Paradigmas de programación

Programming Paradigms

Johnnatan Henao Aguirre

*Universidad tecnológica de Pereira, Risaralda, Colombia.*

Correo-e: johenao@utp.edu.co

***Resumen*— Se presentarán los tipos de paradigmas de programación, enfocándose en los más utilizados en el momento, su utilidad en los diferentes ámbitos, subrayando los procesos para los cuáles escoger uno en específico es más óptimo.**

**El mayor foco de interés es la funcionalidad en diferentes casos, ventajas y desventajas**

***Abstract*— in this work the different paradigms of programming will be presented. It is about understanding first of all, what is a paradigms and what is its definition in this scientific field. We also focus a lot on what kind of programming paradigms exist today, and the most used. Its meaning, influence and why, add that we will make known its advantages and disadvantages through clear and concise examples.**

***Palabras clave—* programación, imperativo, paradigmas, lógica, multi-paradigma.**

***Key Word* — programming, imperative, paradigms, logic, multi-paradigm.**

1. INTRODUCCIÓN

¿QUÉ ES UN PARADIGMA DE PROGRAMACIÓN?

Paradigma de programación es una propuesta tecnológica que es adoptada por una Comunidad de Programadores cuyo núcleo central es incuestionable en cuanto a que unívocamente trata de resolver uno o varios problemas claramente delimitados. La resolución de estos problemas debe suponer consecuentemente un avance significativo en al menos un parámetro que afecte a la ingeniería de Software. Tiene una estrecha relación con la formalización de determinados lenguajes en su momento de definición. Un paradigma de programación está delimitado en el tiempo en cuanto a aceptación y uso ya que nuevos paradigmas aportan nuevas o mejores soluciones que la sustituyen parcial o totalmente.

La introducción puede contener:

* Programación Imperativa
* Programación funcional
* Programación lógica
* Declarativo
* POO Orientado a Objetos
* Por procedimientos



Habitualmente se mezclan todos los tipos de paradigmas a la hora de hacer la programación. De esa manera se origina la programación **multiparadigma**, pero el que actualmente es más usado de todos esos paradigmas es el de la programación orientada a objetos.

1. CONTENIDO

En la evolución de la programación han surgido diversas técnicas de programación que se han ido adaptando a las necesidades tecnológicas e informáticas del momento. Aunque la forma de enfocar la elaboración de los programas es diferente en cada una de ellas, el objetivo es el mismo: facilitar la creación y el mantenimiento de programas informáticos. Estas técnicas se han traducido en diferentes filosofías de creación de programas que son los denominados paradigmas de programación.

Aunque hay muchos paradigmas de programación, los más comunes son:

**Paradigma imperativo**

Los programas imperativos contienen instrucciones que dicen al ordenador cómo realizar una tarea. Los primeros lenguajes imperativos fueron los códigos máquina de los ordenadores, que utilizaban instrucciones sencillas y permitían implementar el hardware fácilmente, pero no servían para desarrollar programas complejos.

El primer lenguaje imperativo que posibilitó la creación de programas con un nivel de complejidad elevado fue FORTRAN. Hoy en día está representado por los lenguajes de programación BASIC, C ó PASCAL, entre otros.

**Paradigma funcional**

Los programas funcionales se basan en el uso de una o más funciones dentro de las cuales se pueden utilizar funciones creadas anteriormente. Su objetivo es dividir el programa en módulos de forma que cada uno de éstos realice una única función.

El primer lenguaje de programación funcional fue LISP. Existen dos tipos de lenguajes funcionales: los puros (como HASKELL) y los híbridos (SAP, ML, Scheme).

**Características**

Los programas escritos en un lenguaje funcional están constituidos únicamente por definiciones de funciones

La no existencia de asignaciones de variables y la falta de construcciones estructuradas como la secuencia o la iteración

Existen dos grandes categorías de lenguajes funcionales: los funcionales puros y los híbridos

En contraste, los lenguajes funcionales puros tienen una mayor potencia expresiva, conservando a la vez su transparencia referencial.

**Ventajas**

Ausencia de efectos colaterales

Proceso de depuración menos problemático

Pruebas de unidades más confiables

Mayor facilidad para la ejecución concurrente

**Desventajas:**

Falta de estandarización

Bajo rendimiento de los programas

**Paradigma lógico**

La programación lógica comprende la programación declarativa y la funcional. El proceso de elaboración de programas está basado en la lógica de primer orden y, a diferencia de los demás paradigmas, especifica qué debe hacer el programa y no cómo hacerlo.

Se emplea en aplicaciones de inteligencia artificial. El lenguaje de programación lógica por excelencia es PROLOG.

Paradigma orientado a objetos (POO)

La programación orientada a objetos expresa un programa como un conjunto de objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener, reutilizar y volver a utilizar. Su uso se popularizó a principios de los 90 y actualmente son muchos los lenguajes de programación asociados a este paradigma.

Las características del paradigma orientado a objetos son: encapsulamiento, abstracción, polimorfismo y herencia.

Muchos lenguajes utilizados en la actualidad están orientados a objetos, como Java, C++, Python o Delphi. Un lenguaje completamente orientado a objetos es Smalltalk.

**Programación estructurada**

**Características**

La estructura de los programas es clara, puesto que las instrucciones están más ligadas o relacionadas entre sí.

Los programas son más fáciles de entender, pueden ser leídos de forma secuencial.

Un programa escrito de acuerdo a los principios de programación estructurada no solamente tendrá una mejor estructura sino también una excelente presentación.

**Ventajas**

Reducción de los costos de mantenimiento

Reducción del esfuerzo en las pruebas y depuración

Los bloques de código son casi auto-explicativos, lo que reduce y facilita la documentación.

Se incrementa el rendimiento de los programadores

**Desventajas**

El principal inconveniente de este método de programación es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático el manejo de su código fuente

Se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar difícil su manejo

**Programación orientada a objetos**

**Características**

Abstracción: denota las características esenciales de un objeto, donde se capturan sus comportamientos

Polimorfismo: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

**Ventajas:**

Permite crear sistemas más complejos

Agiliza el desarrollo de software

Proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible.

Fomenta la reutilización y extensión del código.

**Desventajas:**

Complejidad para adaptarse Mayor cantidad de código

1. CONCLUSIONES

Como dictamen final se afirma que cada paradigma tiene un campo en específico donde actuar para la programación, esto indica que el uso de cada uno es según la conveniencia del proyecto, creando así ventajas y desventajas según donde se planee trabajar con cada cual, por lo tanto es importante un previo análisis para el trabajo optimo con cada uno de estos

RECOMENDACIONES

Plantear a qué se quiere llegar con el proyecto, comprender cada paradigma para encontrar mayor afinidad al tipo de problema en el que se busca la solución para tener un mayor nivel de eficacia.

REFERENCIAS

<http://wiki.uqbar.org/wiki/articles/paradigma-logico.html>

<https://www.4rsoluciones.com/blog/que-son-los-paradigmas-de-programacion-2/>

<https://www.monografias.com/trabajos107/paradigmas-programacion/paradigmas-programacion.shtml>

<https://www.infor.uva.es/~cvaca/asigs/docpar/intro.pdf>