Android Device Mutation Operator Description – REPORT

1. Introduction

This informal report provides an overview of the proposed Android device mutation operators. It provides information about each operator, including its name, acronym, category, and description. Additionally, the report provides examples of the proposed mutation operators.

The proposed mutation operators are organized by their categories.

2. Mutation Operator Description

2.1. Replacement

This operator replaces the current value of variables, method calls, parameters, and constants, or to instantiate an object with a different value instead of null. This category groups six different mutation operators: DDBID, DVO, RAIDD, RCTDGM, RLPDR, and RLPDPDR.

OPERATOR	**DefaultDeviceBuilderInstanceDeclaration**
ACRONYM	DDBID
CATEGORY	REPLACEMENT
DESCRIPTION	The operator replaces the current instantiation of a Location or Bluetooth class with a default implementation. These classes can be instantiated with different parameters and/or properties. Therefore, this operator declares a builder instance without these parameters/properties.

EXAMPLE	`// ORIGINAL_CODE`
	`Beacon beacon = new Beacon.Builder()`
	`     .setId1("2 f234454 - cf6d -4 a0f - adf2 -`
	`f4911ba9ffa6");`
	`     .setId2("1");`
	`     .setId3("2");`
	`     .setManufacturer(0x0118)`
	`     .build();`
	
	`// ORIGINAL_CODE`
	`LocationRequest locationRequest = new LocationRequest`
	`     .Builder(LocationRequest.PRIORITY_HIGH_ACCURACY)`
	`     .setIntervalMillis(5000)`
	`     .build();`

	`// MUTANT`
	`Beacon beacon = new Beacon.Builder().build();`

	`// MUTANT`
	`LocationRequest locationRequest = new LocationRequest();`

OPERATOR	**DeviceVariablesOperator**
ACRONYM	DVO
CATEGORY	REPLACEMENT
DESCRIPTION	The operator mutates scalar variables. In this case, the scalar variable may have its value incremented and decremented.

EXAMPLE	`// ORIGINAL_CODE`
	`discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300); // 5 minutes`

	`// MUTANT`
	`discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, Integer.MAX_VALUE); // 5 minutes`

OPERATOR	**RandomActionIntentDeviceDefinition**
ACRONYM	RAIDD
CATEGORY	REPLACEMENT
DESCRIPTION	The operator replaces the current declaration of a Bluetooth or Location action intent instance with a random value. It is worth noting that the action intent is mutated with another one of the same class.

EXAMPLE	`// ORIGINAL_CODE`
	`Intent bluetoothEnableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);`

	`// ORIGINAL_CODE`
	`Intent bluetoothEnableIntent = new Intent(BluetoothAdapter.ACTION_DISCOVERABLE);`

OPERATOR	**ReplaceCompatibleTypeDeviceGetMethods**
ACRONYM	RCTDGM
CATEGORY	REPLACEMENT
DESCRIPTION	The operator replaces the call of a get method of a given return type with another get method of the same return type and attached to the same object.
EXAMPLE	`// ORIGINAL_CODE`

	for(BluetoothDevice device :
	bluetoothAdater.getBondedDevices()) {
	   String deviceMacAddr = device.getAddress();
	}
	
	// ORIGINAL_CODE
	double latitude = location.getLatitude();

	// MUTANT
	for(BluetoothDevice device :
	bluetoothAdater.getBondedDevices()) {
	   String deviceMacAddr = device.getName();
	}
	
	// MUTANT
	double latitude = location.getLongitude();

OPERATOR	**RandomLocationProviderDeviceReplacement**
ACRONYM	RLPDR
CATEGORY	REPLACEMENT
DESCRIPTION	The operator replaces the Provider constant parameter of a Location instance with a custom String value or an existing LocationManager provider.
EXAMPLE	*// ORIGINAL_CODE* Location location = new Location("custom_provider");
	// MUTANT Location location = new Location("LocationManager.GPS_PROVIDER");

OPERATOR	**RandomLocationRequestBuilderPriorityDeviceReplacement**
ACRONYM	RLRBPDR
CATEGORY	REPLACEMENT
DESCRIPTION	The operator replaces the Priority constant parameter of a LocationRequest builder instance with a custom String value or an existing Priority constant. Similar to the RLPDR operator, it is only applied to Location.

EXAMPLE	*// ORIGINAL_CODE*
	LocationRequest locationRequest =
	LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY)
	.build();

```java
// MUTANT
LocationRequest locationRequest =
LocationRequest.Builder(Priority.PRIORITY_LOW_POWER).bui
ld();
```

2.2. Null

This type of operator replaces the current instantiation of an object with a null value. Four operators are categorized as Null: BDC, BDL, NDID, and NRDM.

OPERATOR	**BuggyDeviceCallback**
ACRONYM	BDC
CATEGORY	NULL
DESCRIPTION	The operator changes the current instantiation of a Bluetooth or Location callback with a null value. That is, the callback has no response to an event or user interaction.

EXAMPLE

```java
// ORIGINAL_CODE
private ScanCallback bleScanCallback = new
ScanCallback() {
   @Override
   public void onScanResult(int callbackType, ScanResult
result) {
       super.onScanResult(callbackType, result);
       // CODE
   }
}
```

```java
// MUTANT
private ScanCallback bleScanCallback = null;
```

OPERATOR	**BuggyDeviceListener**
ACRONYM	BDL
CATEGORY	NULL
DESCRIPTION	The operator changes the current instantiation of a Bluetooth or Location listener with a null value. Similar to the BDC operator, the listener has no response to an event or user interaction.

EXAMPLE

```java
// ORIGINAL_CODE
LocationListener locationListener = new
LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
```

	`        // CODE`
	`    }`
	`};`

	`// MUTANT`
	`LocationListener locationListener = null;`

OPERATOR	**NullDeviceInstanceDeclaration**
ACRONYM	NDID
CATEGORY	NULL
DESCRIPTION	The operator changes a current instance declaration of a Bluetooth or Location class with a null value.

EXAMPLE	`// ORIGINAL_CODE`
	`BluetoothAdapter btAdapter =`
	`bluetoothManager.getAdapter();`

	`// ORIGINAL_CODE`
	`BluetoothAdapter btAdapter = null;`

OPERATOR	**NullReferenceDeviceMethods**
ACRONYM	NRDM
CATEGORY	NULL
DESCRIPTION	The operator changes the reference parameters of a method with a null value. This operator tackles method overloading.

EXAMPLE	`// ORIGINAL_CODE`
	`locationManager.requestLocationUpdates(`
	`     LocationManager.GPS_PROVIDER,`
	`     5000,`
	`     10,`
	`     locationListener`
	`);`

	`// MUTANT`
	`locationManager.requestLocationUpdates(`
	`     LocationManager.GPS_PROVIDER,`
	`     5000,`
	`     10,`
	`     null`
	`);`

2.3. Switch

This type of operator switches the boolean value of a method parameter or changes the boolean return value of a method call to true or false. It groups two operators: SCDM and SCPDM.

OPERATOR	**SwitchConditionalDeviceMethod**
ACRONYM	SCDM
CATEGORY	SWITCH
DESCRIPTION	The operator switches the boolean return value of a Bluetooth or Location method to “true” and “false”. That is, the operator creates one mutant with a “true” value and another with a “false” value.

EXAMPLE	`// ORIGINAL_CODE`
	`boolean enabled = bluetoothAdapter.isEnabled();`

	`// MUTANT`
	`boolean enabled = true;`

OPERATOR	**SwitchConditionalParameterDeviceMethod**
ACRONYM	SCPDM
CATEGORY	SWITCH
DESCRIPTION	The operator switches the boolean parameter of a Bluetooth or Location method. Similar to the previous operator, it switches the value “true” to “false” and “false” to “true”.

EXAMPLE	`// ORIGINAL_CODE`
	`beaconManager.setEnableScheduledJobs(false);`

	`// MUTANT`
	`beaconManager.setEnableScheduledJobs(true);`

2.4. Trap

The Trap operators insert a trap method to reveal the reachability of a code in the application. It is specifically inserted into the callback or listener interface method. Whenever the trap is executed, the mutant is killed and the method is reachable. Two operators are grouped as Trap: TDC and TDL.

OPERATOR	**TrapDeviceCallback**
ACRONYM	TDC
CATEGORY	TRAP

DESCRIPTION	The operator inserts a trap method inside a callback. The mutant execution is finalized (i.e., the mutant is killed) whenever the trap method is executed, showing that the callback method is reachable.

EXAMPLE

```
// ORIGINAL_CODE
private ScanCallback bleScanCallback = new
ScanCallback() {
   @Override
   public void onScanResult(int callbackType, ScanResult
result) {
       super.onScanResult(callbackType, result);
   }
}
```

```
// MUTANT
private ScanCallback bleScanCallback = new
ScanCallback() {
   @Override
   public void onScanResult(int callbackType, ScanResult
result) {
       super.onScanResult(callbackType, result);
       TRAP_ON_CALLBACK();
   }
}
```

OPERATOR	**TrapDeviceListener**
ACRONYM	TDL
CATEGORY	TRAP
DESCRIPTION	The operator inserts a trap method inside a listener method. The mutant execution is finalized (i.e., the mutant is killed) whenever the trap method is executed, showing that the listener method is reachable.

EXAMPLE

```
// ORIGINAL_CODE
LocationListener locationListener = new
LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        // CODE
    }
};
```

```
// MUTANT
LocationListener locationListener = new
LocationListener() {
```

	`    @Override`
	`    public void onLocationChanged(Location location) {`
	`        // CODE`
	`        TRAP_ON_LISTENER();`
	`    }`
	`};`

2.5. Deletion

This type of mutation operator deletes a statement declaration of the application. It encompasses the DDM operator.

OPERATOR	**DeletionDeviceMethods**
ACRONYM	DSM
CATEGORY	DELETION
DESCRIPTION	The operator deletes a statement of methods that close, stop, and/or deallocate an existing process or service. It may cause collateral effects such as increasing the energy consumption of an application.

EXAMPLE	`// ORIGINAL_CODE`
	`// CODE`
	`fusedLocationProviderClient.flushLocations();`
	`// CODE`

	`// MUTANT`
	`// CODE`
	
	`// CODE`

2.6. Shift

This type of mutation operator moves the declaration of a method into another place of the source code.

OPERATOR	**ShiftDeviceMethodCall**
ACRONYM	SDMC
CATEGORY	SHIFT
DESCRIPTION	The operator shifts a Bluetooth or Location method call to another part of the source code. It can be shifted into the same method or in a different method from the same class.

EXAMPLE	`// ORIGINAL_CODE`
	`@Override`

```java
public void onCreate(Bundle savedInstanceState) {
    // CODE
}

@Override
public void onPause() {
   super.onPause();
   bluetoothServerSocket.close();
}
```

```java
// MUTANT
@Override
public void onCreate(Bundle savedInstanceState) {
   // CODE
   bluetoothServerSocket.close();
}

@Override
public void onPause() {
   super.onPause();

}
```