The Approach to Predict Operational Time for Symbian Mobile Devices

Vera Kononova

Open Source Linux Lab in St. Petersburg
Electrotechnical University "LETI"
Prof. Popova 5, St.Petersburg, Russian Federation
osll@osll.spb.ru

Kirill Krinkin

St. Petersburg Electrotechnical University "LETI"
Prof. Popova 5, St.Petersburg,
Russian Federation
kkv.etu@gmail.com

Abstract

This paper describes an approach to plug-in implementation for Energy Power Management Framework on Symbian S60 Platform. Algorithm definition and common class structure are presented. They allow use the least square method in event driven environment.

Index Terms: symbian, power consumption in mobile devices, operational time prediction.

I. Introduction

Nowadays, lots of mobile development efforts are concentrated on reducing energy consumption. From one side, Industry makes more and more effective power sources and from another side user requires more and more greedy features in the mobile device. People want to use lots of applications in the same time; they want to be always on-line. The small part of this big problem is prediction of device operational time. This paper contains base idea and describes an approach to implementation mentioned algorithm with EPM Framework[1].

II. Prediction algorithm

Into the mobile device we can find two types components (activities) that are connected with energy and power management: power consumers and recharge modules.

A. What's going on with battery?

The real power consumption function looks as next sum:

$$P_c = \sum_i p_i(t),$$

where each component $p_i(t)$ – shows how many energy need for one action. For example we can consider one phone call (with corresponded parameter set) as value of $p_i(t)$. The reasoning like that is appropriate for short events, when process not very long. In the reality, we have long processes with participated consumption function, that have different values in different time points. Obviously, on long time interval we can calculate the integral (or sum in discrete case) of power consumption values.

Look at the picture. Here we have one recharge process (blue line), two consumer's process (red, yellow), sum function (green) and first product of sum functions (brown). In the simplest case, first product of sum can be treating as rough approximation for prediction algorithm. For predict operating time and calculate power spending we will use classical statistical methods. Let prediction function will be *expected value* μ with standard deviation σ. We have to mention about recharge process. We can consider it as special power consumer with negative consumer function. It's quite convenient, because we can add a several recharge modules (i. g.

Sun battery). To understand how our model matches to the base statistical hypothesis we should compare deviation with $\pm 3\sigma$ interval. It's a way to be sure about predictions.

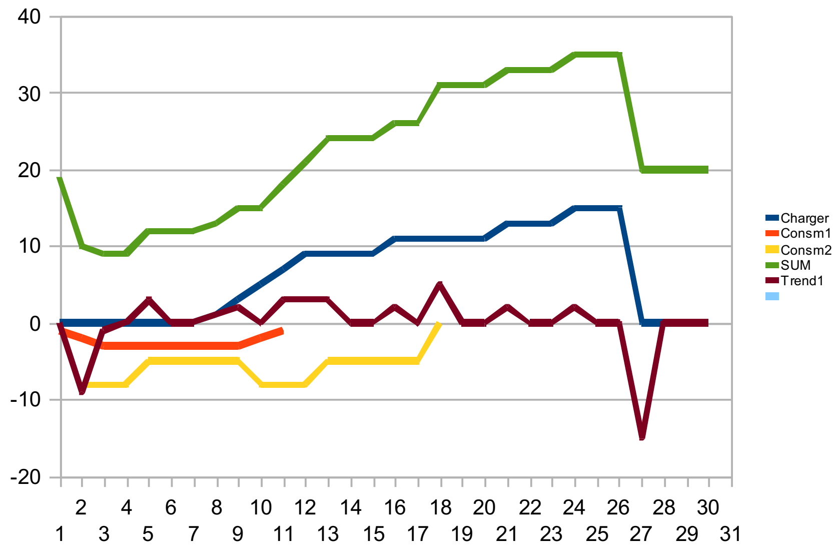

Figure 1. Power consumers and recharges

B. Algorithm definition

The main idea for operation time prediction algorithm based on previous reasoning. The first conclusion from problem analysis is we have to:

- define elementary consumption function for each process in system.
- define result prediction as sum of elementary consumption functions.

The First, there is a difficulty to choose appropriate form of curve for each consumption process. We have to use rough approximation and inject some learning strategy in each component of consumption sum.

The Second, we have to recognize what kind of environment is where device operate just now. There are lots of options: home, work, local network, foreign network and so on. We suggest defining special frequency characteristic for each elementary consumption process. For example, the frequency of data modem using grows up where customer is being in a business trip and phone calls frequency goes down in the same time. The important point is we rely on assumption the one time process structure (and consuming function) is not depend on frequency characteristics. The conclusion from this is we'll define frequency function for each $p_i(t)$ and we will define its value depends on concrete environment option. So, our sum function will look as

$$P_c = \sum_i p_i(t) \cdot f_i(t),$$

where $f_i(t)$ is concrete environment frequency function.

The mobile operating time are defined as a time from ***now*** *until the power consumption function intersect the Ox axis. It can be calculated iteratively.*

III. Implementation

The figure 2 shows the plug-in logical structure. The main class that responds for calculation sum of power consume prediction (operate time) function is PowerPredictor.

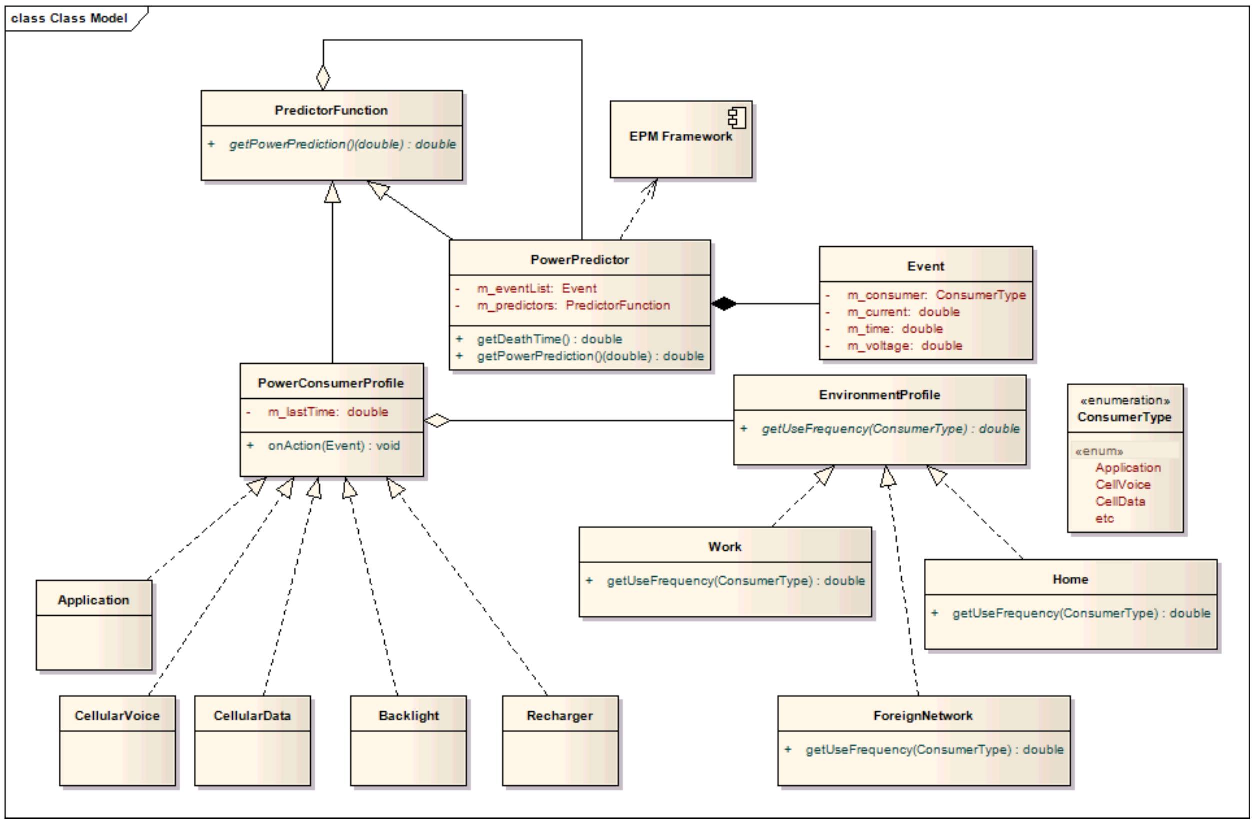
class Class Model
PredictorFunction
+ getPowerPrediction()(double) : double
EPM Framework
PowerPredictor
- m_eventList: Event
- m_predictors: PredictorFunction
+ getDeathTime() : double
+ getPowerPrediction()(double) : double
Event
- m_consumer: ConsumerType
- m_current: double
- m_time: double
- m_voltage: double
PowerConsumerProfile
- m_lastTime: double
+ onAction(Event) : void
EnvironmentProfile
+ getUseFrequency(ConsumerType) : double
«enumeration» ConsumerType
«enum»
Application
CellVoice
CellData
etc
Work
+ getUseFrequency(ConsumerType) : double
Home
+ getUseFrequency(ConsumerType) : double
Application
CellularVoice
CellularData
Backlight
Recharger
ForeignNetwork
+ getUseFrequency(ConsumerType) : double

Figure 2. Logical structure

This class has to communicate with EPM Framework and detect power events. Also, it has to be aware about battery state and overall current and voltage. It contains components of power consumption function, which was described above. PowerPredictor will do next operations:

- Receive events from EPM Framework;
- Notify components of prediction function about new events (they should be able to detect start their process for example);
- Getting partial predictions and calculate sum and time before death.

Components of power consumption functions are presented as hierarchy from PredictorFunction abstract base class. The common trivial implementation is aware about environment option, which provides frequency of elementary processes and can evaluate power consumption for it in case with repetitions. Each leaf implements concrete elementary function.

A. Programming environment

The algorithm should be implemented as plug-in to the EPM Framework. Target system is Symbian S60. EPM Framework provides next types of events and measurements:

- current and voltage measurements;
- remaining mAh in battery;
- cell ID (e.g. home, work, home city);
- foreground application and running application list;

- current cellular network voice (2G or 3G) and data usage (2/2.5/3/3.5G);
- backlight On/Off and Charger In/Out/Charging/NotCharging;
- wireless signal strengths (2G, 3G, WLAN, BT, GPS, DVB-H)
- installed apps (listed when installed).

EPM Framework provides API for all this object on Python language.

B. Algorithm tailoring and ways to improvement

We are seeing two main improvements:

- concordance predictions with real battery life;
- extension applications profiles;

The first means we need to concord sum-power prediction function with real measurements of battery health. We can use real consumption power (from current and voltage in PowerPredictor class) for correct elementary functions values. For each elementary function in sum we should to define scale coefficient with start value =1. Then after each prediction we will correct scale coefficient distribute error between all active elementary functions. This values will be processing by least squares method too.

The second improvement means we can represent power consumption process for application as polynomial of 3rd power. The free coefficients of that polynomial will be treated as parameter vector that should be used for optimization by least squares method [2].

C. Environment profile and $p_i(t)$ values

For prediction power consuming we have to be able take account the next points:

- the power consumer can be *on* or *off*;
- when consumer is off $p_i(t)$ power spending near zero;
- the frequency of repetition concrete power consumer (like Application or WLAN, or Voice..) depends on profile.

The model of elementary power consumption function is presented on the picture.

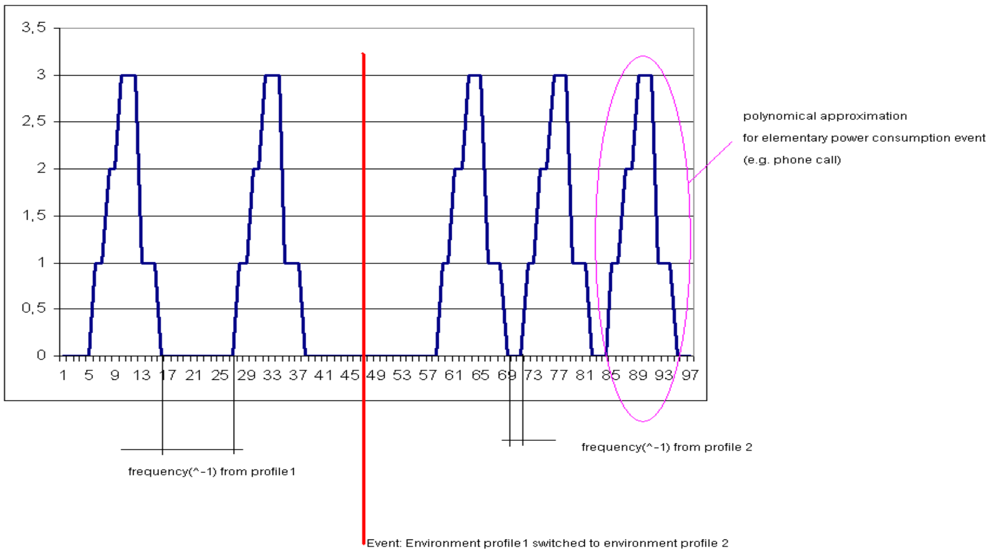

Figure 3. Application profiles

Here we have continuously power consumption function presentation. Environment profile impacts only on frequency actions (like phone call). The polynomial curve (blue humps) presents approximation for one action. The red delimiter is switch profile event: when it happens we are taking a new action frequency. In gaps between humps we have empty values (0) it means the consumer is off. This approach allows taking account to recharging frequency too. Moreover, we can define overlapped humps if it needed (e.g. for defining warm or partially sleep mode, but this investigation out of scope).

This polynomial presentation allows us to predict values next way. We will ask function value, her first and second derivative. It can be rough calculated as difference between two previous function values. The same for second derivative except we have to use first derivative instead function value. It's only several sum/sub operations. We believe it no so much for mobile processor.

According to polynomial presentation we can calculate function value for any argument. We can get user defined value for time granularity what he wants to see as accuracy for prognosis. It will minimal step for function and derivatives calculations. It's like Runge-Kutta [3] method with user defined quality.

D. The algorithm as power consumer

Of course, for power consumption prediction we need spend some power for calculation. There is a question can this algorithm function efficiently on a mobile phone, or does the implementation of the prediction algorithm itself significantly reduce the operating time by consuming CPU and battery? The answer is as described above: we are free to define accuracy for prognosis by getting from user concrete value. The good approach is to take 80% accuracy by 20% calculations. Calculation power depends on accuracy and user has to define what power value he is going to spend for prediction.

E. What was not discussed?

In this paper we didn't mention some mathematical details. There are

- least squares calculation;
- first and second derivatives calculation on the time series;
- statistical average, variance;

It was made intentionally, because this information is general and well known.

ACKNOWLEDGMENT

The authors would like to thank Finnish-Russian University Cooperation in Telecommunications program (FRUCT), Open Source Linux Laboratory in Saint-Petersburg Electrotechnical University “LETI” (OSLL) and Nokia Research Center.

REFERENCES

[1] Petri Niska, “Graduate Seminar on Energy Awarness”, 19th September 2007, http://www.tml.tkk.fi/Opinnot/T-110.7190/2007/fall/Petri.pdf

[2] J. Wolberg, Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments, Springer, 2005.

[3] Kendall E. Atkinson. An Introduction to Numerical Analysis. John Wiley & Sons – 1989.