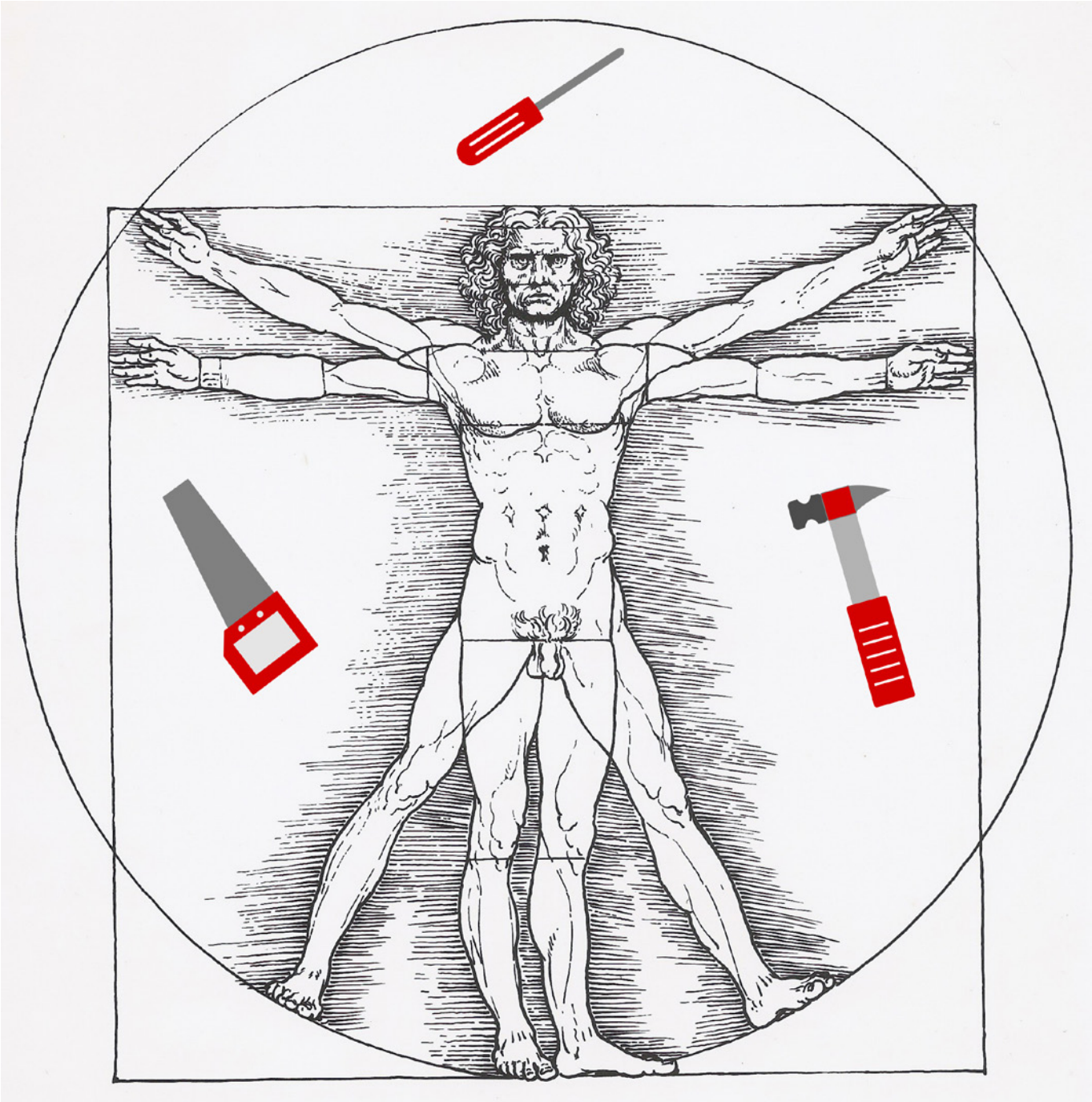
Conflicto entre las herramientas de automatización de pruebas de software y la accesibilidad en dispositivos móviles.

Jonathan Chacón Barbero
Consultor en accesibilidad, usabilidad y nuevas tecnologías
Socio de Asepau

Dentro de las distintas etapas del desarrollo de software se encuentran tareas de verificación y mantenimiento de la calidad del producto. Estas tareas consisten en probar el software que se está desarrollando, cubriendo dentro de las distintas pruebas las distintas pantallas y operaciones que puede realizar un usuario durante una sesión completa de la aplicación.

Estas pruebas suelen ser muy repetitivas y siempre mantienen un mismo proceso por lo que es muy habitual y recomendable automatizar estas pruebas para agilizar el proceso de mantenimiento del proyecto.

Para crear esta automatización se utilizan herramientas como Appium, Selenium o frameworks creados por Google o Apple para sus plataformas como sucede con la librería XCTest y su extensión XCUITest.



Uso básico de las herramientas de automatización

Estas herramientas realizan operaciones sencillas para manipular la interfaz de la aplicación que están probando. Entre las distintas operaciones se pueden encontrar estas:

- Obtener todos los botones presentes en una pantalla
- Buscar un elemento de la interfaz (sea botón, etiqueta u otro tipo de control) identificándolo por el texto que se muestra por pantalla.

- Buscar un elemento que se muestra por pantalla identificándolo por un identificador interno

- Simular un gesto de toque o arrastre sobre un punto de la pantalla o un elemento previamente localizado

Con estas operaciones podemos programar scripts de pruebas que simulen, por ejemplo, un proceso de **iniciar sesión**. El script podría ser algo como:

- Arranca la app

- Busca el campo de texto con el identificador **imputUser**

- Haz tap sobre el campo

- Escribe el texto **Usuario de prueba**

- Busca el campo de texto con el identificador **inputPassword**

- Haz tap sobre el campo

- Escribe el texto **contraseña**

- Busca el botón con el texto **Iniciar sesión**

- Haz tap sobre el botón

- Espera 5 segundos

- Busca la etiqueta con el texto **Bienvenido a nuestra aplicación**

De esta forma podemos automatizar una prueba que puede resultar tediosa y repetitiva para una persona. Si el proceso no se puede completar bien porque en algún momento del desarrollo se ha cambiado algo en la interfaz o algo no funciona como se espera la herramienta de automatización notificará del problema y el equipo de desarrollo podrá resolver la inconsistencia entre lo que sucede y lo que se esperaba que sucediese en un uso con calidad.

Identificación interna de los elementos de la interfaz

Los distintos elementos de la interfaz se pueden localizar por la herramienta de automatización bien por su tipo (botón, etiqueta de texto, campo de texto, casilla de verificación, etc), por el texto que muestra por pantalla o por un *identificador interno* que suele ser

una cadena de texto. Es este *identificador interno* el que resulta ser el causante de los problemas de accesibilidad que existen a la hora de utilizar herramientas de automatización de pruebas de software. Esto se debe a que este *identificador interno* requerido por estas herramientas se genera tomando ciertos atributos de accesibilidad.

Las herramientas de automatización de pruebas utilizan estos atributos de accesibilidad simplemente por una razón de seguridad en las condiciones de ejecución de las pruebas. Para realizar estas pruebas se utilizan entornos simulados de los distintos dispositivos Android e iOS. Estos entornos simulados sólo permiten acceso a los atributos visibles de los elementos de la interfaz y a sus atributos semánticos.

Los atributos semánticos se componen de valores como el tipo de control (botón, etiqueta de texto, casilla de verificación, etc.), el estado (activo, inhabilitado, seleccionado, etc.) y algunos de sus atributos de accesibilidad como su identificador, su descripción alternativa o su rol para la accesibilidad.

Estos atributos dependen de cada plataforma ya que la API de accesibilidad de Android es distinta de la de Apple iOS.

En el caso de Apple iOS es el atributo *accessibilityIdentifier* utilizado en el sistema de dispositivos iPhone, iPad, Apple Watch y Apple TV para identificar individualmente cada uno de los elementos en pantalla. También se utiliza el valor del atributo *accessibilityLabel* que proporciona una descripción alternativa de los elementos visuales para lectores de pantalla.

En el caso de Android el atributo utilizado es el *contentDescription* que contiene la cadena de texto que se utilizará como alternativa textual a un elemento visual de la interfaz.

Barreras de accesibilidad involuntarias

Sabiendo que ese *identificador interno* se origina en elementos de accesibilidad y tomando la premisa que los responsables de crear estas automatizaciones desconocen para qué pueden ser utilizados esos atributos de accesibilidad a parte de las funciones que su herramienta de automatización les indica en la documentación de ayuda podemos deducir que involuntariamente se crean multitud de barreras de accesibilidad en las interfaces de las aplicaciones de dispositivos Android e iOS.

Los responsables de utilizar estas herramientas de automatización, normalmente los miembros del departamento de QA de software,

son los que modifican los valores de *accessibilityIdentifier* y *contentDescription* de algunos elementos de las pantallas de una aplicación para poder generar los scripts de automatización para realizar las pruebas de calidad.

Problemas con un botón

Veamos un ejemplo sencillo en el que un botón que utiliza una imagen en lugar de texto se puede convertir en un problema de accesibilidad por un mal uso de las herramientas de automatización de pruebas.

El botón al no mostrar texto por pantalla requiere de un *identificador interno* para que la herramienta de automatización pueda localizarlo. Por este motivo es necesario ajustar en el código fuente de la aplicación el valor para ciertos atributos de accesibilidad de dicho botón.

Los valores que puede utilizar un responsable de QA para nombrar a un botón que no muestra texto por pantalla ya que utiliza una imagen puede ser, por ejemplo, el texto **btnSubmit**. Este valor, recordemos, tiene que tener significado para el script que se ejecutará por la herramienta de automatización de pruebas y debe ser útil sólo para el responsable de QA y la herramienta por lo que no tiene por qué cumplir ninguna norma o criterio de comprensibilidad general para la aplicación. Esto hace que identificadores como **inputUser**, **tablaDeResultados**, **casilla-de-verificacion-de-privacidad** o **campo_estupido** puedan ser valores aceptables y reales para alguien de QA.

Esto en Apple iOS puede provocar un problema menor ya que lo que el lector de pantallas leerá es el valor del atributo *accessibilityLabel* y no *accessibilityIdentifier* pero, en cambio, en Android el problema si es muy preocupante ya que es el valor de *contentDescription* el que se utiliza como alternativa textual generando una barrera de accesibilidad importante para los usuarios ciegos de esa aplicación en Android.

Este tipo de problemas pueden resultar divertidos cuando por ejemplo un usuario ciego de lector de pantallas en la aplicación para solicitar cita previa en su servicio de asistencia sanitaria encontraba imágenes con textos alternativos como **campo-estupido** o en una aplicación de compra online de una cadena de supermercados nacional todas las imágenes del banner de ofertas rápidas tenían el texto alternativo de **imagenParaProductoMalo**. El problema no era tan divertido cuando a la hora de confirmar la petición de cita previa entre el botón de aceptar la cita o cancelarla los textos

identificados por el lector de pantallas eran **btn1001** y **btn1023** siendo totalmente incomprensibles para el usuario ciego.

Pero pueden suceder problemas más importantes. Imaginemos por ejemplo que el responsable de QA necesita acceder a una tabla de resultados de una pantalla para consultar cuántos elementos aparecen en dicha tabla. Esto se puede realizar, por ejemplo, para crear una prueba de un componente de búsqueda de una base de datos creando un script de pruebas que utiliza el campo de texto para buscar y debe verificar que se han encontrado resultados.

A nivel de código en la aplicación en Android se utiliza un elemento TableView para contener las distintas celdas de la tabla. En Apple iOS se utiliza el elemento UITableView. Ambos elementos **no** deben ser accesibles ya que la accesibilidad debe ajustarse para las celdas de la tabla y no para su contenedor.

En el caso de iOS el utilizar el atributo *accessibilityIdentifier* puede crear problemas para los lectores de pantalla si se están utilizando interfaces Web o híbridas provocando que todas las celdas de ese contenedor sean identificadas como una única línea de texto por el lector de pantallas haciendo la lectura y comprensión de los datos más complicada para un usuario ciego.

En el caso de Android el problema es aún más grave ya que cualquier lector de pantallas que encuentre un elemento con *descripción alternativa* lo tratará como un contenido y no como un contenedor por lo que el lector de pantallas de Android, al encontrar la tabla de celdas simplemente identificará todo ese contenido como una etiqueta de texto equivalente al valor del atributo *contentDescription*.

Ejemplos de este problema en la vida real lo podemos encontrar en cierta aplicación para pedir comida a domicilio en la que la lista de comentarios de un restaurante era identificada por el lector de pantallas como **tableOfComments** pero el lector de pantallas no podía acceder a ninguno de los comentarios mostrados en la pantalla.

El problema se complicaba aún más para las personas ciegas cuando en una aplicación de mensajería instantánea en la lista de mensajes enviados y recibidos el usuario de lector de pantallas sólo encontraba el texto **listOfMessages**.

Complicando el problema para iOS

Algunos responsables de QA a la hora de crear los scripts de automatización para aplicaciones iOS en lugar de utilizar el atributo

accessibilityIdentifier utilizan en su lugar el atributo *accessibilityLabel* creando todos los problemas graves que aparecen para Android ya que el atributo *accessibilityLabel* se utiliza como descripción alternativa para los elementos visuales de la interfaz.

La solución

Nadie discute la utilidad y la necesidad de las herramientas de automatización de pruebas dentro de la industria del software. Pero no debe ser admisible que un criterio de comodidad para el equipo de desarrollo y mantenimiento de un proyecto de software sea más importante que la calidad y la accesibilidad del proyecto final.

No es problema de los creadores de herramientas de automatización de pruebas el utilizar los atributos de accesibilidad. Los entornos de simulación no dan opción a utilizar otros valores debido a las capas de seguridad a la hora de ejecutar una aplicación para hacer las pruebas. Su responsabilidad para solucionar este problema debe ser la de proporcionar una buena documentación explicando el uso real que se hace de esos atributos de accesibilidad e incluir recomendaciones a la hora de aportar valores para dichos atributos de accesibilidad.

Por otra parte, es necesario que los *identificadores internos* sean apropiados para la accesibilidad de la aplicación y que el conjunto de atributos de accesibilidad de un elemento contenedor sea ajustado de forma apropiada. Esta responsabilidad recae en los miembros encargados de crear esos scripts de pruebas automatizadas debiendo priorizar la accesibilidad final de la aplicación a la comodidad a la hora de crear los scripts de automatización de pruebas.

Esto hace necesario que los responsables de utilizar esta herramienta conozcan el impacto que se crea en la accesibilidad de las aplicaciones al hacer un uso irresponsable de los distintos atributos de accesibilidad utilizados por las herramientas de automatización.