Estrategia de Pruebas

Se desarrolló una herramienta buscando automatizar lo que más se pueda del proceso debido a la numerosa cantidad de apks y el poco tiempo que se tiene de horas hombre. Con la herramienta construida se prepara el conjunto de datos, se prepara el ambiente de ejecución, se ejecutan las pruebas *en paralelo* con base en un oráculo que se especifica previamente, y se dan reportes automáticos de los defectos encontrados.

Presupuesto

Se contó con 25 horas hombre para este trabajo, las cuales fueron distribuidas de la siguiente forma:

- 5 horas para preparar el conjunto de datos.
- 5 horas para construir el ambiente de ejecución.
- 5 horas para automatizar la ejecución de los casos de prueba.
- 5 horas para desarrollar el módulo de reportes.
- 5 horas para la documentación de la estrategia de pruebas.

Prerrequisitos

- NodeJS.
- Haber clonado el repositorio donde se encuentran los apk.
- SDK de Android, preferiblemente el 27.
- Haber creado una llave para firmar apks.

Instalación

En el directorio raíz:

- Ejecute el comando *npm install* para instalar las dependencias.
- Cree un archivo *.env* con las siguientes variables de entorno:
  - SIGN_KEY_PATH con el path a las llaves para firmar los apk.
  - STORE_PASS con la contraseña de la store donde está la llave para firmar los apk.
  - KEY_PASS con la contraseña de la llave para firmar los apk.
  - ALIAS_NAME el alias que creó para firmar los apk.
- Modifique el archivo *config.js* con la configuración deseada para las siguientes variables:

- PACKAGE_NAME con el nombre del paquete de la aplicación sobre la cuál se va a hacer regresión.
- PARALLEL_EXECUTIONS_AT_SAME_TIME con el número de ejecuciones de pruebas que puede haber en paralelo según la máquina en donde se va a ejecutar el programa.

Igualmente, se deben crear pares de emuladores del mismo dispositivo con la siguiente estructura:

*pruebas-**dispositivo**-**y***
*pruebas-**dispositivo**-**z***

donde ***y*** y ***z*** son enteros consecutivos y ***dispositivo*** un nombre con el cual se referencian a los dispositivos, debe ser el mismo para cada par.

Como ejemplo, este par de emuladores sirve:

pruebas-pixel2-0
pruebas-pixel2-1

Es de suma importancia tener en cuenta que todos los emuladores van a estar corriendo al mismo tiempo, y va a haber ejecución en un número especificado de parejas de emuladores al mismo tiempo, por lo que se deben crear la cantidad de parejas de emuladores que no consuman todos los recursos de la máquina.

Preparación del conjunto de datos

El input del programa se realiza mediante el directorio *data/input/*, en este directorio hay tres subdirectorios *baseline/*, *compareTo/* y *tmp/*. En el directorio baseline se copia el apk que va a servir de oráculo para identificar los mutantes, en el directorio *tmp/* se copian los mutantes, tal cuál como se recibieron en el repositorio *parcial2*, es decir, en este directorio van archivos con la siguiente estructura:

*com.evancharlton.mileage-mutant**x**/com.evancharlton.mileage_3110.apk*

donde ***x*** es el número del mutante

Una vez copiado el conjunto de datos como se especifica anteriormente, se debe correr, en el directorio raíz, el siguiente comando:

npm run init

Este comando, para cada apk mutado, lo firma y lo copia en el directorio *compareTo* con la estructura:

*compareTo/mutant**x**/*
donde ***x*** es el número del mutante

El código de lo que hace este comando se puede observar en el archivo *scripts/init.js*.

Creación y ejecución de los casos de prueba

La aplicación automáticamente toma cada archivo con extensión *.js* como caso de prueba. Estos archivos deben exportar un objeto de la clase *TestCase,* la cual se puede observar en el archivo *test/index.js,* esta clase tiene dos atributos, *name* como el nombre con el cual se quiere referenciar el caso, y *steps* como un arreglo de *instrucciones.* Cada *instrucción* es un objeto con dos atributos, *description* con una breve descripción del lo que hace la instrucción, y *command* con el comando de ADB.

La ejecución consiste en, en paralelo y con el número de parejas de dispositivos especificados anteriormente, correr todos los apk mutados. La herramienta permite que el tiempo de ejecución total disminuya de modo lineal con el número de parejas de dispositivos.

Una ejecución consiste en ejecutar los steps definidos en los casos de prueba, tomando una captura de pantalla automáticamente después de cada paso, y comparar las dos imágenes para ver si hay diferencias. Si hay diferencias entonces se tiene certeza que el apk está mutado con respecto al baseline pues la ejecución debería ser la misma.

Reportes

Los reportes se pueden observar en el directorio *ouput,* en este directorio hay, por cada ejecución, un subdirectorio que tiene como nombre, la fecha de ejecución en milisegundos de las pruebas, y dentro de este otro subdirectorio con el nombre del apk mutante que se está comparando.

Cada ejecución contiene tres subdirectorios, *base, mutated* y *differences;* como su nombre lo indica, *base* contiene las capturas de pantalla de la ejecución de los apk sin mutar, *mutated* contiene las capturas en los apk mutados y *differences* tiene la comparación de estas capturas. Después de cada *step* definido en el caso de prueba va a haber una captura de pantalla.

Con respecto a los reportes en sí, existen dos tipos de reportes, uno global y otro local por cada apk, ambos llamados *report.txt.* El reporte global se encuentra dentro del directorio de la ejecución de la prueba, es decir, el que tiene el nombre numérico, y este contiene un índice de los apks que probablemente eran mutantes. Por otro lado, el reporte local, que se encuentra dentro del directorio de la ejecución del

mutante, indica los steps en los cuales se encontraron, optimísticamente, los mutantes.

Conclusiones

Se logró construir satisfactoriamente una herramienta de automatización de pruebas, lo cuál es básicamente el objetivo del curso. Se logró automatizar casi todo el proceso de testing, exceptuando el de crear los casos de prueba ya que, debido a la naturaleza del problema, es más adecuado que sean desarrollados de forma manual, y, desafortunadamente el presupuesto no se ajustó para que se puedan realizar.

En los videos *doc/preparacionDatos* y *doc/ejecucion* se puede ver cómo se preparan los datos y cómo se ejecutan las pruebas.