:

* Aplicar mejores prácticas.
* Mejorar la comunicación.
* Estandarizar las actividades a realizar en las fases del ciclo de vida del proyecto.
* Proporcionar herramientas para la ejecución del proyecto y para la toma de decisiones.
* Entre otros aspectos.

Se puede indicar que una Metodología de Gestión de Proyectos es una sistematización que facilita la ejecución de los proyectos.

**Programación Estructurada**

La programación estructurada es una teoría orientada a mejorar la claridad, calidad y tiempo de desarrollo utilizando únicamente subrutinas o funciones. Basada en el teorema del programa estructurado propuesto por Böhm y Jacopini, ha permitido desarrollar software de fácil comprensión.

La programación estructurada es una corriente que nació con la vocación de facilitar la vida de los programadores, sobre todo cuando estos debían abordar fases de mejora posteriores a la creación del programa, y de ordenar la forma en la que se creaba cualquier tipo de programa.

En el año 1966, cuando *Böhm* y *Jacopini* proponen el teorema del programa estructurado, con el que demuestran que cualquier programa puede ser escrito utilizando solo tres instrucciones de control; implicaba la construcción de programas más sencillos y más rápidos, en los que disminuía la complejidad de las pruebas y el testing para ponerlos en funcionamiento.

En 1968, *Edsger Dijkstra* publicó un célebre artículo que impactó en la computación moderna: ***Go To Statement Considered Harmful***. ¿Por qué es tan importante? Pues porque este científico holandés promovió activamente el uso de lenguajes de programación estructurada, fomentando la verificación formal de programas y la eliminación de la sentencia Goto. De hecho, *Dijkstra* participó en el comité que diseñó Algol 60, el primer lenguaje de programación estructurado.

La programación estructurada se convierte así, junto con la programación orientada a objetos, en **uno de los paradigmas de programación más populares** que ejecuta los lenguajes más potentes, incluidos, entre otros, Java, C, Python y C++.

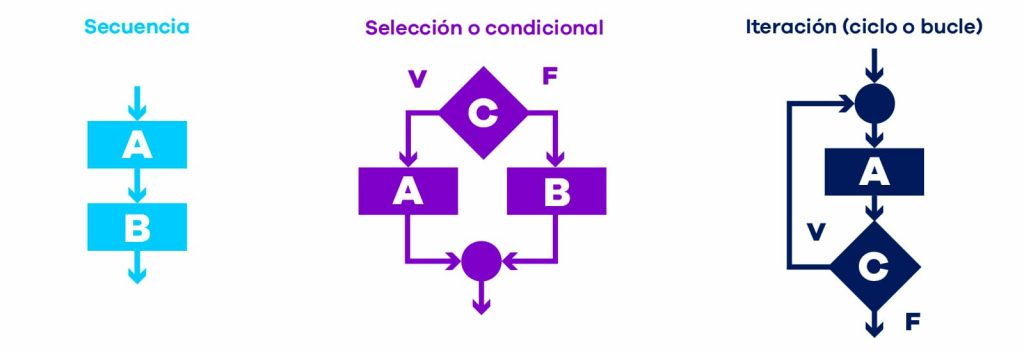
El **teorema del programa estructurado** es la base teórica sobre la que se construyó esta nueva forma de programar, ya que nos da la característica fundamental de la programación estructurada. Postula que, **simplemente con la combinación de tres estructuras básicas, es suficiente para expresar cualquier función computable.**

Los Digital Workers de los años 70 empezaban a ver poco a poco las grandes ventajas que ofrecía la programación estructurada sobre el *código espagueti* (los programas de computación antiguos que tenían una estructura de control de flujo compleja e incomprensible). Y, ¿qué ventajas eran esas?

* **Los programas desarrollados con la programación estructurada son más sencillos de entender,** ya que tienen una estructura secuencial y desaparece la necesidad de rastrear los complejos saltos de líneas (propios de la sentencia Goto) dentro de los bloques de código para intentar comprender la lógica interna.
* Como consecuencia inmediata de lo anterior, otra ventaja es que **los programas resultantes tendrán una estructura clara,** gracias a que las sentencias están ligadas y relacionadas entre sí.
* **La fase de prueba y depuración de los programas se optimiza,** ya que es mucho más sencillo hacer el seguimiento de los fallos y errores y, por tanto, detectarlos y corregirlos.
* **El coste del mantenimiento de los programas que usan la programación estructurada es más reducido.**¿Por qué? Pues porque modificar o extender los programas es más fácil al estar formados por una estructura secuencial.
* Al ser más sencillos los programas, **son más rápidos de crear** y los programadores aumentan su rendimiento.

Ya nos ha quedado claro que la programación estructurada es una forma de programar más sencilla que se basa únicamente en la combinación de tres órdenes. Pero, ¿cuáles son esos tipos de estructuras de control que son capaces de expresarlo todo?

1. **Secuencia.**La estructura secuencial es la que se da de forma natural en el lenguaje, porque las sentencias se ejecutan en el orden en el que aparecen en el programa, es decir, una detrás de la otra.
2. **Selección o condicional.** La estructura condicional se basa en que una sentencia se ejecuta según el valor que se le atribuye a una variable booleana. ¡Un pequeño inciso! Una variable booleana es aquella que tiene dos valores posibles. Por tanto, esta estructura se puede ejecutar de dos formas distintas, dependiendo del valor que tenga su variable.  
   Como apunte para los verdaderos amantes de la programación: para las estructuras condicionales o de selección, Python dispone de la **sentencia *if***, que puede combinarse con ***elif***y/o ***else***.
3. **Iteración (ciclo o bucle).** La estructura de repetición ejecuta una o un conjunto de sentencias siempre que una variable booleana sea verdadera. Para los bucles o iteraciones, los lenguajes de programación usan las estructuras ***while*** y ***for***.



**Programación Orientada a Objetivos**

La Programación Orientada a Objetos (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

A lo largo de la historia, han ido apareciendo diferentes paradigmas de programación. Lenguajes secuenciales como COBOL o procedimentales como **Basic o C**, se centraban más en la lógica que en los datos. Otros más modernos como **Java, C# y Python**, utilizan paradigmas para definir los programas, siendo la Programación Orientada a Objetos la más popular.

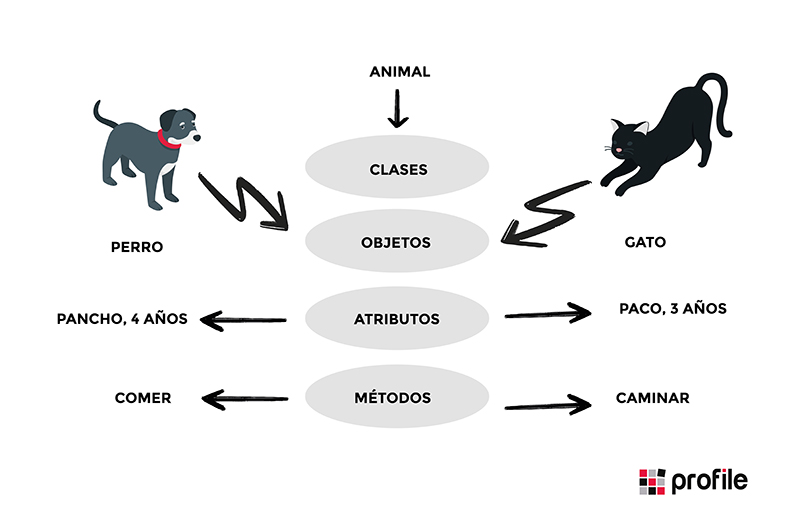
Con el paradigma de **Programación Orientado a Objetos** lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de este paradigma. Esto nos ayuda muchísimo en sistemas grandes, ya que, en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

Un programador diseña un programa de software organizando piezas de información y comportamientos relacionados en una plantilla llamada clase. Luego, se crean objetos individuales a partir de la plantilla de clase. Todo el programa de software se ejecuta haciendo que varios objetos interactúen entre sí para crear un programa más grande.

La Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener. Sigue el principio de desarrollo de software utilizado por muchos programadores **DRY (Don’t Repeat Yourself)**, para **evitar duplicar el código** y crear de esta manera programas eficientes. Además, evita el acceso no deseado a los datos o la exposición de código propietario mediante la encapsulación y la abstracción,

*El primer y más importante concepto de la POO es la****distinción entre clase y objeto****.*

Una **clase**es una plantilla. Define de manera genérica cómo van a ser los objetos de un determinado tipo. Por ejemplo, una clase para representar a animales puede llamarse ‘animal’ y tener una serie de **atributos**, como ‘nombre’ o ‘edad’ (que normalmente son propiedades), y una serie con los comportamientos que estos pueden tener, como caminar o comer, y que a su vez se implementan como métodos de la clase (funciones).  
  
Un **ejemplo sencillo de un objeto,**como decíamos antes, podría ser un animal. Un animal tiene una edad, por lo que creamos un nuevo atributo de ‘edad’ y, además, puede envejecer, por lo que definimos un nuevo método. Datos y lógica. Esto es lo que se define en muchos programas como la definición de una clase, que es la definición global y genérica de muchos objetos.



**Con la clase se pueden crear instancias de un objeto**, cada uno de ellos con sus atributos definidos de forma independiente. Con esto podríamos crear un gato llamado *Paco*, con 3 años de edad, y otro animal, este tipo perro y llamado *Pancho*, con una de edad de 4 años. Los dos están **definidos por la clase animal**, pero son dos instancias distintas. Por lo tanto, llamar a sus métodos puede tener resultados diferentes. Los dos comparten la lógica, pero cada uno tiene su estado de forma independiente.  
  
Todo esto, junto con los principios que vamos a ver a continuación, son herramientas que nos pueden ayudar a escribir un código mejor, más limpio y reutilizable.

**4 Principios de la Programación Orientada a Objetos**

* **La encapsulación**

La encapsulación contiene**toda la información importante de un objeto dentro del mismo** y solo expone la información seleccionada al mundo exterior.   
Esta propiedad permite asegurar que la información de un objeto esté oculta para el mundo exterior, agrupando en una Clase las características o atributos que cuentan con un acceso privado, y los comportamientos o métodos que presentan un acceso público.

* **La abstracción**

La abstracción es cuando **el usuario interactúa solo con los atributos y métodos seleccionados de un objeto**, utilizando herramientas simplificadas de alto nivel para acceder a un objeto complejo. En la programación orientada a objetos, los programas suelen ser muy grandes y los objetos se comunican mucho entre sí. El concepto de abstracción **facilita el mantenimiento de un código de gran tamaño**, donde a lo largo del tiempo pueden surgir diferentes cambios.

Así, la abstracción se basa en usar **cosas simples para representar la complejidad**. Los objetos y las clases representan código subyacente, ocultando los detalles complejos al usuario. Por consiguiente, supone una extensión de la encapsulación.

* **La herencia**

La herencia define **relaciones jerárquicas entre clases**, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias. A través de la definición en una clase de los atributos y comportamientos básicos, se pueden crear clases secundarias, ampliando así la funcionalidad de la clase principal y agregando atributos y comportamientos adicionales.

* **El polimorfismo**

El polimorfismo consiste en**diseñar objetos para compartir comportamientos**, lo que nos permite procesar objetos de diferentes maneras. Es la capacidad de presentar la misma interfaz para diferentes formas subyacentes o tipos de datos. Al utilizar la herencia, los objetos pueden anular los comportamientos principales compartidos, con comportamientos secundarios específicos.

El polimorfismo permite que el mismo método ejecute diferentes comportamientos de dos formas: anulación de método y sobrecarga de método.

Alrededor de estos principios de la programación orientada a objetos se construyen muchas cosas. Por ejemplo, los Principios SOLID, o los Patrones de diseño, que son recetas que se aplican a problemas recurrentes que se han encontrado y se repiten en varios proyectos.

**Beneficios de Programación Orientada a Objetos**

* **Reutilización** del código.
* Convierte cosas complejas en **estructuras simples reproducibles**.
* Evita la **duplicación de código**.
* Permite **trabajar en equipo** gracias al encapsulamiento ya que minimiza la posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.
* Al estar la clase bien estructurada permite la **corrección de errores** en varios lugares del código.
* **Protege la información** a través de la encapsulación, ya que solo se puede acceder a los datos del objeto a través de propiedades y métodos privados.
* La abstracción nos permite **construir sistemas más complejos** y de una forma más sencilla y organizada.

**Bibliografía:**

Canelo, M. M. (2021, 24 diciembre). *¿Qué es la Programación Orientada a Objetos?* Profile Software Services. https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/

López T., J. (2021, 2 agosto). *Metodologías de Gestión de Proyectos* [Vídeo]. OPM Integral. https://opmintegral.com/gestion-de-proyectos/metodologias-de-gestion-de-proyectos/