Mobile Systems Engineering

Joana Iljazi
Ringaile Valiaugaite
Stanislav Kimov

Android Power Profiler

Introduction

Android device functions are not optimal and consume a lot of energy. The battery of a device could be used more efficiently. To optimize the battery consumption, we need to monitor the behavior of a device by gathering different metrics and developing several energy consumption modes that alter the device behavior according to these modes to use energy efficiently.

Literature review

Nowadays there is an increasing need for more and more independent mobile devices. In the same time, the computation power has become such that the same devices are more thirsty than ever for energy. The immediate solution to this issue would be better, longlife batteries. However, there has not been a major breakthrough in battery capacity for years, and is not expected to be for years to come [1].

Independently from the research question, on the literature review we noticed that the general methodology of the studies follows three main steps: measuring, analysing data collected, taking decisions and improving[2-8]. However the two main streams seem to be the ones that focus on measuring and collecting data either focused on the device as a whole, or on application level and particular applications of interest.The second stream is of the ones who are interested on a particular perspective of power consumption optimizations, and their study is guided by more focused research questions.

The motivation for the papers of the first type, is the importance of a smart, exhaustive and accurate measurement, as the first step toward energy optimization[2,3]. The usual questions are: which OS processes are consuming my battery, which applications are consuming my battery or is a specific application, function consuming too much battery?[4,5] Under these questions we find enough work done in creating the so-called profilers, applications that measure different parameters[6], and also studies that make a qualitative comparison on the market of energy profilers[6]. From this part we can derive some best practices for gathering measures, and building code optimized applications. To provide a positive user experience and develop energy-efficient applications, developers need to monitor the energy

consumption of their applications, in order to further optimise them [6]. Another thing to do when considering an energy-friendly mobile device is coding our applications with that in mind. For instance the case of web browsers energy consumption optimization, through coding a green webpage [4].

The second type of papers, seems to have learned from the literature that identifies the biggest consumers and focuses on advanced design techniques, in level of OS, communication protocols etc to optimize power management[7,8]. We would like to mention a paper that after reaching the conclusion that cellular networks impose high energy consumption on the mobile devices due to the radio resource allocation performed at the operator end, they develop a prototype of an energy saving algorithm on a commodity device and evaluate its operation in a real environment. [7]

Examples of gathering metrics

We divided the project in three parts. The first part we gathered the measurements about the device, while in the second part we showed the list of all the currently running processes and the selected process' measurement. The third part is to set some modes which gives a warning if a process is overloading the device.

1st part.

For the first part, we were able to gather these metrics:

- Cpu Frequency Measurement (read from the file "/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq");
- Cpu Usage Measurement (read from the file "/proc/stat");
- Network Measurement (read from the file "/proc/net/dev");
- Gps Status;
- Memory Measurement;
- Mobile Status;
- Screen Status;
- Wifi Status;
- Battery measurement;
- Bluetooth Status;

Most of the measurements are collected from the Android Context Class which is an interface that provides the information about the application environment. [1]

Problems

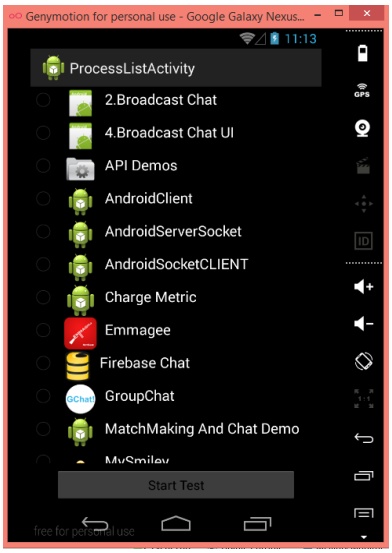
[1] More info: http://developer.android.com/reference/android/content/Context.html

To test our application in a real environment, we started to use Genymotion emulator[2] rather than an already build-in emulator in Android Studio because of its speed and it is easy-to-use. Although while running the application on the emulator, we were able to get only some measurements, while CPU frequency and usage measurements were not read , therefore we decided to test on a real device (Google Nexus 5). But we were still not able to modify these values to directly affect the device energy consumption and performance unless the device is rooted.

2nd Part.

To continue with our application, the second part was to list all the currently running processes and after the selected process to show the single process measurements.
We gathered these measurements:

- CPU Info (read from file “sys/devices/system/cpu/”);
- Memory Info (read from file “/proc/meminfo”);
- Battery Info (read from files “/sys/class/power_supply/battery/current_now”, “/sys/class/power_supply/battery/batt_current”, “/sys/class/power_supply/battery/batt_current_adc” and “/sys/class/power_supply/battery/current_avg”);
- Traffic Info (read from files “/proc/uid_stat/" + uid + "/tcp_rcv” and “/proc/uid_stat/" + uid + "/tcp_snd”).



We can see the list of the currently running applications and choose one of them to see the current load of the application every second. PowerProfiler shows in the log the current memory of application and the device memory as well as application CPU usage and System CPU usage and Network Traffic as for example:

[2] https://www.genymotion.com/

App/Available Memory:13.76/854.51MB
App/System CPU:2.97%/31.68%
CurrentN/A,Net Traffic(KB)N/A

Every time, when an application is chosen from the list, it is also started, so it is possible to see the application load while the application is in use and how it changes during the time.

Problems
To show the list of currently running processes we used
ActivityManager.getRunningAppProcesses()
but instead of listing all the processes, PowerProfiler just shows the partial list of currently running processes, as on Google Nexus 5 it shows just the applications installed by a user.

3rd Part

We set three modes on the device: “Saving”, “Normal” and “Performance”.
For the “Saving” mode, we check if the process load is bigger than 50 % or there is less than 75% free memory.
For the “Normal mode”, we check if the process load is more than 75% or there is less than 25% free memory.
If the criteria is met, a user gets a warning about the current state of Processor load and memory free and it turns off wifi and bluetooth and reduces the brightness of the screen.
The “Performance” mode is a default mode, just to monitor the performance of a device.

Problems
When there are several warnings that need to be shown to a user, they just overlap, therefore we created the message with all the information about the current process load in one warning.

Data analysis

From the first part of our project we understand what actual parameters can be measured from the phone as a whole. The point is to understand how all this parameters, which on themselves represent a process running in the device (for instance: gps activation, wifi, bluetooth etc), influence the discharge of the battery. Thus we design the experiment with two phases: online and offline phase as follow:

Online phase

We activate all the functionalities of the mobile device that we are measuring, and we gather measures each minute for two hour, how their values change. Two hours is actually the minimum time needed, since we are dealing with a learning problem where the historical data are characterized by 12 attributes. Calculating the min number of data points needed when the dimensionality is such, based on the VC-dimension [9], the threshold for reliable experimental results is 120 observations. Thus in the end of the process we will have a file with 120 points in time, whose uniques identifier is the timestamp (serving as the id and in the same time showing the time-effect) and with parameters as field of the Observation matrix. The vector of a single historical data has the format below:

d=[*timestamp,battery_used,wifi,bluetooth,mobile,gps,screen_on,screen_brightness,memory_free,cpu_usage,cpu_frequency,sent_net,received_net*]

Offline phase

This phase is conducted on three main steps:

1. Deciding on the type of analysis we will conduct and algorithms to use
2. Pre-processing the data
3. Running the algorithm and discussing the results

We use software R, for data preprocessing and analysis.

Deciding on the type of analysis we will conduct and algorithms to use

We are in the situation when we have historical data from which we want to learn something, in our case a model that would allow us to understand how different device parameters, influence the discharge life cycle of the battery. Since the dependent variable, which is the subject of our predictive model, takes continuous values, we notice that we are dealing with a regression analysis. Moreover, since we have more than one parameter-attribute we decide to use multivariable linear regression. Thus we will have the following equation and vector format:

$$D: \{ W \times O = Y \}$$

D is the vector of the observations ID.
W is the weight vector
O is the matrix of observations
Y is the vector of the output result (battery_used until current moment)

At this point each vector d, element of matrix O is of the form:

d=[*wifi,bluetooth,mobile,gps,screen_on,brightness,memory_free,cpu_usage,cpu_frequency,sent_net,received_net*]

Reflecting on the problem we thought as an alternative the use of stepwise regression instead. This is why:

The association of energy consumption to processes is a non linear combination. As an example, we know that the process of swapping induces a higher consumption of energy because it triggers a physical movement of data. So, let us now assume that because of a particular organisation of the state of a computing device a given memory cell is swapped if two processes consequently do not use it, and we have the following pattern of memory access:

Process	Memory Cell
A	M1
B	M2
C	M3
D	M1

Now if we have a sequence of process executions like A,B,C,A,B,C,A,B,C I will have to swap memory cells constantly, since:

Process	LastUM1	LastUM2	LastUM3	SwapOut	SwapIn
A	0	?	?	-	M1
B	-1	0	?	-	M2
C	-2	-1	0	M1	M3
A	0	-2	-1	M2	M1
B	-1	0	-2	M3	M2
C	-2	-1	0	M1	M3
A	0	-2	-1	M2	M1
B	-1	0	-2	M3	M2
C	-2	-1	0	M1	M3
A	0	-2	-1	M2	M1
B	-1	0	-2	M3	M2
C	-2	-1	0	M1	M3

So after twelve cycle I will have 10 Swap Outs and 12 Swap Ins.

Now If I have also process D

Process	LastUM1	LastUM2	LastUM3	SwapOut	SwapIn

A	0	?	?	-	M1
B	-1	0	?	-	M2
D	0	-1	?	-	-
C	-1	-2	0	M2	M3
A	0	-3	-1	-	-
B	-1	0	-2	M3	M2
D	0	-1	-3	-	-
C	-1	-2	0	M2	M3
A	0	-3	-1	-	-
B	-1	0	-2	M3	M2
D	0	-1	-3	-	-
C	-1	-2	0	M2	M3

So after twelve cycles I will have 5 Swap Outs and 7 Swap Ins with a clear result of saving battery.

Therefore, a simple assumption that increasing the number of processes leads to a higher battery consumption is apparently wrong.

However, we finally decide to use multivariable linear regression with making an assumption; We assume that the processes running are independent of each other and that the “noise” example presented above can actually be captured by the parameter “memory_free” that we already have among our measurements.

Pre-processing the data

In this step the first thing to do is that we eliminate the attribute “mobile”. Since we had no sim card inside the device we were using for experimentation, this parameter appears “False” in all data points. Thus, it presence adds no value to the analysis and delivers no information. Secondly, we decide to normalize the data, given the different numeric scale they have. Of course, before normalizing the categorical variables are converted on numeric ones of 0/1. The final data we used, are presented in the file data.csv attached to this report.

Running the algorithm and discussing the results

In the figure below there is the result of the experiment:

```
> summary(model)

Call:
lm(formula = battery_used ~ wifi + bluetooth + gps + screen_on +
    screen_brightness + memory_free + cpu_usage + cpu_frequency +
    sent_network + received_network, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3689 -2.8615 -0.4292  2.1486 18.3038

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       151.6345    23.5631   6.435 3.39e-09 ***
wifi               -1.2485     0.8908  -1.402   0.1639
bluetooth          -3.3545     1.7526  -1.914   0.0582 .
gps                 2.7957     1.0952   2.553   0.0121 *
screen_on          -1.0734     0.8516  -1.260   0.2102
screen_brightness -13.3689     1.1120 -12.023  < 2e-16 ***
memory_free       -179.0971    33.1505  -5.403 3.89e-07 ***
cpu_usage           1.7393     1.3365   1.301   0.1959
cpu_frequency      -2.4406     1.0435  -2.339   0.0212 *
sent_network       29.4980     6.5179   4.526 1.54e-05 ***
received_network   -8.5757     5.6004  -1.531   0.1286
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.439 on 109 degrees of freedom
Multiple R-squared:  0.8773,  Adjusted R-squared:  0.866
F-statistic: 77.93 on 10 and 109 DF,  p-value: < 2.2e-16
```

As we can see from the result we have now a model that gives a prediction for the discharge level of battery over time. This model should be improved, since we see that it still has quite high error rate.

REFERENCES

[1] Jar. Peak battery: Why smartphone battery life still stinks, and will for years. From techland.time.com, April 2014.

[2] Corral L.,Georgiev A.,Succi G. , and Sillitti A.,A Method for Characterizing Energy Consumption in Android Smartphones

[3] Carroll A. ,and Heiser, G., An analysis of power consumption in a smartphone.

[4] Thiagarajan N, Who Killed My Battery:Analyzing Mobile Browser Energy Consumption

[5] Ming Zhang, Hu Charlie, *Where is the energy spent inside my app?* Fine Grained Energy Accounting on Smartphones with Eprof

[6] Bakker A., Comparing Energy Profilers for Android

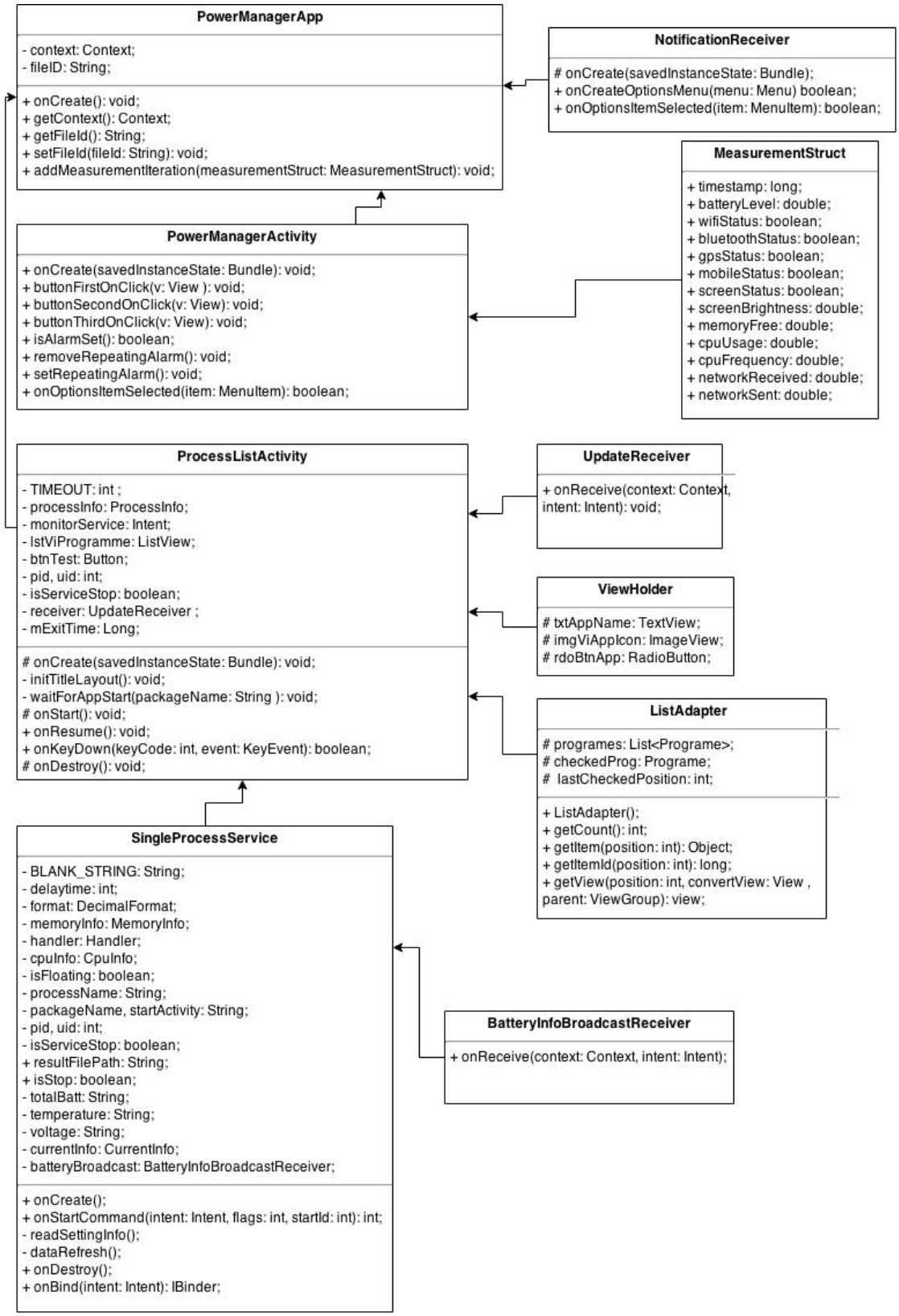
[7] Ekhiotz Jon Vergara, Joseba Sanjuan, Simin Nadjm-Tehrani,Kernel Level Energy-Efficient 3G Background Traffic Shaper for Android Smartphones

[8] Ding Li and William Halfond, An Investigation into Energy-Saving Programming Practices for Android Smartphone App Development

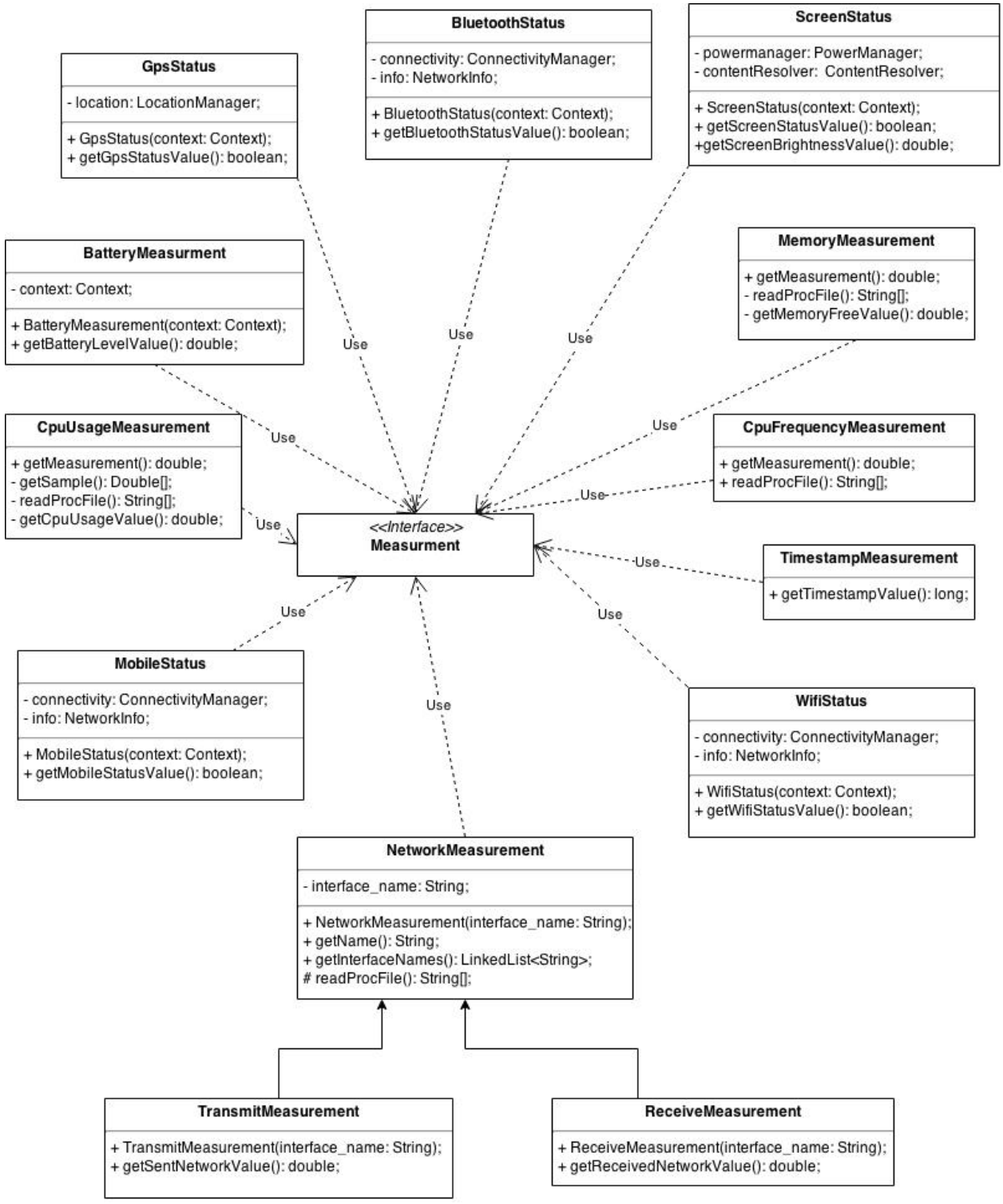
[9] Abu Moustafa,Learning from data.

UML Class diagrams

Application Class Diagram



Application Measurements Class Diagram



Single Process Measurements Class Diagram