Universidad Autónoma de Nuevo león

Facultad Ciencias Fisco Matemático

DOO

LSTI

Patrones de diseño

Miguel Salazar

Edgar Vázquez Márquez

1657462 Aula: 413

Octubre de 2017

San Nicolas, Nuevo León

Patrones de diseño

¿Qué son? ¿Para qué sirven?

Los patrones de diseño son el principal esqueleto de las soluciones a problemas comunes en el momento del desarrollo de software. Estas nos brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Existen varios patrones de diseño popularmente conocidos, los cuales se clasifican como se muestra a continuación:

- Patrones Creacionales: Inicialización y configuración de objetos.
- Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

¿Cómo se documentan?

El uso de un patrón de diseño en un diseño concreto se documenta, fundamentalmente, señalando la relación que hay entre los elementos estructurales del patrón (según [GHJV03]) y los elementos estructurales del diseño. De esta forma se intenta capturar la semántica estructural de cada elemento del diseño concreto al ligarlo a un elemento del patrón de diseño pues en este se explica la semántica de cada termino estructural.

Estas relaciones se establecen por medio de asignaciones de la forma elemento patrón:= elemento diseño, donde elemento puede ser un módulo, subrutina o variable de estado. La documentación está completa si el conjunto de asignaciones incluye todos los elementos estructurales del patrón. Puede incluirse un comentario informal que explique la relación entre los elementos del patrón y los del diseño concreto. Además, es muy conveniente acompañar la documentación con una justificación en términos del análisis de cambio realizado.

Algunos formatos en la documentación de patrones, los más representativos son los siguientes:

Esquema del Patrón

Nombre del patrón: CreaciónGLO

Clasificación del patrón: Creacional

Contexto: Repositorio de RLO (CETL)

Problema: Crear objetos de aprendizaje personalizados, adaptados para un proceso de enseñanza aprendizaje específico.

Solución: Separar del RLO claramente la estructura superficial del objeto, es decir, el contenido de éste, del diseño del RLO, de esta manera se obtendrán objetos más complejos y potentes que partiendo de un diseño genérico pueden ser personalizados para su uso generalizado en distintos ámbitos.

A considerar ahora: SeparacionEstructurasRLO, DiseñoCoreGLO.

Esquema del Patrón II

A partir del esquema anterior se puede obtener un segundo patrón de diseño más específico y donde se especifique con más detalles los pasos a dar para la creación del GLO. Este patrón de diseño vendría definido como sigue:

Nombre del patrón: CreaciónGLO

Clasificación del patrón: Creacional

Contexto: Repositorio de RLO (CETL)

Problema: Crear objetos de aprendizaje personalizados y adaptados para un proceso de enseñanza aprendizaje específico.

Solución: Definir una estructura pedagógica común para todos los RLO que se creen basándose en el objeto GLO creado. La estructura pedagógica del GLO se puede dividir en tres grandes bloques:

U Orientación: donde se darán respuestas a cuestiones como la utilidad del GLO, índice, nombre del creador o institución a la que representa, etc. Cada una de estas cuestiones vendrá dispuesta en páginas diferentes.

o Desarrollo: donde se diseñarán las pantallas que serán visibles para el GLO final. Estas pantallas serán configurables en contenido pero no en diseño y forma.

o Cuestiones: donde se diseñarán las cuestiones con múltiples respuestas de las que consta el GLO. Estas cuestiones serán configurables para cada RLO que se cree a partir del GLO que se obtenga.

A considerar ahora: CreaciónRLOapartirdeGLO, PublicacionRLOenRepositorio

Descripción de 3 ejemplos

Para poder poner algunos ejemplos busque en algunas páginas pero no encontré alguno que respete lo que pide en sí, pero en la misma página que me ayude para escribir esta información había algunos ejemplos en que yo creo que la mayoría de mis compañeros pondrán los siguientes ejemplos:

Patrones Creacionales

Método de Fabricación

Parte del principio de que las subclases determinan la clase a implementar.

```
public class ConcreteCreator extends Creator
  {
  protected Product FactoryMethod()
      {
            return new ConcreteProduct();
      }
}
public interface Product{}
public class ConcreteProduct implements Product{}
      public class Client
      {
            public static void main(String args[])
            {
                  Creator UnCreator;
                  UnCreator = new ConcreteCreator();
                  UnCreator.AnOperations();
            }
      }
```

Prototipado

Singleton

Restringe la instanciación de una clase o valor de un tipo a un solo objeto.

```
      public sealed class Singleton
      {
            private static volatile Singleton instance;
            private static object syncRoot = new Object();
            private Singleton()
            {
                  System.Windows.Forms.MessageBox.Show("Nuevo
Singleton");
            }
            public static Singleton GetInstance
            {
                  get
                  {
                        if (instance == null)
                        {
                              lock(syncRoot)
                              {
                                    if (instance == null)
                                          instance = new Singleton();
                              }
                        }
                        return instance;
```

```
                }
            }
        }
```

Patrones de Comportamiento

```
Public Class Articulo
   Delegate Sub DelegadoCambiaPrecio(ByVal unPrecio As Object)
   Public Event CambiaPrecio As DelegadoCambiaPrecio
   Dim _cambiaPrecio As Object
   Public WriteOnly Property Precio()
      Set(ByVal value As Object)
         _cambiaPrecio = value
         RaiseEvent CambiaPrecio(_cambiaPrecio)
      End Set
   End Property
End Class
Public Class ArticuloObservador
   Public Sub Notify(ByVal unObjecto As Object)
      Console.WriteLine("El nuevo precio es:" & unObjecto)
   End Sub
```

Conclusión

En este documento ilustramos algunas categorías y tipos de los patrones de diseño. La mayoría parte vi sobre que es el patrón de diseño y su esqueleto para la solución a problemas comunes en el desarrollo.

Referencias

- https://msdn.microsoft.com/es-es/library/bb972240.aspx
- http://www.um.es/ead/red/M10/caceres.pdf