Preguntas teóricas de exámenes anteriores:

Similitudes y diferencias entre pruebas basadas en cobertura de código y pruebas basadas en mutantes.

Ambos son métodos de pruebas estructurales, disponemos del código para hacer el análisis. Las pruebas basadas en cobertura de código y mutantes se desarrollan de forma muy parecida. Tanto los mutantes como la cobertura de código nos dan un valor que nos indica "cómo de bueno" es el conjunto de pruebas disponibles hasta el momento. Y tanto uno como otro nos dan indicaciones sobre la parte del código pendiente de ser probado. La principal diferencia entre uno y otro es en la forma de conseguir este valor y el tipo de información que proporcionan. Mientras las pruebas basadas en cobertura de código solo intentan conseguir que se ejecuten el mayor número de líneas de código, las basadas en mutantes tienen indirectamente en cuenta cómputos realizados por esas instrucciones, introduciendo modificaciones en el código y comparando con los resultados obtenidos. En ambos casos se prueba lo que hay, el código disponible, y no son por tanto útiles para detectar lógica perdida (p.ej., funcionalidades no implementadas), ambas son malas con código muerto, etc.

¿Por qué no es recomendable utilizar los xpath para recuperar componentes de las aplicaciones probadas?

No es recomendable usar xpath por dos razones fundamentales:

- Dependencia del navegador, distintos navegadores lo interpretan de forma distinra, incluso algunos no lo soportan como Explorer.
- Poca manteniblidad, cambios en la estructura de las páginas pueden hacer fallar las pruebas.

¿Qué diferencia hay entre esperas implícitas y explítas? ¿Por qué es necesario utilizarlas?

Las esperas implícitas son manejadas por el gestor de eventos de Selenium, fijamos una espera con una duración máxima para todas las acciones. Las esperas explícitas se realizan sobre elementos concretos esperando a la satisfacción de condiciones concretas.

Dado un mock mock, al intentar definir su comportamiento con when(mock.smellyMethod(anyInt(), contains("asparag"), "red")).thenReturn(true); se producirá una excepción, ¿por qué?

Si usamos un matcher para un argumento entonces todos los argumentos han de definirse con machers. Podemos sustituir "red" por "eq("red")".

¿Qué diferencia hay entre un mock y un spy?

Un mock es un objeto con la habilidad de tener un comportamiento programado y verificar las interacciones en las que interviene. Un spy es un mock creado como un proxy de un objeto real (simula algunos métodos y otros los redirige al objeto real). Mientras que a un mock simple es necesario implementarle funcionalidad, un spy permite ejecutar métodos del objeto real al que apunta.

¿Para qué sirve Cucumber?

Cucumber es una herramienta para automatizar pruebas en BDD, Desarrollo Dirigido por Comportamiento (Behaviour Driver Development). Cucumber perrmite ejecutar descripciones funcionales en texto plano como pruebas de software automatizadas.

BDD es un enfoque colaborativo al desarrollo de software que pretende eliminar la brecha de comunicación entre el negocio y la tecnología informática. La principal ventaja es el entendimiento entre clientes y desarrolladores.

¿Qué es PIT? Objetivos y fundamentos en los que se basa.

Es una librería para mutación de código compatible con otros frameworks como JUnit, constantemente actualizado y capaz de ejecutar numerosas pruebas de mutantes en un tiempo bajo. Para ello crea un proceso padre y gestiona los hijos mediante la paralelización. Si un hijo entra en bucle es capaz de finalizar este proceso y no escribe nada en disco.

¿Qué es EvoSuite? Objetivos y fundamentos en los que se basa.

Evosuite es una herramienta de apoyo a los tester que trata de optimizar la cobertura añadiendo casos de prueba siguiendo algoritmos de búsqueda de ejecución simbólica dinámica. Sus objetivos son:

- Cobertura total de casos de prueba.
- Generación de assert basados en mutantes.
- Mantener el conjunto de pruebas tan pequeño como se pueda.

Los problemas son su costo computacional, desgaste muy alto en tiempo y esfuerzo, puede generar casos depruebas no esenciales y que solo trata con aplicaciones de un thread.

Evosuite no es capaz de predecir errores ya que su único recurso es el código fuente de la clase probada, solo detecta errores de eficiencia