IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

Automated Energy Problem Diagnosis In Android Applications

A.MAHESWARI[#1], G.MUPPIDATHI[#2],R.NANDHINI[#3], G.SANTHIYA[#4]
[#]U.G SCHOLAR, DEPARTMENT OF CSE,
P.S.R. RENGASAMY COLLEGE OF ENGINEERING FOR WOMEN, SIVAKASI.
a.maheshce@gmail.com[#1],gk.mupps94@gmail.com[#2],nandhinirscse@gmail.com[#3]
gsanthiyacs.89@gmail.com[#4]
[#4]ASSISTANT PROFESSOR, DEPARTMENT OF CSE,
P.S.R.RENGASAMY COLLEGE OF ENGINEERING FOR WOMEN, SIVAKASI.

ABSTRACT

Smart Phone applications energy efficiency is vital but many android applications suffer from serious energy inefficiency problems. So we introduce this approach to find an energy problem by using Java Path Finder. It monitors sensors and wake lock operations to detect missing deactivation of sensors and wake locks. It also tracks transformation and usage of sensory data and judges whether they are effectively utilized by the application using our state sensitive data utilization metric. We built our approach as a tool, Green droid, on top of JPF. Technically we addressed the challenges of generating user interaction events and scheduling event handlers in extending JPF for analyzing android applications. Green Droid completed energy efficiency diagnosis for these applications in a few minutes. It successfully located real energy problems in these applications and additionally found new unreported energy problems that were later confirmed by developers.

Indexterm: Smartphoneapplication,Energy inefficiency,automated diagnosis, Sensor data utilization, green computing.

I. INTRODUCTION

The smartphone application market is growing rapidly.Up until July 2013,the one million Android applications on Google Play store had received more than 50 billion downloads.Many of these applications leverage smartphones rich features to provide desirable user experiences.For example Google Maps can navigate users when they hike in the countryside by location sensing. However, sensing operations are usually energy consumptive,and limited battery capacity always restrict such as an application usage.As such energy efficiency becomes a critical concern for smartphone users.Existing shows that many android applications are not energy efficient due to two major reasons.First the android framework exposes hardware operations APIs(e.g.,APIs for controlling screen brightness)although these APIs

Provide flexibility, developers;have to be responsible for using them cautiously because hardware misuse could easily lead to unexpectedly large energy waste.Second android applications are mostly developed by small teams without dedicatedquality assurance effort.The developers rarely exercise due diligence assuring energy saving.

Locating Energy problems in android applications is difficult.After studying 66 real bugs report concerning energy problem,we found that many of these problems are intermittent and only manifest themselves at certain applications states.Reproducing these energy problems is labor intensive.Developers have to extensively test their applications on different devices and perform detailed energy profiling.To figure out the root causes of energy problem they have to instrument their programs with additional code to log execution traces for diagnosis. Such process is typically time consuming.This may be explain why some notorious energy problems have failed to be fixed in a timely fashion [15][18].In this work, we set out mitigate this difficulty by automating the energy problem diagnosis process.A key research challenge for automation is the lack of decidable

IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

criteria,which allows the mechanical judgement of energy inefficiency problem. As such we started by conducting a large scale empirical study to understand how energy problem have occurred in real world smartphone applicaions.We investigate 173 open source and 229 commercial android applications.By examining the bug reports,commit logs ,bug fixing patces,patch reviews and release logs,we made an interesting observation:Although the root causes of energy problem can vary with different applications,many of them(over 60%)are closely related to two types of problematic coding phenomena:

1.1Missing sensor or wake lock deactivation

To use a smartphone sensor,an application needs to register a listener with the android OS.The listener should be unregistered when they concerned sensor is no longer being used.Similarly to make a phone stay awake for computation,an application has to acquire a wake lock from android OS.The acquired wake lock should also release as soon as they computation completes. Forgetting to unregistered sensor listener or release wake locks could quickly deplete a fully charged phone battery [5], [8].

1.2 Sensory data underutilization

Smartphone sensor probes their environment and collects sensory data.These data are obtained at high energy cost and therefore should be utilized effectively by applications.Poor sensory data utilization can also result in energy waste.For example OSMDROID a popular navigation application,may continuously collect GPS data simply to render an individual map.This problem occurs occasionally at certain applications state.Battery energy is thus consumed, but collects a GPS data failed to produce any observable user benefits.

With these findings,we propose an approach to automatically diagnosing such energy problems in android applications.Our approach explores an android applications state space by systematically executing the applications using JPF, a widely used model checker for java programs.It analyze ho sensory data are utilized at each explored state,as well as monitoring whether sensor wake locks are properly used and unregistered/released.We have implemented these approaches as on 18 KLOCextensions to JPF.The resulting tool I named Green droid.As we will show in our later evaluation Greendroid is able to analysis the utilization of location data for the aforementioned Osmdroid applications over its 120K states within 3 minutes,and successfully locate our discussed energy problem.To release such efficient and effective analysis , we need to address two research issues and two major technical issues as follows.

II.RESEARCH ISSUES

While existing techniques can be adapted to monitor sensor and wake lock operations to detect their missing deactivation, how to effectively identify energy problems arising from ineffective uses of sensory data is an outstanding challenge, which requires addressing two research issues.First, sensory data, once received by an application,would be transformed into various forms and used by different application components.Identifying program data that depend on these sensory data typically requires instrumentation of additional code to the original programs.Manual instrumentation is undesirable because it is labor intensive and error prone.Second,even if a program could be carefully instrumended there is still no ell defined metric for judging ineffective utilization of sensory data automatically. To address these research issues, we proposed to monitor on applications execution and perform dynamic data flow analysis at a byte code instruction level.This allows sensory data usage to be continuously tracked without any need for instrumenting the consumed programs.We also propose a state sensitive metric to enable automated analysis of sensory data utilization and identify those application states whose sensory data have been under-utilized.

2.1Technical issues

JPF was originally designed for analyzing conventional java programs with explicit control flows.It executes the byte code of a target java program in its virtual machine.However,android applications are event driven and depend greatly an user interactions. The android framework, which builds on hundreds of native library classes. As such, applying JPF to analyze android applications

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

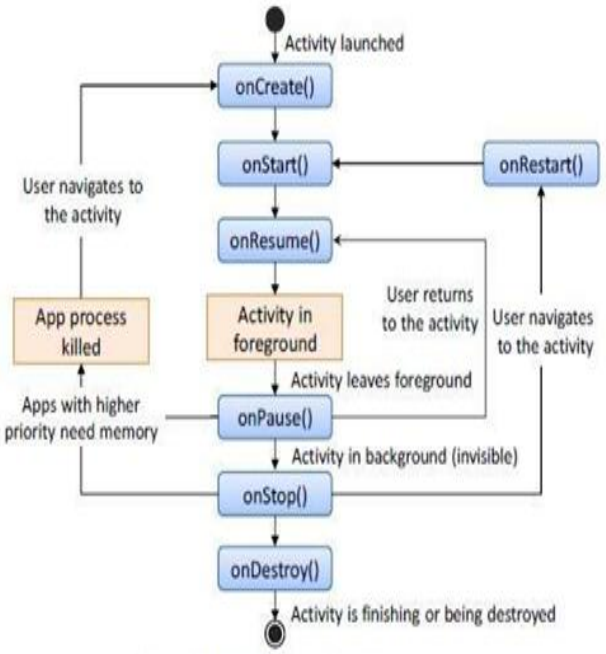
required :(1) generatingvalid their program code

Category	Number of inefficient commercial applications
personalization	59(25.8%)
Tools	34(14.8%)
Brain & Puzzles	15(6.6%)
Areca & Action	13(5.7%)
Travel & Local	11(4.8%)

comprises many loosely coupled event handlers, among which no explicit control flow is specified. At run time, these event handlers are called by user interaction events, and (2) correctly scheduling event handlers. To address the first technical issue, we propose to analyze an android applications GUI layout configuration files, systematically enumerate all possible user interaction events sequence with bounded length does not impair the effectiveness of our analysis, but instead helps quickly explore different application states and identify energy problems. To address the second technical issue, we present an application

Execution model derived from Android specifications. This model captures application generic temporal rules that specify calling relationship between event handlers. With this model, we are able to ensure an android application to be exercised with correct control flows, rather than being randomly scheduled on its event handlers. In summary, we make the following contributions in these articlesthis study identifies two major types of coding phenomena that commonly cause energy problems. We make our empirical study data public for research purpose. Weproposed state based approach for diagnosing energy problems arising from sensory data underulization Android applications. The approach systematically explores an applications state space for such diagnosis purpose

We present our ideas for extending JPF to analyze the Android applications. The analysis is based on a derived application execution model, which can also support other Android applications analysis tasks. Weimplementour toolGreen droid and evaluate it using 13 real world popular Android applications.Green Droid effectively detected 12 real energy problems that had been reported and further found two new energy problems that were later confirmed by developers.We were also invited by developers to make a patch for one of the two new problems and the patch was accepted.These evaluation results confirm Greendroid effectiveness and practical usefulness.



An activity's lifecycle diagram

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

III. PROPOSED WORK

We report our findings from an archival study of real energy problems in Android applications. For ease of permutation. We may use "energy problems and energy bugs"interchangeably in subsequent discussions.Our study aims to answer the following three research questions:

A. RQ1(Problems magnitude):

Are energy problems in Android applications serious?Do the problems have a severe impact on smartphone users?

B. RQ2(Diagnosis and fixed efforts):

Are energy problems relatively more difficult to diagnosis and fix than non-energy problems?What informationdo developers need in the energy problem diagnosis and fixing problems?

C. RQ3(Common causes and patterns):

What are common causes of energy problems?What patterns can we distill from them to enable automated diagnosis of these problems?

A. Energy Efficiency Analysis

Smartphones applications energy efficiency is vital.In past several years, researches have worked on this topic mostly form two perspectives.First various design strategies have been proposed to reduce energy consumption for smartphone applications.For example MAUI [18] helped offload "energy consuming "tasks to resource rich infrastructures such as remote server's.Second different technique have been proposed to diagnosis energy problems in smartphones applications [19].Proposed to use power signatures based on system hardware states to detect energy greedy malware conducted the first study of energy bugs in smartphone applications and proposed to use reaching definition dataflow analysis algorithms to detects no sleep energy bugs which can arise form mishandling of power control APIs in Android applications.

B. Energy Consumption Estimation

One major reason why to so many smartphone applications are not energy efficient is that developers lack viable tools to estimate energy consumption for their applications.Extensive research has been conducted to address this topic.Power Tutor uses system level power consumption models to estimate the energy consumed by major system components (e.g. display) during the execution of Android application consider an application component for continually uses collected GPS data to render a map for navigation. This component can consume a lot of energy and thus be identified as ahotspot.However although the energy cost can be high, thiscomposes is evitable in that it produce great benefits for its users by smartnavigation. As such developers may not have to optimize it. Based on this observation our GreenDroid work helps diagnosis whether certain energy consumed by sensing operations can produce corresponding benefits (i.e., high sensory data utilization).this can help developers make wise decisions when they face the choice of whether or not to optimize energy consumption for certain application components. For example if they find that at some states sensing operations are performed frequently but thus collected optimizing such sensing mechanisms to save energy as GeohashDroid developers.

C. Information Flow Tracking

Dynamic information flow tracking (DFT for short) observes interesting data as they in a program execution.DFT has many useful applications. For example TaintCheck uses DFT to protect commodity software from memory corruption attacks such as buffer overflows. It tints input data from untrustworthy sources and ensures that they are never used in a dangerous way.Taint Droid prevents Android applications from leaking users' private data. It tracks such data from privacy sensing sources and warms users when these data leave the system.LEAKPOINT leverages DFT to pinpoint memory leaks in C and C++ programs. It taints dynamically allocated blocks and monitors

IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

them in case their release might be forgotten .Our Green Droid work demonstrates another applications of DFT.we showed that DFT can help track propagation sensory data such that their utilization analysis against consumption can be conducted to detect potential energy problems in smartphones applications.

IV. CONCLUSIONS

Release Wake locks could quickly deplete a fully charged phone battery and we will shoe in our later evaluation GreenDroid is able to analyze the utilization of location data for the OsmDroid applications over its 120K states within three minutes.

V. RESULTS

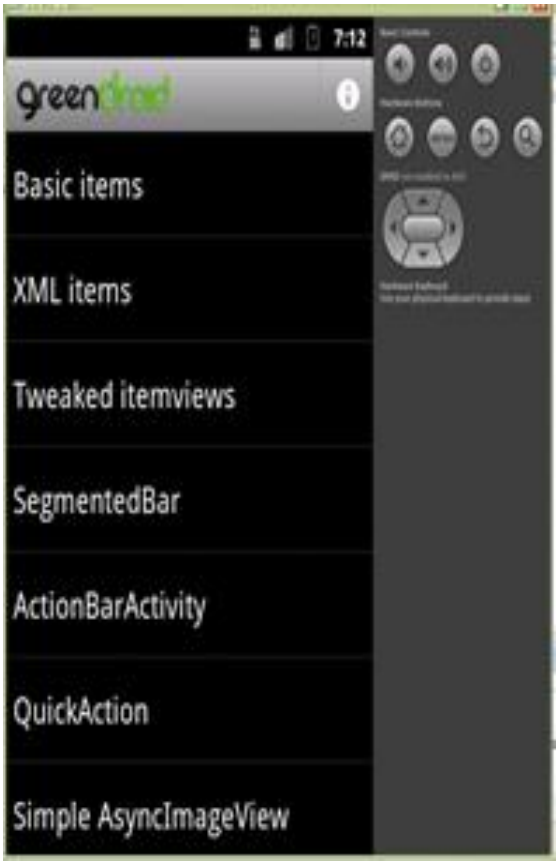


Fig.1 User interaction analysis

Fig2: Event handler scheduling

Fig3: API Modeling

VI.REFERENCES

1. S. Anand, M. Naik, M. J. Harrold, and H. Yang, "Automated concolic testing of smartphone apps," Proc. ACM SIGSOFT 20th Int'l Symp. Foundations of Soft. Engr. (FSE 12), ACM, 2012, pp. 59:1-59:11.
2. "Android Activity Lifecycles." URL: http://developer.android.com/ guide/components/activities.html.
3. "Android Developers Websites." URL: http://developer.android.com/ index.html.
4. "Android Emulator." URL: http://developer.android.com/tools/help/ emulator.html.
5. "Android Sensor Management." URL: http://developer.android.com/ reference/android/hardware/SensorManager.html.
6. "Android Platform Versions." URL:http://develpersandroid.com/reference/app//sensor.html.
7. "Android Process Lifecycle." URL: http://develpers.android.com/reference/ android/app/Activity.html#ProcessLifecycle.
8. "Android Power Management." URL: http://develpers.android.com/reference/android/os/PowerManger.html.
9. "And Tweet issues 29." URL: http://code.google.com/p/andtweet/issues/ detail?id=29.
10. M. Arnold, M. Vetches, and E. Yahav,"QVM: an efficient runtime for detecting defects in deployed systems," ACM Trans. Software Engineering and Methodology, vol.21, 211, pp., 21-235.
11. T. Azim and I. Neamtiu, "Targeted and depth-first exploration for systematic testing of android apps," Proc ACM SIGPLAN Inset Conf, Object-oriented Programming systems Language &Applications (OOPSLA 13), ACM,pp.61-660.
12. J. Clause and A. Orso, "Dytan: a generic dynamic taint analysis framework,"Proc.Int'I Symp. Software Testing and Analysis (ISSTA07), 2007, pp.196-206.
13. J.Clause and A. Orso, "LEAKPOINT: pinpointing the causes of memory leaks, "Proc.Int'I Conf.Soft.engr.(ICSE 10),pp.515-524.
14. "Crawler4j."URL:https://code.google.com /p/crawler4j/.
15. "CSip Simple issues 81" URL: https://code.google.com/p/csipsimple/issues/detail?id=81.
16. "CSipSimple issues 744" URL: https://code.google.com/p/csipsimple/issues/detail?id=744.
17. "CSip Simple issues 1674" URL: https://code.google.com/p/csipsimple/issues/detail?id=1674.
18. "dex2jar." URL: https://code.google.com/p/dex2jar/.
19. T.Dillig,, E.Yahav,andS.Chandra,"The CLOSER: automating resources management (ISMM08),ACM,2010,pp,49-62.
20. M.Dong and L.Zhong,"Seasme:Self-constructive high tate system energy modeling for battery powered mobile systems,"Proc.Int.I Conf.Mobile Systems,Apliations, and services(Mobisys 11),ACM,2011,pp