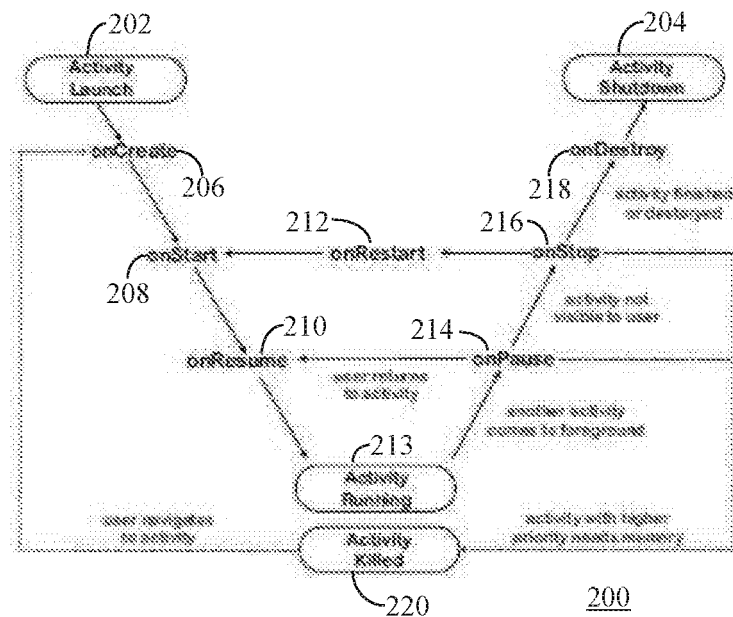


102		104	
#	Category	Energy Bug	Energy Hotspot
110	Hardware Resources	<p><i>Resource Leak:</i> Resources (such as the WiFi) that are acquired by an application during execution must be released before exiting or else they continue to be in a high-power state. <u>112</u></p>	<p><i>Suboptimal Resource Binding:</i> Binding resources too early or releasing them too late causes them to be in high-power state longer than required. <u>114</u></p>
		<p><i>WakeLock Bug:</i> WakeLock is a power management mechanism in Android through which applications can indicate that the device needs to stay awake. However, improper usage of WakeLocks can cause the device to be stuck in a high-power state even after the application has finished execution. This situation is referred to as a WakeLock bug. <u>122</u></p>	
120	Sleep-state transition heuristics		<p><i>Tail-Energy Hoop:</i> Network components tend to linger in a high power state for a short period of time after the workload imposed on them has completed. The energy consumed by the component between the period of time when the workload is finished and the component switches to the sleep-state is referred to as Tail Energy. Note that tail energy does not contribute to any useful work by the component. Scattered usage of network components throughout the application code increases power loss due to Tail Energy. <u>124</u></p>
		<p><i>Vacuous Background Services:</i> In the scenarios where an application initiates a service such as location updates or sensor updates but doesn't move the service explicitly before exiting, the service keeps on reporting data even though no application needs it. <u>132</u></p>	<p><i>Expensive Background Services:</i> Background services such as sensor updates can be configured to operate at different sampling rates. Unnecessarily high sampling rate may cause energy hotspots and therefore should be avoided. [8] Similarly, fine-grained location updates based on GPS are usually very power intensive and can be replaced by inexpensive, WiFi-based coarse-grained location updates, if an application is using both the WiFi and the GPS. <u>134</u></p>
130	Background Services		
140	Defective Functionality	<p><i>Immortally Bug:</i> Buggy applications may respawn when they have been closed by the user, thereby continuing to consume energy. <u>142</u></p>	<p><i>Loop-Energy Hoop:</i> Portions of application code are repeatedly executed in a loop. For instance, a loop containing network login code may be executed repeatedly due to reasons such as unreachable server. <u>144</u></p>

FIG. 1

2/11

**FIG. 2**

```

1 List<Integer> data;
2 LocationManager locationManager;
3 long Min_Update_Time = 10, Min_Distance = 1000 * 60 * 1;
4
5 @Override
6 public void onCreate(Bundle savedInstanceState) {
7     super.onCreate(savedInstanceState);
8     setContentView(R.layout.main);
9     locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
10    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
11                                           Min_Update_Time, Min_Distance, this);
12    someFunctionToManipulateDataList();
13 }
14 @Override
15 public void onStop() {
16     super.onStop();
17     try {
18         data.clear(); //this can throw an exception
19         locationManager.removeUpdates(this);
20     } catch (Exception ex) {
21         Log.v("Demo", "some exception happended");
22     }
23 }

```

300

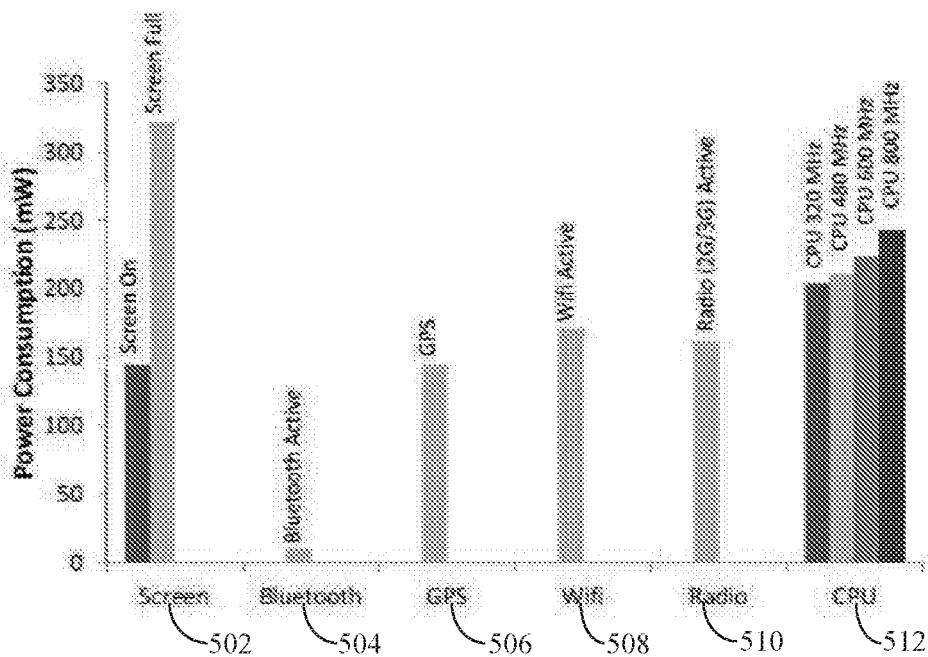
FIG. 3

3/11

<div style="margin-bottom: 10px;">402</div> <div style="margin-bottom: 10px;">406</div> <div>404</div>	<pre> 1 public Object[] nonAggregatedComm() 2 { 3 Object[] objectArray = 4 new Object[10]; 5 for(int i=0; i<10; i++){ 6 Object temp = downloadObject(i); 7 objectArray[i] = 8 processObject(temp); 9 } 10 return objectArray; 11 } 12 13 </pre>	<pre> public Object[] aggregatedComm() { Object[] tempArray = new Object[10]; for(int i=0; i<10; i++){ tempArray[i] = downloadObject(i); } Object[] objectArray = new Object[10]; for(int i=0; i<10; i++){ objectArray[i] = processObject(tempArray[i]); } return objectArray; } </pre>
--	---	---

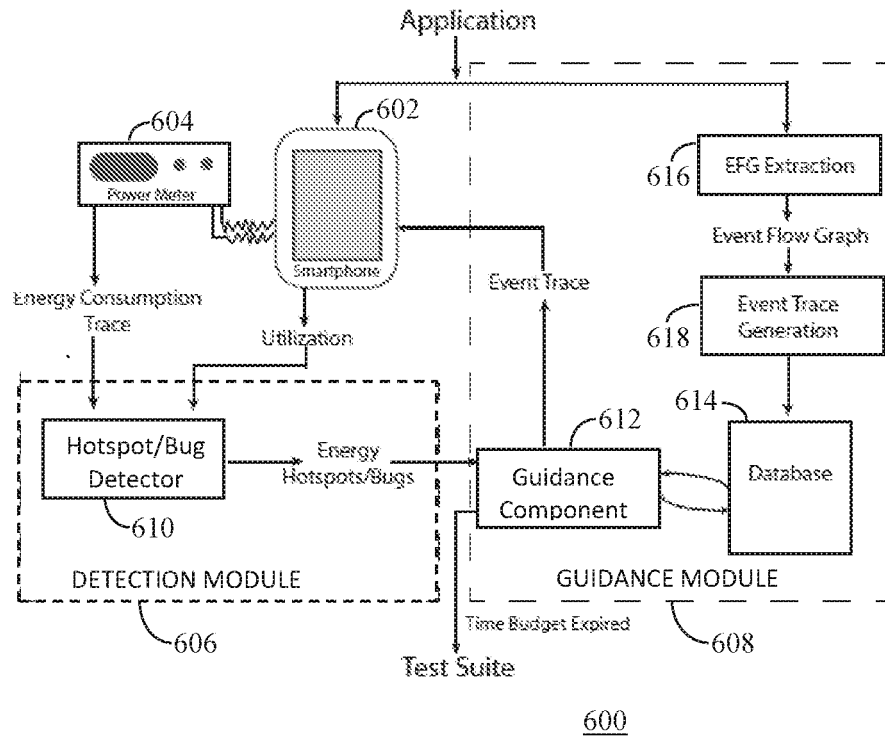
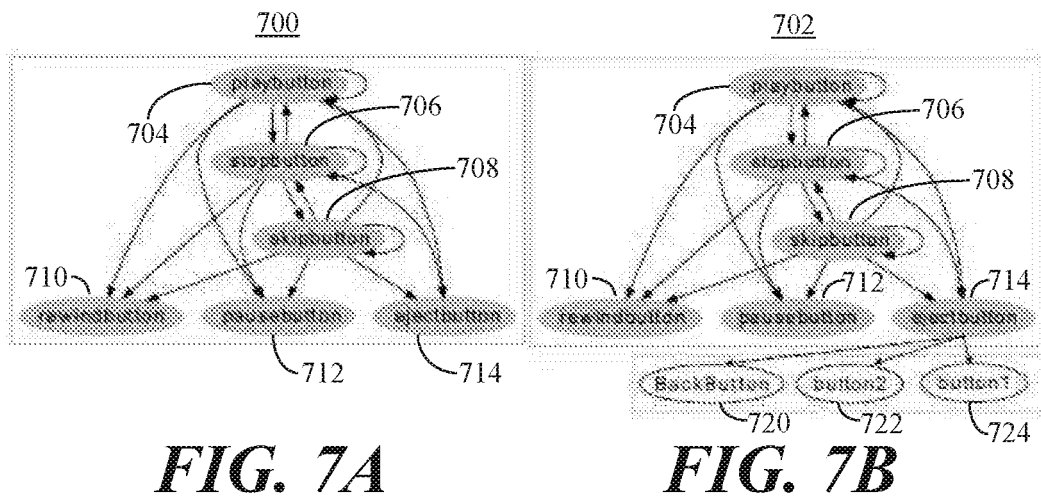
400

410

FIG. 4A**FIG. 4B****FIG. 5**

500

4/11

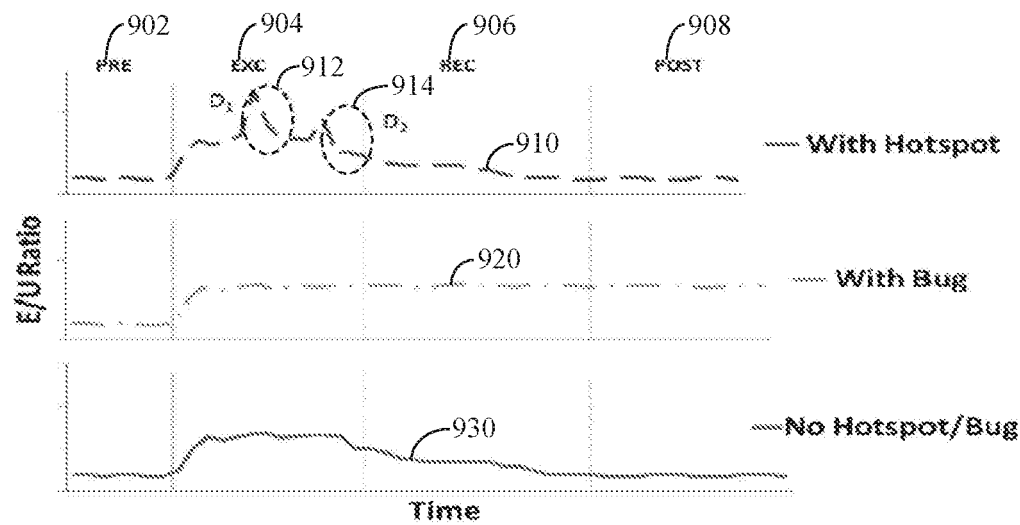
**FIG. 6****FIG. 7A****FIG. 7B**

5/11

800

Functionality	Number of APIs	Example
Power Management	3135	WakeLock.acquire()
Local Area Wireless Networks	2116	WifiLock.acquire()
Telecomm Networks	1691	SmsManager.sendMessage()
Haptic Feedback	783	Vibrator.vibrate()
GPS	146	LocationManager.requestLocationUpdates()
Audio/Video	94	Camera.startPreview()
Storage	66	DownloadManager.enqueue()
Others	25	SensorManager.getAltitude()

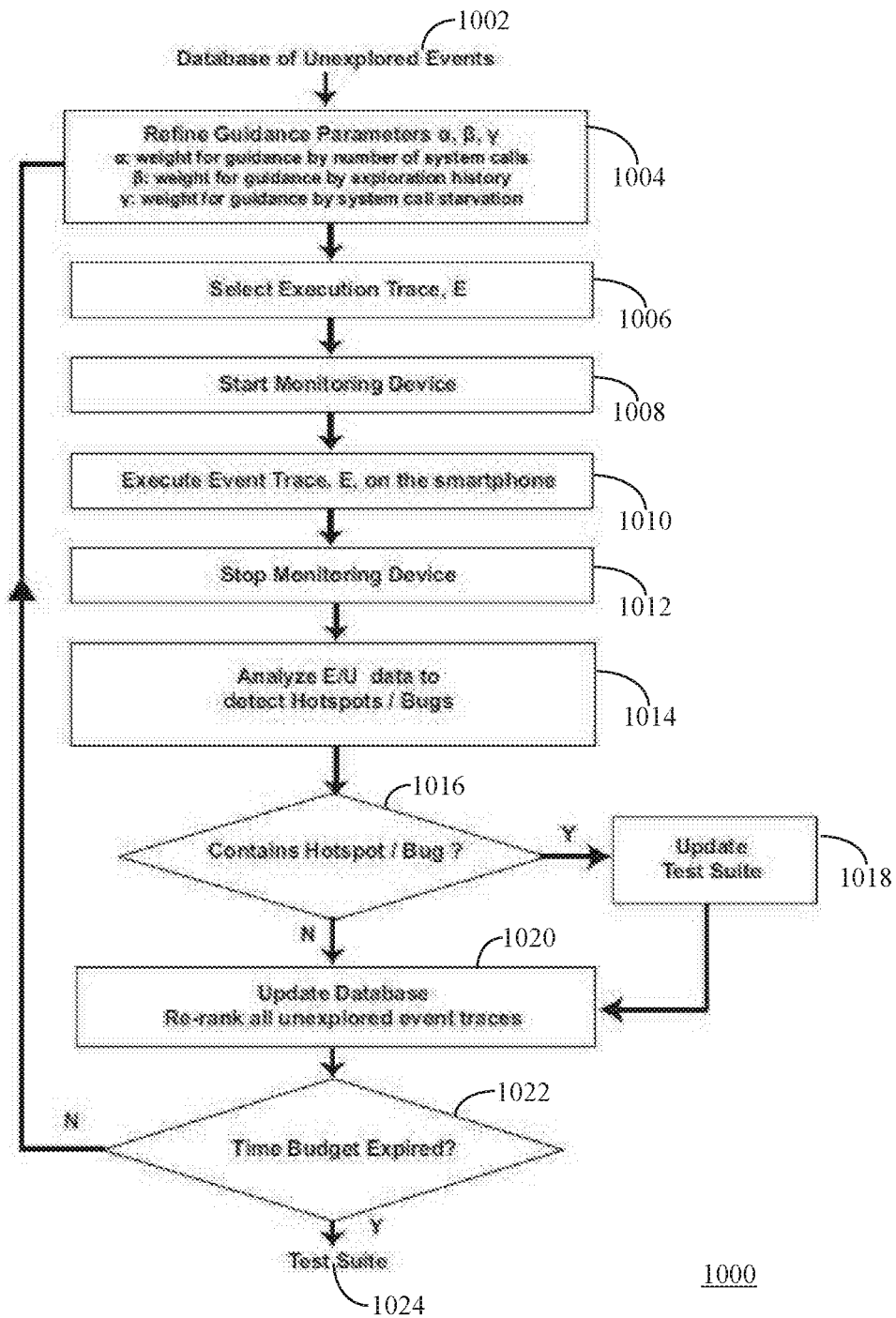
FIG. 8



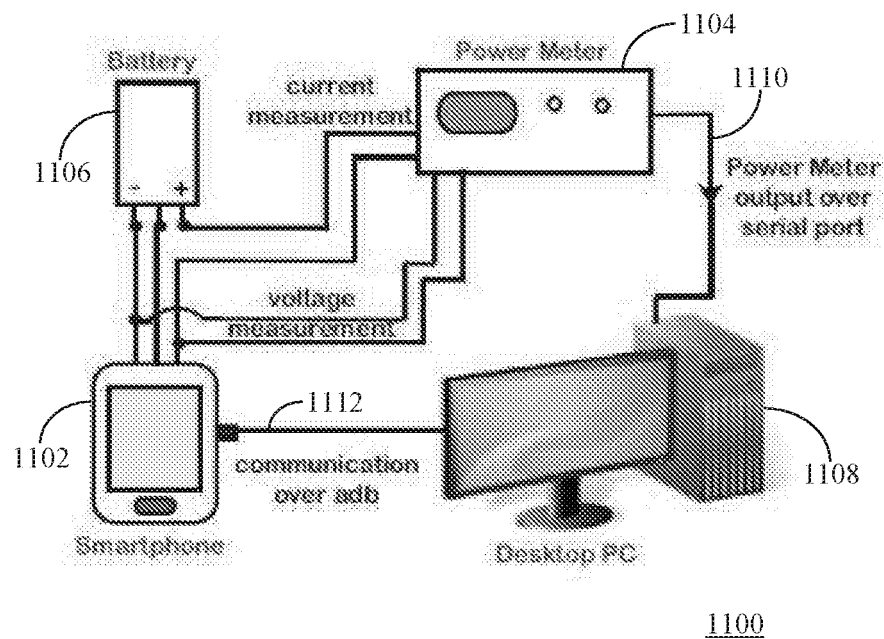
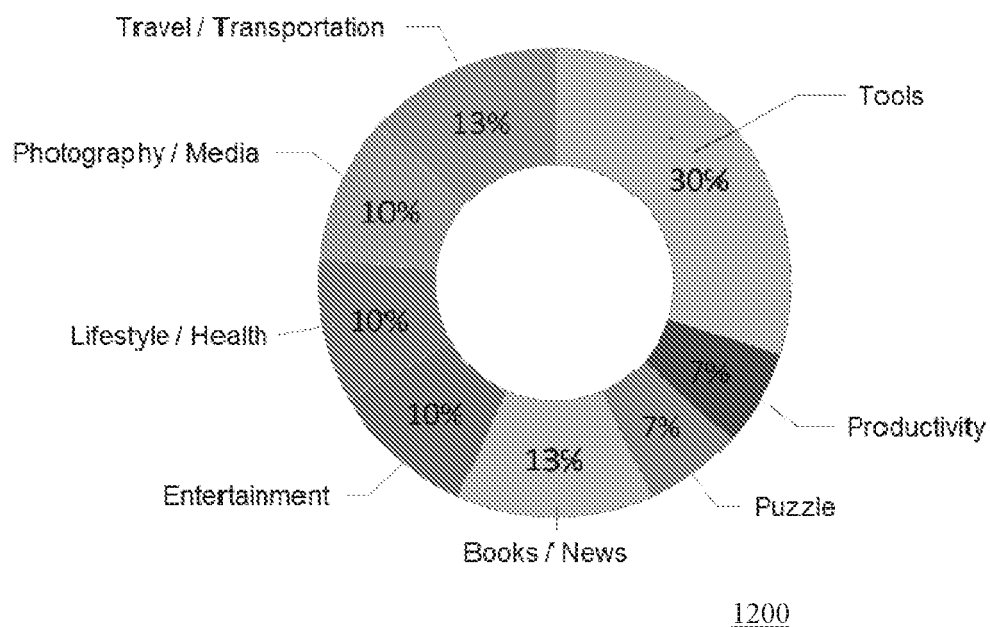
900

FIG. 9

6/11

**FIG. 10**

7/11

**FIG. 11****FIG. 12**

8/11

Application	Description	Feasible: Traces (k=4)	Bugs Found / False Positive	Hotspots Found / False Positive	Hotspot / Bug Type	Previously Reported
Aggl	A geocaching tool	131	Yes / No	Yes / Yes	Resource Leak	No
Aspica	Records tracks and waypoints	502	Yes / No	No / n/a	Vacuous Background Services	No
Montreal Transit	Fetches bus, subway and other transit information	64	Yes / No	No / n/a	Expensive Background Services, Suboptimal resources binding	No
Omniroid	Automated event/action manager	233	Yes / No	No / n/a	Vacuous Background Services, Im-mortality bug	Yes
Zamnam	Shows location-aware timeline	965	Yes / No	No / n/a	Vacuous Background Services	Yes
Sensor Test	Monitors and logs sensor output	2,800	Yes / No	No / n/a	Immutability bug	No
Epcorle	Displays traffic information	200	No / n/a	Yes / No	Suboptimal resources binding	No
760 KFMB AM	Listens to online radio	26	Yes / No	Yes / No	Vacuous Background Services, Sub-optimal resources binding	No
Food Court	Finds restaurants near a location	42	Yes / No	No / n/a	Vacuous Background Services	No
Fire and Blood	Simple touch and draw game	156	Yes / No	No / n/a	Vacuous Background Services	No
Speedometer	Shows measurements of sensors	2,492	Yes / No	No / n/a	Vacuous Background Services	No

1300

FIG. 13

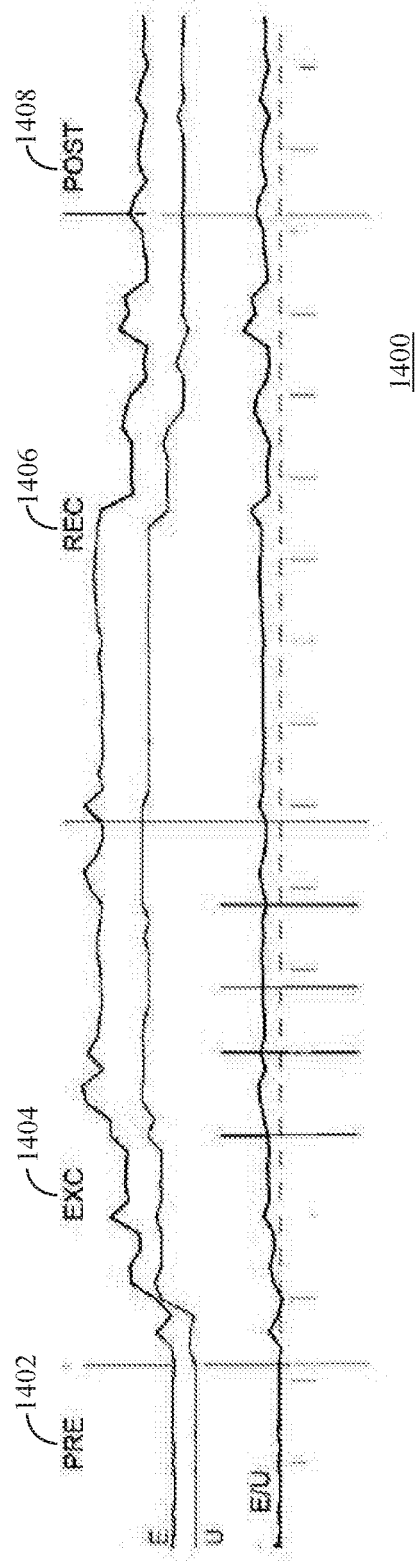


FIG. 14A

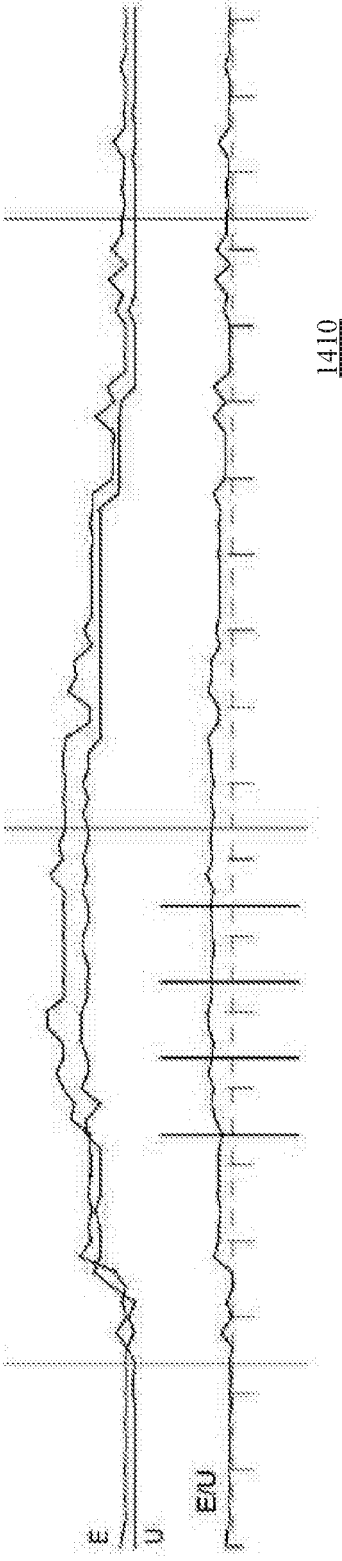


FIG. 14B

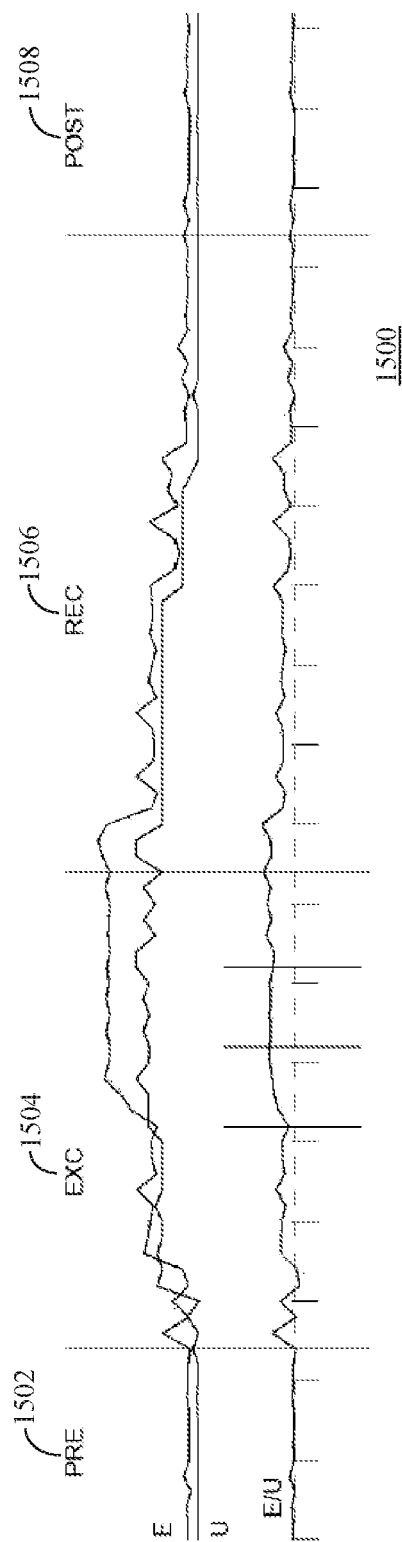


FIG. 15A

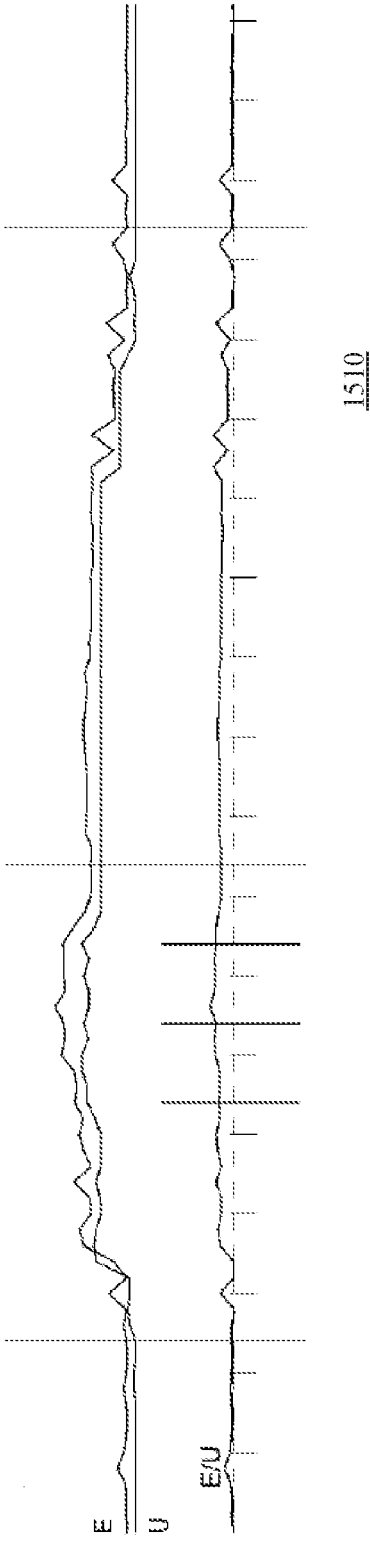


FIG. 15B

11/11

Application Name	System Call Coverage (%)	Code Coverage (%)	Lines of Code
Aagtl	100	21	11,612
Android Battery Dog	100	17	463
Aripuca	100	15	4,353
Kitchen Timer	100	30	1,101
Montreal Transit	89	11	10,925
NPR News	100	24	6,513
Omnidroid	83	36	6,130
Pedometer	100	56	849
Vanilla Music Player	86	20	4,081
Simple Chess Clock	100	49	448
WiFi ACE	100	27	504
World Clock	100	90	1,147

1600***FIG. 16***