Revista Antioqueña de las
Ciencias Computacionales y la Ingeniería de Software

ISSN: 2248-7441

www.fundacioniai.org/raccis
raccis@fundacioniai.org

Automate functional testing

Automatizar las pruebas funcionales

Ramesh Kalindri
Tata Consultancy Services Limited. *ramka(AT)tata.in*

INFORMACIÓN DEL ARTÍCULO

Tipo
Artículo corto

Historia
Recibido: 19-01-2014
Correcciones:
Aceptado: 28-04-2014

Keywords
Automatización, plan de pruebas, pruebas funcionales, pruebas manuales.

Palabras clave
Automation, functional testing, manual testing, plan testing.

ABSTRACT
Currently, software engineers are increasingly turning to the option of automating functional tests, but not always have successful in this endeavor. Reasons range from low planning until over cost in the process. Some principles that can guide teams in automating these tests are described in this article.

RESUMEN
Actualmente, los ingenieros de software están recurriendo con mayor frecuencia a la opción de automatizar las pruebas funcionales, pero no siempre tienen éxito en esta empresa. Las razones van desde una baja planeación hasta un sobre costo en el proceso. En este artículo se describen algunos principios que pueden guiar a los equipos en la automatización de estas pruebas.

1. Introducción

Cada proyecto software que genera entregas continuas tiene entre sus elementos de producción a la integración permanente y a las pruebas automatizadas. Estas pruebas se realizan en diferentes niveles, desde las unitarias, las difusas, hasta las funcionales. Una de las ventajas de la automatización de estas últimas es la reproducibilidad y el tiempo de ejecución predecible, que deben servir como indicadores de la calidad del software, después de cada versión. A menudo, la automatización de las pruebas funcionales se convierte en un cuello de botella, por lo que el ingeniero de software se debe familiarizar con los principios básicos de cómo crear este tipo de pruebas. A continuación se describen algunos principios básicos para lograr la automatización de estas pruebas.

2. Principios para la automatización

2.1 Diseñar las pruebas

La suite de pruebas se puede comparar con un árbol bonsái: al principio el centro son las raíces y el tronco, luego se eligen las principales ramas que van a crecer, y posteriormente se espera a que broten hojas sanas. En las pruebas se puede proceder de manera similar: crear una clase base que sea responsable de las funciones principales de la aplicación, como el despliegue; con base en las especificaciones se identifica la funcionalidad principal de la aplicación, que cubrirán las pruebas; y luego cada día se añaden más pruebas, en paralelo con la implementación. Cada método de apoyo a la prueba, como la creación de nuevos usuarios, se debe realizar separado de las pruebas, es decir, se implementa en diferentes clases. El ingeniero de software debe mantener las clases en los directorios que incluyen la funcionalidad principal de la aplicación, lo que constituye una buena práctica para crear una clase abstracta que defina los métodos comunes a muchas características. Si se están probando aplicaciones web, lo recomendable es aplicar el patrón de diseño del objeto página, donde una clase y sus métodos corresponden son la funcionalidad de una sola página, o, en el caso de grandes páginas, a uno de los componentes en la misma.

2.2 No todo es automatizable

La automatización es costosa, así que lo mejor es probar en primer lugar la funcionalidad principal de la aplicación. Algunas pruebas se pueden hacer rápida y fácilmente de forma manual, pero los *scripts* potenciales podrían ser difíciles de implementar. Vale la pena automatizar las tareas tediosas que necesitan ser repetidas muchas veces, lo mismo que las pruebas que requieren gran esfuerzo de validación de datos.

2.3 Escribir pruebas cortas

En situaciones en las que falla una o más pruebas, cualquier miembro del equipo de desarrollo debe ser capaz de localizar fácilmente la causa del error. Por esta razón, no debería haber un máximo de cinco aserciones en cada método de prueba, y cada método debe proporcionar

el registro completo de las acciones de prueba. Es aconsejable el uso de técnicas de Behavior-Driven Development (BDD), pero cuando no se utiliza un *framework* de prueba específico los próximos pasos de la prueba se deben colocar bajo comentarios //*given* //*when* //*then*.

2.4 Pruebas independientes

Cada método en la clase de prueba debe ser una entidad separada, no depender de otras pruebas, y se debe poder ejecutar una sola prueba en cualquier momento. De lo contrario, será costoso mantener la suite de estas pruebas en el futuro (la relación entre las pruebas debe ser seguida y actualizada sobre una base regular). Muchas veces hay situaciones en las que la prueba requiere ciertas condiciones previas que se deben cumplir, por lo que se deben extraer del método externo, y ejecutar al inicio de la prueba. Si estas condiciones son comunes a todos los métodos en la clase de prueba, se pueden incluir en el método aplicado al inicio.

2.5 El centro es la legibilidad

El código fuente debe ser auto-documentado, y cada uno de los interesados que revise las demás líneas del código debe entender lo que están haciendo las pruebas, y por qué fue escrito, aunque se recomienda tratar de evitar los comentarios en el código fuente, porque también tienen que ser actualizados. Vale la pena pasar un poco más de tiempo de lo habitual en los métodos de nomenclatura, para lograr que el código sea más legible. Refiriendo nuevamente a las técnicas BDD, cada método de prueba debe comenzar con la palabra *debería* en lugar de *prueba*, porque de acuerdo con esta convención es posible notar de inmediato lo que hace un método de prueba en particular, o que es especialmente útil cuando se analiza el informe de la pruebas.

2.6 Pruebas rápidas

Como se mencionó al principio, la pruebas funcionales automatizadas debe ser un indicador de la calidad de la aplicación, por lo que cada paso en el proceso de entregas continuas debe especificar la duración máxima, y, de acuerdo con el concepto y tan pronto como le sea posible, el equipo de desarrollo deberá obtener información acerca de la calidad del software. Para automatizar las pruebas funcionales el tiempo de duración debe ser no más de unos pocos minutos, y para una suite de pruebas exhaustiva es necesario ejecutar pruebas en paralelo, a menudo en máquinas separadas. La virtualización puede ser muy útil en este caso.

2.7 Pruebas que resistan el cambio

El inconveniente de las pruebas funcionales automatizadas más frecuentemente mencionado es su baja resistencia al cambio dentro de la aplicación, especialmente en la interfaz gráfica de usuario. Por lo que en el caso de las aplicaciones web las pruebas deben ser resistentes a los cambios en el contenido del sitio. Lo recomendable es que las pruebas comprueben sólo la funcionalidad, una característica que hace posible ejecutarlas desde diferentes lugares. Aunque esto no quiere decir que no se deban escribir pruebas automatizadas para comprobar el contenido de una página, si la decisión es crear este tipo de pruebas, se debe seguir la técnica Data-Driven Testing (DDT). Lo que significa que el contenido de la comprobación se debe separar del código fuente. En las aplicaciones web los cambios de diseño de las páginas son frecuentes, y tienen alto impacto en la interfaz de usuario, por lo que al diseñar esa interfaz se le debe asignar un identificador único para cada sección, y la página se puede utilizar para las pruebas, incluso si hay cambios en la jerarquía del sitio.

2.8 La automatización no reemplaza lo manual

Las pruebas funcionales automatizadas pueden ser la principal técnica de prueba en un proyecto, pero de ninguna manera la única. Las pruebas automatizadas son reproducibles y su cobertura siempre es la misma, además, las pruebas exploratorias se caracterizan por una baja reproducibilidad, pero son capaces de cubrir áreas a las que no llegan las automatizadas. También es conveniente recordar que el estado *verde* de las pruebas automatizadas no significa que la aplicación está libre de errores. Esta situación a menudo distrae a los probadores, y puede afectar la precisión de la prueba.

3. Conclusiones

- Automatizar las pruebas consiste en que el probador escribe *scripts* y utiliza otro software para probar el software. Es un proceso en el que se automatiza un proceso manual.

- La automatización de las pruebas se utiliza para ejecutar los escenarios de prueba que se realizan manualmente, pero de forma rápida e iterativamente.

- Además de las pruebas funcionales, también se pueden automatizar otros procesos en la aplicación, como la carga, el rendimiento, y el estrés.

- Con ella se incrementa la cobertura de la prueba, se mejora la precisión, y se ahorra tiempo y dinero en comparación con las pruebas manuales.

- Los principios descritos en este trabajo les sirven a los equipos de pruebas para planificar la automatización y para decidir, en cada aplicación bajo prueba, si es conveniente o no implementarla.