Comprendiendo el rol del diseño orientado a objetos en el Desarrollo de Software

Por Ing. Willis Polanco

Es Ingeniero en Sistemas y Computación con concentración en Desarrollo de Software egresado de PUCMM. Tiene un Master en Business Administration de PUCMM y un Master in Science of Information System del Stevens Institute of Technology. El Ing. Polanco es un emprendedor y educador, ha sido catedrático de reconocidas instituciones académicas tanto local como en el extranjero y colaborador de múltiples comunidades como la ACM, IOI, ANDROID, SCRUM, Cisco NetAcad, entre otras, con enfoque al desarrollo profesional en el marco tecnológico. Ha sido reconocido en varias ocasiones con premios locales e internacionales, y como ponente internacional. Se ha desempeñado como Ingeniero de Infrastructura en Mercasid, Encargado del área de Administración de Servidores en Edesur, Sub-Director de Tecnología en la Presidencia, Director Académico en ITLA y Consultor de TIC con experiencia comprobada.

Probablemente en las últimas décadas nos hemos encontrado con el término **Orientado a Objetos**. Si profundizamos un poco más sobre este término podemos notar que también suele colocarse como si se tratara de un apellido de las diferentes responsabilidades dentro de un proyecto de software:

- Análisis **Orientado a Objetos**
- Diseño **Orientado a Objetos**
- Programación **Orientada a Objetos**

Ahora bien, cada una de estas responsabilidades dentro del proceso de construir un software se enfoca en un objetivo muy específico e importante para alcanzar la meta. De igual manera,

podemos decir que cada responsabilidad puede recaer sobre personas distintas, o todas las fases pueden recaer sobre una misma persona.

El **análisis orientado a objetos** persigue comprender el problema, y entonces llegar a la mejor solución posible al problema identificado. El **diseño orientado a objetos** persigue formular un plan a través de un diseño conceptual que explique el cómo se implementará la solución encontrada. La **programación orientada a objetos** persigue construir el software mediante herramientas de codificación tomando como punto de partida la correcta comprensión del problema y siguiendo el plano conceptual diseñado para la solución.

Sin embargo, esto no explica por qué es tan importante emplear el **enfoque de Orientación a Objetos** en miras a cumplir dichas responsabilidades críticas, las cuales son parte del proceso de construir un software. Si bien es cierto, podemos continuar con nuestro enfoque tradicional de construir un software a través de las estructuras fundamentales para programar una secuencia de instrucciones, un flujo de decisión y/o un flujo de instrucciones repetitivas.

Ese enfoque tradicional, aunque inflexible a cambios por las enormes cantidades de estructuras de código que se producen sin capacidad de mucho rediseño, generan resultados concretos si se trata de necesidades bien definidas que probablemente no cambien mucho con el pasar del tiempo.

Aunque la finalidad de todas las actividades vinculadas al desarrollo de un producto de software es poder traducir las necesidades de los usuarios a funcionalidades del producto, no menos cierto es que un software es un producto dinámico que cambia constantemente porque las necesidades de los usuarios también cambian.

Por lo tanto, podemos inferir que un software debe ser flexible, y debe permitir la fácil administración de sus funcionalidades, ya sea facilitando la forma en que se agregan, modifican o eliminan funcionalidades en el producto de software.

De ahí podemos deducir que el enfoque tradicional no es una técnica eficiente a la hora de producir un software que se adapta al negocio, y evitar lo que sucede en muchos casos, donde el negocio tiene que adaptarse a las limitaciones del software.

El enfoque de construcción de software en base a un modelo de objetos, promete ser una técnica mucho más eficiente a la hora de rediseñar las funcionalidades, o adaptarse a los constantes cambios que inevitablemente impactarán al ciclo de vida del software.

Las grandes preguntas que quedan expuestas son ¿Qué exactamente significa un **enfoque Orientado a Objetos**? ¿Por qué pensar en función de objetos es una técnica más eficiente para el análisis, diseño y programación de software?

Para entender la respuesta, vamos a poner en perspectiva el siguiente escenario. Imaginemos que tenemos un software realizado bajo el enfoque tradicional y se solicita un cambio. La solicitud sería algo muy parecido a esto:

> "Solicitar al desarrollador cambiar las líneas 3,450, 9,850 y 16,530 donde se encuentra el código afectado por el pago de facturas con saldo pendiente del módulo de facturación. Tener en cuenta el impacto de cambiar estas líneas, ya que puede afectar otra porción de código del software."

Por otro lado, imaginemos que tenemos el mismo software realizado bajo el enfoque de **Orientación a Objetos** y se solicita un cambio. La solicitud sería algo muy parecido a esto:

> "Solicitar al desarrollador cambiar el comportamiento de pago de facturas con saldo pendiente dentro del objeto Factura."

Si analizamos ambos escenarios, podemos rápidamente resaltar la facilidad del cambio de funcionalidades, si en vez de pensar en estructuras de códigos, pensamos en objetos.

Los **objetos** nos ayudan a entender la forma en que cada proceso dentro del software revela los datos que maneja y las operaciones que éste puede ejecutar. Cada objeto contiene en sí mismo los datos y la lógica para producir el resultado esperado.

Esa interpretación de los objetos, es la misma que podemos dar a los objetos fuera de un contexto informático. En un contexto real los objetos los percibimos conforme las características que poseen y los que estos son capaces de hacer. Por eso, resulta más eficiente analizar, diseñar y programar un software usando la misma mecánica de interpretación de los objetos que usamos en nuestra realidad.

De manera que la construcción de software desde el enfoque de **orientación a objetos**, naturaliza la forma en que entendemos cada fase del proceso de creación de un software. Esto nos conduce a simplificar el modo en que podemos comprender un problema, ayuda a agilizar la formulación de un plano conceptual en torno a objetos y sus interacciones, y permite construir el producto mediante herramientas habilitadas para codificar objetos.