Patrones de Diseño: GRASP

Introducción a los Patrones GRASP

- GRASP (General Responsability Assignment Software Pattern) propone los patrones como una codificación de principios básicos ampliamente utilizados por los expertos en objetos.
- Un sistema OO se compone de objetos que envían mensajes a otros objetos para que lleven a cabo ciertas operaciones.
- Cada clase tiene ciertas responsabilidades, que son cumplidas a través de sus métodos, y por la forma en que colabora con otras clases.
- Los patrones GRASP, se basan principalmente en la asignación de responsabilidades en el diseño OO, fase donde nos encontramos con la tarea de crear las clases y las relaciones entre ellas.

Introducción a los Patrones GRASP

- Las responsabilidades se relacionan con las obligaciones de un objeto respecto de su comportamiento. Estas responsabilidades pertenecen a dos categorías: *conocer y hacer*.
- Entre las responsabilidades de un objeto relacionadas con el *hacer* se encuentran:
  - Hacer algo en uno mismo.
  - Iniciar una acción en otros objetos.
  - Controlar y coordinar actividades en otros objetos.
- Entre las responsabilidades de un objeto relacionadas con el *conocer* se encuentran:
  - Conocer los datos privados encapsulados.
  - Conocer la existencia de objetos conexos.
  - Tener información que se puede derivar o calcular.

Introducción a los Patrones GRASP

- Las responsabilidades se asignan a los objetos durante el diseño orientado a objetos.
  - Por ejemplo, podría decirse que una Venta es responsable de imprimirse ella misma (*un hacer*), o que una Venta tiene la obligación de conocer su fecha (*un conocer*).
- Responsabilidad no es lo mismo que método: los métodos se usan para cumplir con las responsabilidades. Éstas se implementan usando métodos que operen solos o en colaboración con otros métodos y objetos.
  - Por ejemplo, la clase Venta podría definir uno o varios métodos para imprimir una instancia de Venta. Para hacer esto, Venta puede colaborar con otros objetos, por ejemplo, enviando mensajes a LineadeVenta para que se impriman ellos mismos.

Patrones GRASP. Clasificación

1. Bajo Acoplamiento
2. Alta Cohesión
3. Experto en Información
4. Creador
5. Controlador
6. Polimorfismo
7. Fabricación Pura
8. Indirección
9. Variaciones Protegidas

C. Larman.
UML y Patrones:
Una introducción al Análisis y Diseño Orientado a Objetos
y al Proceso Unificado.
Ed. Prentice Hall.

Patrón Bajo Acoplamiento

- **Objetivo o Propósito:**
  - Propone que el nivel de acoplamiento entre los objetos sea bajo, entendiendo por acoplamiento el número de elementos (clases, subclases, sistemas, etc.) a los que un objeto está conectado a, tiene conocimiento de, confía en otros elementos.
- Si todas las clases dependen de todas las clases, ¿cuánto software podemos extraer y reutilizar en otro proyecto?.

Patrón Alta Cohesión

- **Objetivo o Propósito:**
  - Cada elemento debe realizar una labor única dentro del sistema, y no desempeñada por el resto de los elementos.
- Asigna una responsabilidad de manera que la cohesión permanezca alta. *Cohesión: medida de la fuerza con la que se relacionan los objetos o de la cantidad de trabajo que realizan.*
- De esta forma, una clase que tiene una cohesión baja, hace muchas cosas que no tienen relación entre sí; estas clases con baja cohesión suelen ser clases demasiado abstractas a las que se les han asignado demasiadas responsabilidades.

Patrón Experto

- **Objetivo o Propósito:**
  - Asigna la responsabilidad de realizar una tarea determinada, a aquel objeto que tiene la información (atributos) necesaria para ello. Este objeto expresa la "intuición" común de que los objetos hacen el trabajo relacionado con la información que tienen.
- A menudo nos encontramos con que la información necesaria para realizar una tarea o responsabilidad se encuentra dispersa en diferentes clases de objetos.
- Cada vez que la información se encuentre en varios objetos diferentes, necesitarán interactuar mediante el paso de mensajes para compartir el trabajo.

Patrón Creador

- **Objetivo o Propósito:**
  - Este patrón asigna a la clase B la responsabilidad de crear una instancia de la clase A (B es Creador de los objetos A) si:
    - B contiene objetos de A
    - B agrega objetos de A
    - B registra instancias de objetos de A
    - B tiene los datos de inicialización de A (datos que requiere su constructor)
    - B utiliza más estrechamente datos de A

Patrón Controlador

- **Objetivo o Propósito:**
  - El patrón Controlador se encarga de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas.
  - El objeto controlador no será el que realice estas actividades, sino que las delegará en otras clases con las que mantiene un modelo de alta cohesión.

Patrón Polimorfismo

- **Objetivo o Propósito:**
  - El patrón se basa en asignar la responsabilidad para el comportamiento, utilizando el polimorfismo, cuando las alternativas y comportamientos relacionados varíen según el tipo de clases.
  - No es conveniente implementar comportamientos alternativos con sentencia IF-ELSE, para hacer comprobaciones acerca del tipo del objeto, ya que lo único que se consigue es limitar la reutilización y el crecimiento del sistema.
  - Entendemos por polimorfismo, asignar el mismo nombre a servicios en diferentes objetos cuando los servicios son parecidos o están relacionados.

Patrón Fabricación Pura

- **Objetivo o Propósito:**
  - Este patrón desarrolla clases que se encargan de construir los objetos adecuados en cada momento (factorías).
  - Estas clases factorías son clases artificiales que no representan ningún concepto del dominio del problema, tienen un conjunto de responsabilidades altamente cohesivo y tienen un bajo acoplamiento, con lo que se consigue un diseño limpio y puro, de ahí el nombre de Fabricación Pura.

Patrón Indirección

- **Objetivo o Propósito:**
  - Este patrón se basa en la creación de clases intermedias para desacoplar componentes o servicios, o asigna la responsabilidad a un objeto intermedio que medie entre dos componentes o servicios de manera que no se acoplen directamente.
  - En los diseños OO se tiene en cuenta un viejo dicho que dice que:

    "la mayoría de los problemas en informática se pueden resolver mediante otro nivel de indirección"
  - Muchos de los patrones existentes son especificaciones del patrón Indirección.

Patrón Variaciones Protegidas

- **Objetivo o Propósito:**
  - Este patrón identifica los puntos de variaciones previstas o de inestabilidad, y asigna las responsabilidades para crear una interfaz estable alrededor de ellos.

- Proporciona flexibilidad a un sistema y protección frente a las variaciones. El patrón motiva la encapsulación de datos, interfaces, polimorfismo e indirección.
- Se aplica tanto a puntos de variación (variaciones en el sistema actual, existente o en los requisitos) como a puntos de evolución (variaciones que podrán aparecer en el futuro).