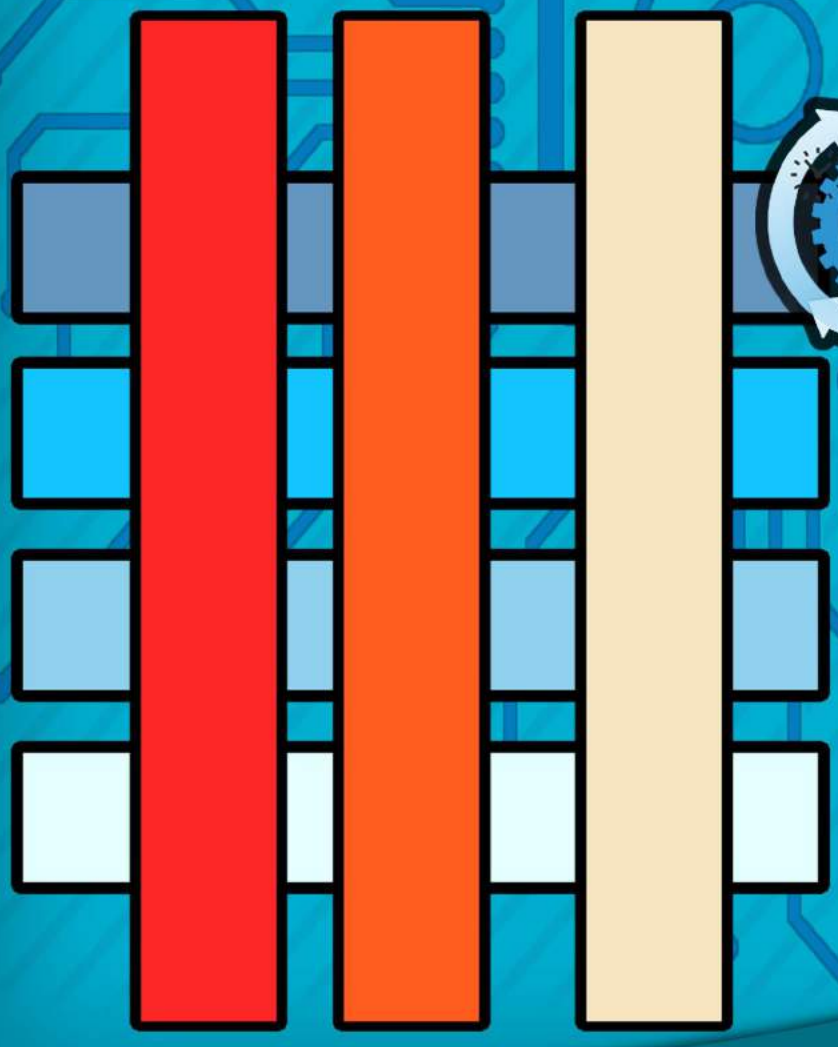
Programación Orientada a Aspectos

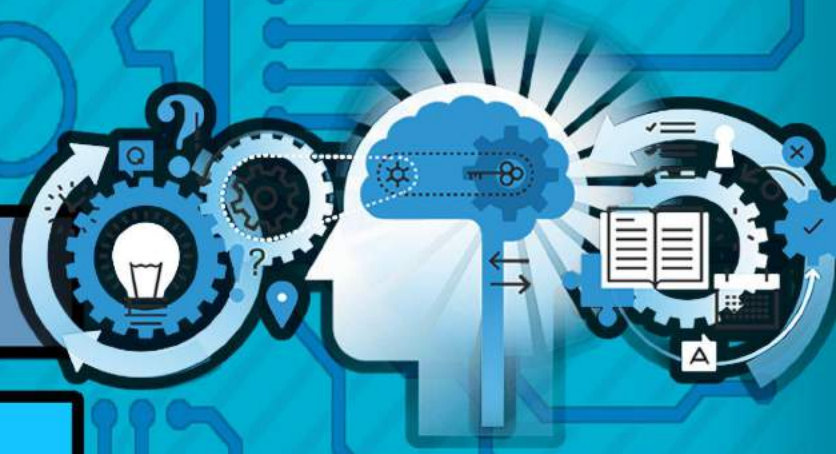
POA

Universidad Luterana Salvadoreña

Análisis Estructurado

Lic. José Luis Alvarado





RIDL AspectS

ULS

Presentado por:

Elmer Alexander Granados Maradiaga

Estudiante de la Licenciatura en Ciencias de la Computación, en la Universidad Luterana Salvadoreña desde el año 2012, actualmente cursa su décimo ciclo de estudio.
Nació el 9 de junio de 1989, en la Ciudad de San Miguel.
Sus intereses: la informática, programación y deportes.

Carlos Efraín Antonio Rivera Sánchez

Nació el 29 de julio de 1984 en Zacatecoluca, La Paz, cursó su educación básica en el Colegio El Espíritu Santo, y bachillerato opción General en el Instituto Nacional José Simeón Cañas, ambos de la ciudad de Zacatecoluca. Actualmente estudiante de la carrera Licenciatura en Ciencias de la Computación a nivel octavo Ciclo en la Universidad Luterana Salvadoreña, labora como Coordinador del Aula de Informática en el Centro Escolar San Luis la Herradura.
Intereses: Robótica, informática, redes informática, diseño web

Yorleni Verenice Palma Ascencio

Nació el 13 de febrero de 1986
Actualmente es estudiante de la carrera de Ciencias de la Computación, en la Universidad Luterana Salvadoreña.

Introducción

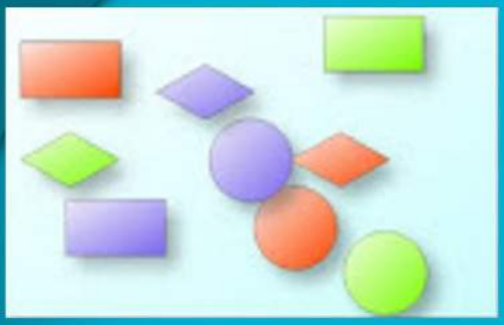
La Programación Orientada a Aspectos pretende dar solución a las tareas o eventos repetitivos que pueden, en un momento dado, retrasar la construcción de un software.

La Programación Orientada a Aspectos constituye uno los avances más importantes de los últimos años en la ingeniería del software para edificar sistemas complejos empleando el principio de descomposición, ya que el modelo de objetos subyacente se ajusta mejor a los problemas del dominio real que la descomposición funcional.

La ventaja que tiene es que es fácil, la integración de nuevos datos, aunque también quedan las funciones esparcidas por todo el código, y tiene los inconvenientes de que, con frecuencia, para realizar la integración de nuevas funciones hay que modificar varios objetos, y de que se produce un enmarañamiento de los objetos en funciones de alto nivel que involucran a varias clases.

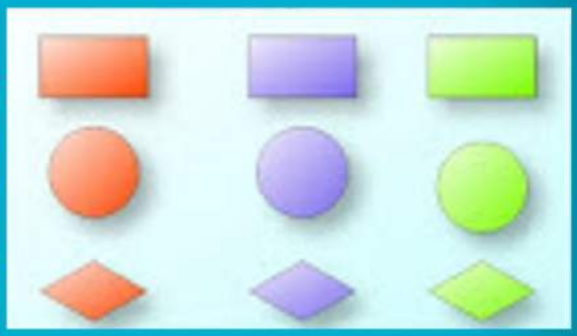
La POA es una nueva metodología de Programación que aspira a soportar la separación de las propiedades para los Aspectos antes mencionados. Esto implica separar la funcionalidad básica y los aspectos, y los Aspectos entre sí, a través de mecanismos que permitan abstraerlos y componerlos para formar todo el sistema.

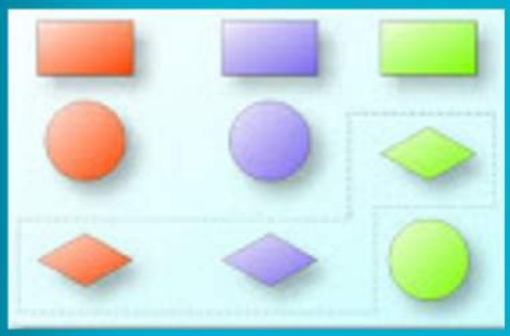
¿Qué es? POA



La Programación Orientada a Aspectos se ha planteado como un nuevo paradigma de Programación, el cual consiste en separar los conceptos que entrecruzan varias clases y se extienden a lo largo de éstas, pero que no pertenecen a esas clase en sí mismas.

Incluso, los conceptos pueden aparecer no sólo en varias clases, sino de una sola. Ahora la pregunta: ¿de qué tipo de conceptos hablamos? Pensemos como ejemplo, en una aplicación que requiere del registro (logging) de las acciones que realiza. Bien, estos registros "alguien" tiene que llevarlos a cabo, y es normal cargar a muchas clases con la responsabilidad de hacerlo.





Cuando, seguramente, no sea la función principal de esas clases hacer dicha tarea. Como vemos, se trata de una misma funcionalidad (registrar acciones) que se encuentra entrecruzada en varias clases, pero que a la vez no es parte de ninguna.

Objetivos de POA

Separar las funcionalidades comunes utilizadas en toda la aplicación como: algunos eventos, validaciones, mensajes, etc.

Separar las funcionalidades propias de cada módulo que no hacen parte de las anteriores como por ejemplo en cálculo de alguna fórmula en un módulo especifico o lo que se podría llamar las reglas del negocio o la capa BL , por sus siglas en Ingles.

Características de POA

De la consecución de estos objetivos se pueden obtener las siguientes ventajas:

Un código menos enmarañado, más natural y más reducido.
Una mayor facilidad para razonar sobre las materias, ya que están separadas y tienen una dependencia mínima.

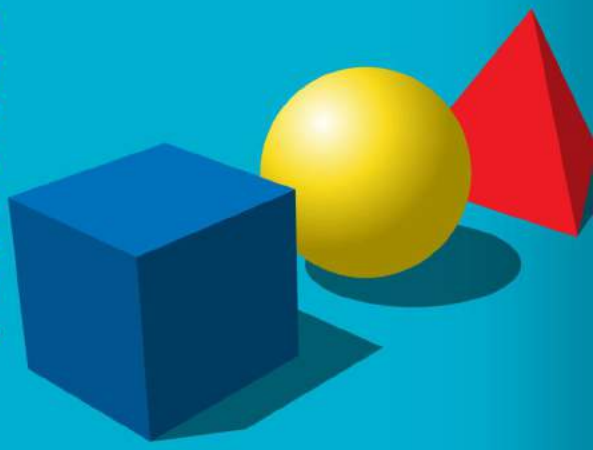
Más facilidad para depurar y hacer modificaciones en el código.
Se consigue que un conjunto grande de modificaciones en la definición de una materia tenga un impacto mínimo en las otras.

Se tiene un código más reusable y que se puede acoplar y desacoplar cuando sea necesario.

Lenguaje de programación AspectS

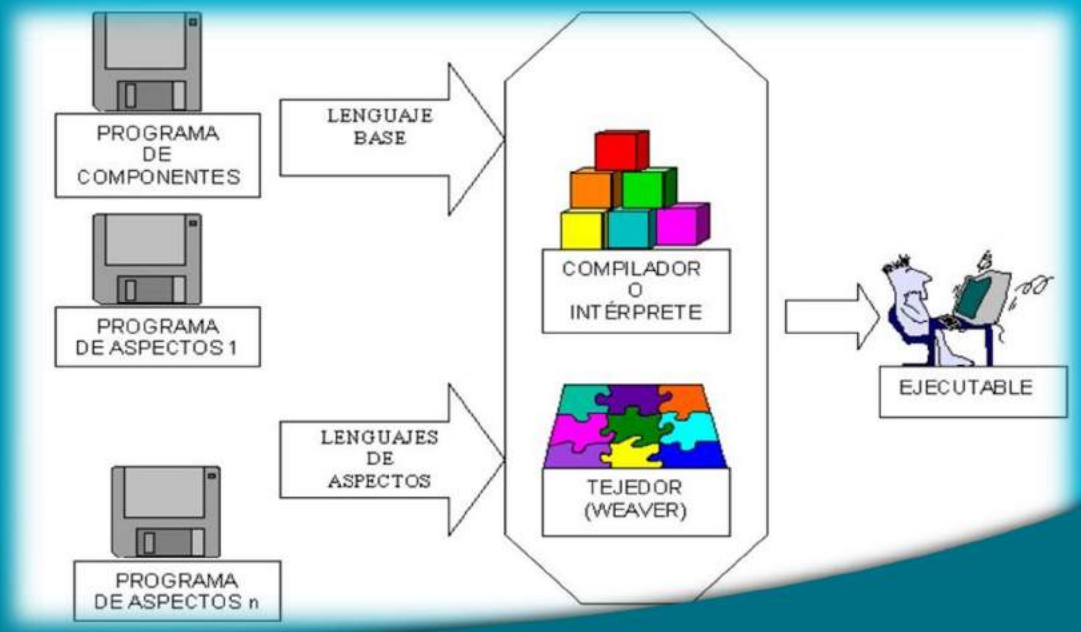
AspectS, un lenguaje de Aspectos de propósito general, utiliza el modelo de lenguaje de AspectJ y ayuda a descubrir la relación que hay entre los aspectos y los ambientes dinámicos. Soporta Programación en un metanivel, manejando el fenómeno de Código Mezclado a través de módulos de Aspectos relacionados. Está implementado en Squeak sin cambiar la sintaxis, ni la máquina virtual.

En este lenguaje los Aspectos se implementan a través de clases y sus instancias actúan como un objeto, respetando el principio de uniformidad. Un Aspecto puede contener un conjunto de receptores, enviadores o clases enviadoras. Estos objetos se agregan o se remueven por el cliente y serán usados por el proceso de tejer en ejecución para determinar si el comportamiento debe activarse o no.



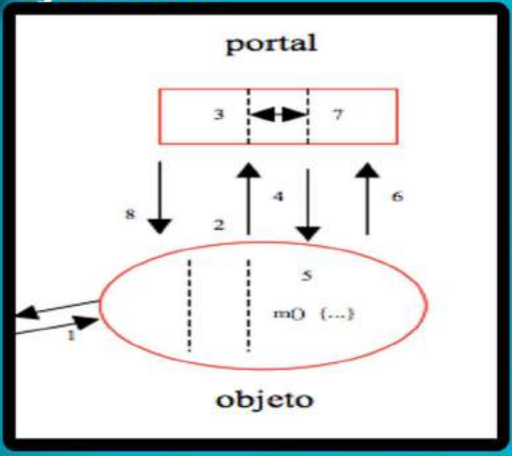
Los tipos de avisos definibles en AspectS son:

Antes y después de la invocación a un (AsBeforeAfterAdvice).
Para manejo de excepciones(AsHandlerAdvice).
Durante la invocación de un método(AsAroundAdvice).



Lenguaje de programación RIDL

RIDL es un lenguaje de Aspectos de dominio específico que maneja la transferencia de datos entre diferentes espacios de ejecución.



Un programa RIDL consiste de un conjunto de módulos de portales. Los módulos de portales o directamente portales se asocian con las clases por el nombre. Un portal es el encargado de manejar la interacción remota y la transferencia de datos de la clase asociada a él, y puede asociarse como máximo a una clase. La unidad mínima de interacción remota es el método.

La declaración de los portales identifica clases cuyas instancias pueden invocarse desde espacios remotos. Dichas instancias se llaman objetos remotos. La declaración de un portal identifica qué métodos de una clase serán exportados sobre la red. En el portal estos métodos se llaman operaciones remotas. Para cada una de estas operaciones se describe qué objetos remotos esperan y qué datos enviarán a los llamadores.

Los portales no son clases, al igual que los coordinadores en COOL, no pueden ser instanciados y sirven para un propósito muy específico. Tampoco son tipos en el sentido estricto de la palabra. Un portal se asocia automáticamente con una instancia de la clase en el momento que una referencia a esa instancia se exporta fuera del espacio donde la instancia fue creada. Durante el tiempo de vida de la instancia esta relación se mantiene mediante un protocolo bien definido.

Este último punto establece una dependencia explícita entre los portales y la relaciones estructurales completas de las clases. Esta dependencia expone la necesidad de controlar la transferencia de datos entre los distintos espacios de ejecución.

Conclusiones

Los lenguajes de Aspectos de dominio específico juegan un papel crucial en la Programación Orientada a Aspectos y que son preferibles a los lenguajes de Aspectos de propósito general, porque soportan mejor la separación de funcionalidades. Pero aún no se ha experimentado mucho en lo relacionado a programación Orientada a Aspectos.

La programación por Aspectos ofrece un conjunto de herramientas adicionales que permiten Modularizar software mediante varios criterios

La separación de los Aspectos a todos los niveles (diseño, codificación y ejecutable) es un gran paso que se ha efectuado, pero se debe refinar en cuestiones de eficiencia.