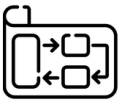
Patrones de diseño

Los patrones de diseño son soluciones probadas y documentadas a problemas comunes de desarrollo de software. También son usados en la automatización de software.

Patrones de diseño en automatización

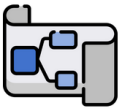
Screenplay:

Este patrón tiene un enfoque de desarrollo encaminado por el comportamiento Behaviour Driven Development (BDD). Es una estrategia de desarrollo que se enfoca en prevenir defectos en lugar de encontrarlos en un ambiente controlado.



O Presenta un alto desacoplamiento de la interfaz de usuario.
O Propone trabajar en términos de tareas y no de interfaz de usuario.
O Cumple con los principios SOLID.

Page Object:



O Es un patrón de diseño que representa los componentes web o página web en una clase.
O Se utiliza en las pruebas automatizadas para evitar código duplicado y mejorar el mantenimiento de las mismas.
O No cumple con los principios SOLID.
¿Qué son los principios SOLID?

SOLID

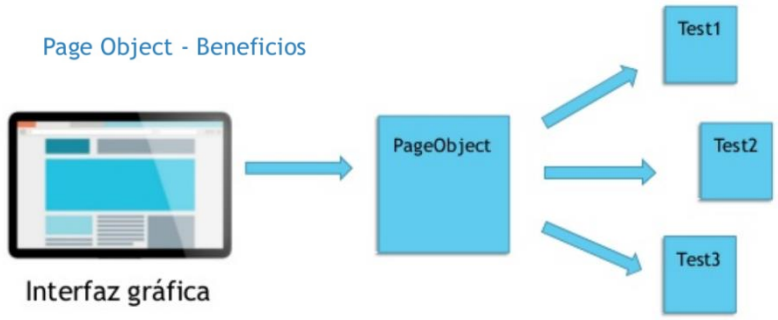
Es un acrónimo introducido por Robert C.Martin para definir los cinco principios básicos de la programación orientada a objetos:

O Single responsibility.
O Open-closed.
O Liskov substitution.
O Interface segregation.
O Dependency inversion.
SOLID tiene bastante relación con los patrones de diseño.
Los principios SOLID son guías o buenas prácticas que pueden ser aplicadas en el desarrollo de software para eliminar malos diseños.
Provocan que el programador tenga que refactorizar el código fuente hasta que sea legible y extensible, es decir, que pueden ayudar a escribir un mejor código: más limpio, mantenible y escalable.
Algunos de los objetivos a tener en cuenta estos principios para código son:
O Crear un software eficaz que cumpla con su cometido y que sea robusto y estable.
O Escribir un código limpio y flexible ante los cambios: que se pueda modificar fácilmente según necesidad, que sea reutilizable y mantenible.
O Permitir escalabilidad: que acepte ser ampliado con nuevas funcionalidades de manera ágil.
En definitiva, desarrollar un software de calidad.

Page Object Model también llamado POM.

Es la implementación del patrón de diseño Page Object utilizado en la automatización de pruebas.
Tiene como objetivo mejorar el mantenimiento de las pruebas y reducir la duplicación de código.
El concepto básico en el que se basa el Page Object Model es el de representar cada una de las pantallas que componen al sitio web o la aplicación que nos interesa probar como una serie de objetos que encapsulan las características (localizadores) y funcionalidades representadas en la página, permitiendo consolidar el código para interactuar con los elementos de una página en cada uno de los PageObjects.
Las páginas web se representan como clases y los diversos elementos de la página se definen como variables en la clase.
Todas las interacciones de usuario posibles se pueden implementar como métodos en la clase.
Estas interacciones se dividen en acciones y validaciones.
Las acciones reflejarán los pasos que el usuario realiza dentro del sitio web o aplicación que se encuentra bajo prueba.
Las validaciones serán métodos que reflejen la confirmación de que el sistema se comporta como se espera luego de una acción específica.

Cualquier cambio que se produzca en la UI únicamente afectará al PageObject en cuestión, no a los test ya implementados.



Selenium WebDriver.

Es uno de los frameworks más importantes con los que se generan pruebas automatizadas.
Es un conjunto de herramientas para la automatización de navegadores web que utiliza las mejores técnicas disponibles para controlar las instancias de los navegadores y emular la interacción del usuario con el navegador.
Permite a los usuarios simular interacciones realizadas por los usuarios finales como:
O Insertar texto en los campos.
O Seleccionar valores de menús desplegables y casillas de verificación.
O Etc.
Además también permite realizar validaciones en el comportamiento de cualquier componente.
Esto nos sirve para verificar el comportamiento esperado que se encuentra detallado en los casos de prueba diseñados y listos para ser automatizados.
WebDriver utiliza las API de automatización del navegador proporcionadas por los desarrolladores de los navegadores para controlarlo y ejecutar pruebas.
Es como si un usuario real estuviera manipulándolo con la principal diferencia en la velocidad de ejecución, WebDriver es más veloz que una ejecución manual.
Selenium es compatible con muchos lenguajes de programación como Java, C #, Python, etc., y con múltiples navegadores como Google Chrome, Firefox, Intebrowser driverSelenium Webdriver se enmarca en tres componentes principales:
O Los tests a ejecutarse (escritos con un lenguaje a elección entre varios como Java, C#, Node JS, etc);“Client API”.
O Un servidor standalone en donde se ejecutarán los casos de prueba.
O Un browser driver que conectará los scripts generados con la client API con el browser a través del Selenium Server.
Gráficamente puede verse así: