EnSights: A Tool for Energy Aware Software Development

Hamza M. Alvi*, Hareem Sahar†, Abdul A. Bangash‡ and Mirza O. Beg§
Department of Computer Science
National University of Computer and Emerging Sciences, Islamabad, Pakistan
Email: *i141622@nu.edu.pk,†hareem.sahar@nu.edu.pk, ‡abdul.ali@nu.edu.pk, §omer.beg@nu.edu.pk

***Abstract*—Software developers strive to make mobile applications energy efficient because applications that consume high energy are unpopular among mobile users. Unfortunately, software developers lack knowledge and tools which would help them make their applications efficient in terms of energy. To facilitate them in this regard, we introduce EnSights, a tool that provides critical energy change information to the developer using structural properties of the software application. We test EnSights tool on different versions of three open-source android applications namely BeHe ExploreR, PDFCreator, and QKSMS. Our tool successfully estimates change in energy consumption across versions of these applications with F-scores of up to 86%.**

I. Introduction

The smartphone market has grown substantially since the launch of first iPhone in 2007[1]. According to Statista[2] one third of the world population will own smart phones in 2017 and by 2020 half of the world population will have access to the Internet. More than 86% of the smartphone users are android users[3] and this creates substantial demand for mobile applications. Currently there are about 2.8 million apps on Google Play Store[4]. The number is on rise due to the ever increasing demand for better and more innovative features leading to the development of computationally intensive applications every day. However mostly these applications are poorly rated because of high energy consumption and faster drainage of battery [13]. With mobiles now promoting huge versatility by covering more and more of our daily life tasks, mobile battery is heavily consumed causing mobile battery to quickly finish and to be recharged multiple times. Hence keeping in mind the basic need of smartphone users to have longer battery life, developers are motivated to build energy efficient applications but they have limited knowledge and tool support to reduce the energy consumption of their applications [21] and make them more popular. Moreover, they cannot measure energy consumption of written code [17], or predict how changes in code effects energy consumption.

Ideally users would like to see energy ratings of applications just as they see the popularity ratings on Play Store. Unfortunately, crowd-sourcing is not a reliable way of determining energy efficiency of mobile applications and is hence not very useful for the developers.

Figure 1 shows two different PDFCreator applications and the energy consumption of different user scenarios is measured for both of them. The first application is simple PDFCreator that lacks user friendliness but it is energy efficient. The second application, on the contrary, is user friendly and aesthetically pleasing but does not perform editing operation in an energy efficient way. We know that developers want their applications to be energy efficient, but they also want to be user friendly and have pleasing user interfaces. The developer goal is to achieve both user friendliness and energy efficiency as the application evolves. They need continuous feedback to identify which changes adversely affect energy consumption behavior of application as new features are added.

In this paper we propose a tool, EnSights, to help developers write energy efficient code. Our tool analyzes the source code to provide energy change insights to developers resulting from changes in structure of code. The structure of code is defined by popular object oriented metric suites; the Chidamber and Kermerer (CK) metrics [8], MOOD metrics [4], and Martin Package metrics [15]. EnSights monitors changes in metrics values from previous version after revision is made in any part of the source code. It then makes an estimate of energy change in the application due to these structural changes in source code. A detailed technical explanation of the EnSights plugin and its functionality is given in Section III.

The rest of the paper is organized as follows: Section II gives background of the used metrics. Section IV details the evaluation of the EnSights plugin. Section V gives a brief overview of literature related to paper. Section VI discusses why metrics effect energy consumption of application and Section VII concludes our study.

II. Background

Researchers have been using the popular Chidamber and Kermerer (CK) metric suite [8], MOOD metric suite [4], and Martin Package metric suite [15] to characterize software structure. The effectiveness of these metrics has been evaluated by Basili et al. [5] and Braind et al. [7], [6]. Earlier studies such as Gyimothy et al. [11] have shown CK metrics to be good predictors of code quality. Olague et al. [16] empirically evaluated MOOD metrics in addition to CK metrics, and

[1]https://en.wikipedia.org/wiki/History_of_iPhone

[2]https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide

[3]http://www.gartner.com/newsroom/id/3725117

[4]https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores

TABLE I
A BRIEF DESCRIPTION OF CK, MARTIN AND MOOD METRICS

Chidamber and Kemerer (CK) Metric Suite - Class Level Metrics	
Metric Name Value	**Definition**
Depth of inheritance tree (DIT)	Number of ancestors of a class
Number of immediate sub-classes of a class (NOC)	Number of direct descendants of a class
Coupling Between Object (CBO)	Number of classes coupled to a specific class
Martins Package Metric Suite - Package Level Metrics	
Metric Name Value	**Definition**
Efferent Couplings (Ce)	Classes in the package that depend on the other packages. It is an indicator of the package's independence.
Afferent Couplings (Ca)	Number of other packages that depend upon classes within the package. It is an indicator of the package's responsibility
Abstractness (A)	Ratio of the number of abstract classes (and interfaces) in the analyzed package to the total number of classes in the analyzed package.
Instability (I)	Ratio of efferent coupling (Ce) to total coupling (Ce + Ca) such that I = Ce / (Ce + Ca).
Distance from the Main Sequence (D)	Perpendicular distance of a package from,the idealized line A + I = 1.
Fernando Britoe Abreus MOOD Metric Suite - Application Level Metrics	
Metric Name Value	**Definition**
Attribute Inheritance Factor (AIF)	Number of inherited attributes divided by total attributes available in class. A class that inherits large number of methods/attributes from its ancestor class has very high MIF/AIF.

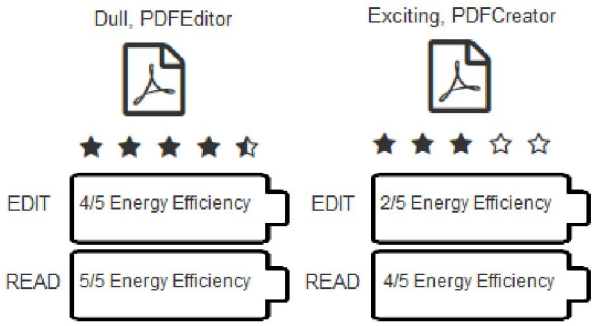

Fig. 1. Rating Applications based on Energy

showed that former do not have the ability to predict fault proneness of software. Bangash et al. [3] study the correlation of energy with these metrics and showed that change in the values of these metrics significantly affect the energy consumption of application. Some metrics however did not correlate to energy. In this paper, we use the previous correlation results published in [3] to provide the energy change information through EnSights tool. Only metrics that show significant correlation with energy are used in this tool.Table I provides a description of these metrics.

III. METHODOLOGY

EnSights is developed as a plugin for Android Studio, the widely used IDE for mobile application development. After installation user can use our plugin by clicking on Analyze button of the IDE, and selecting EnSights Metrics option as shown in Figure 2. The EnSights plugin then shows the metrics that correlate with energy consumption as determined by Bangash et al. [3]. These metrics are computed using Metrics Reloaded plugin[5] created for Android Studio. After executing the plugin the results of energy change i.e. increase or decrease are displayed, by comparing the current values of the metrics and the values of metrics from the previous run.

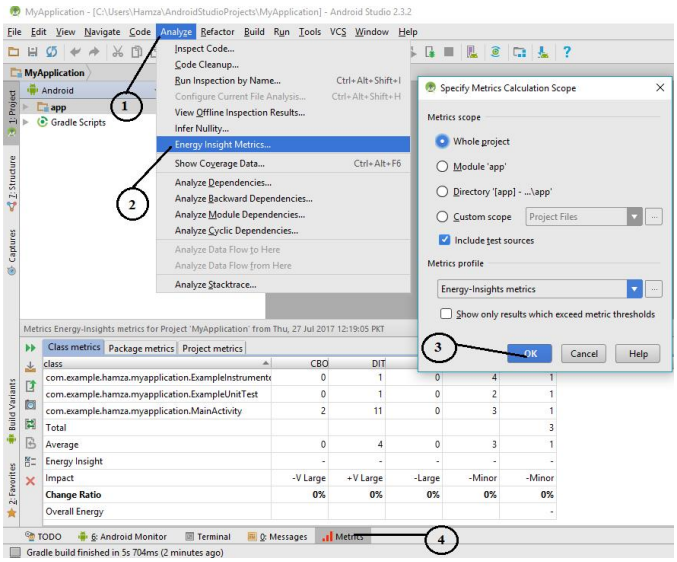

Fig. 2. A View of the EnSights Plugin

Figure 2 shows an example run of the EnSights plugin after it has been installed on Android Studio. Before clicking OK in step 3 the user can also choose whether he wants to calculate metrics for the entire project, or the module in which he is currently present, or only the directory in which he is present. It also confirms whether the user wants to add test sources in the project or not. When the user presses OK a new window named Metrics will open in the tools window of Android Studio. This window contains information about energy change and the metrics responsible for this change.

Figure 3 shows an overview of how energy change is estimated in EnSights. The approach has two basic steps; in the first step source code files of the application are provided to the

[5]https://plugins.jetbrains.com/plugin/93

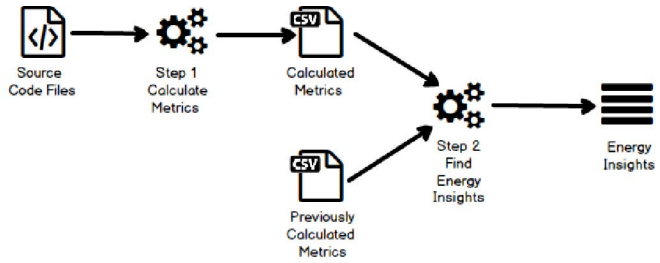

Fig. 3. An overview of EnSights

tool and structural metrics from the code are calculated. The second step takes the source code of new version, calculates new metrics values and determines the change in the two sets of metrics. The metrics change is then mapped to energy change by the EnSights tool.

To determine whether energy is increasing or decreasing EnSights uses change in magnitude of metrics as well as direction of this change along with the previously established correlation results [3] shown in Table II. The Equations 1 and 2 below shows how our tool estimates energy change. In equation 1 the change in metrics value is calculated between two versions. The correlation coefficient ($Correlation_i$) is factored into the result in Equation 2 and if the result is greater than α (-0.015) then this particular metric is contributing to an increase in energy and vice versa for a value less than α. If there is no change in the value of this metric then it does not contribute to a change in energy as shown below.

$$Change_i = CurrrentValue_i - PreviousValue_i \quad (1)$$

$$EnergyInsight_i = \begin{cases} inc & , Change_i * Correlation_i > \alpha \\ dec & , Change_i * Correlation_i < \alpha \\ - & , Change_i * Correlation_i = 0 \end{cases} \quad (2)$$

In Equation 1 and Equation 2, i represents the metric for which the calculations are being performed.

TABLE II
METRICS AND THEIR CORRELATION WITH ENERGY

	Impact	Correlation	ρ-value
CBO	V Large	-ve	0.7163
DIT	V Large	+ve	0.7028
NOC	Large	-ve	-0.5134
A	Large	-ve	0.6626
Ca	Large	-ve	0.6874
Ce	Large	-ve	0.6639
D	Minor	+ve	0.2755
I	Moderate	+ve	0.4397
AIF	Minor	+ve	0.2546

Once we know how each metric contributes to energy change, we estimate the combined effect of metrics on energy of a version. The combined effect is calculated using change ratio in metrics values. Change ratio for metric i is the ratio of $Change_i$ and previous value of metric i, as shown in Equation 3. The net effect on energy shown in Equation 4 is measured by taking sum of change ratio of each metric multiplied by its corresponding correlation and ρ values. The values of correlation and ρ for each metric are given in Table II. If the summation is greater than α (-0.015) than energy consumption of the application is said to increase and if it is less than α then energy is said to decrease. In the case when summation is equal to zero, we say that there is no change in the energy consumption of application due to changes made in project after the previous run of the plugin.

$$ChangeRatio = \frac{Change}{PreviousValue} \quad (3)$$

$$y = \sum_{i=1}^{n} \rho_i \times ChangeRatio_i \times Correlation_i \quad (4)$$

$$OverallEnergyInsight = \begin{cases} inc & , y > -0.015 \\ dec & , y < -0.015 \\ - & , y = 0 \end{cases} \quad (5)$$

Figure 4 shows a screenshot of the results after running the plugin, in a newly created empty project on Android Studio. It shows Class-level, package-level, and project-level metrics in separate tabs along with their current values and energy impact as determined by [3]. The change ratio is currently zero because the project is newly created.

$$ChangeRatioin\% = ChangeRatio \times 100 \quad (6)$$

Class metrics | Package metrics | Project metrics

class	CBO	DIT	NOC	RFC	WMC
com.example.hamza.myapplication.ExampleInstrumente	0	1	0	4	1
com.example.hamza.myapplication.ExampleUnitTest	0	1	0	2	1
com.example.hamza.myapplication.MainActivity	2	11	0	3	1
Total					3
Average	0	4	0	3	1
Energy Insight	-	-	-	-	-
Impact	-V Large	+V Large	-Large	-Minor	-Minor
Change Ratio	**0%**	**0%**	**0%**	**0%**	**0%**
Overall Energy					-

Fig. 4. Change in metrics as shown in EnSights plugin

Class metrics | Package metrics | Project metrics

class	CBO	DIT	NOC	RFC	WMC
com.example.hamza.myapplication.ExampleInstrumente	0	1	0	4	1
com.example.hamza.myapplication.ExampleUnitTest	0	1	0	2	1
com.example.hamza.myapplication.MainActivity	2	11	0	4	2
Total					4
Average	0	4	0	3	1
Energy Insight	-	-	-	dec	dec
Impact	-V Large	+V Large	-Large	-Minor	-Minor
Change Ratio	**0%**	**0%**	**0%**	**11.11%**	**33.33%**
Overall Energy					Decreased

Fig. 5. Energy Insights as shown in EnSights plugin.

Figure 5 shows screenshot after changes made to the previous version. Here the updated metrics values and change ratio of each metric can be seen. The overall energy of application can also be seen to have decreased from the previous run.

IV. Evaluation

To evaluate our tool, we used several releases of three open source iteratively developed android applications namely QKSMS, BeHe ExploreR, and PDFCreator. We extracted 22 versions of QKSMS, 5 versions of PDFCreator, and 8 versions of BeHe ExploreR from GitHub repository and calculated structural metrics of all these versions using MetricsReloaded. After gathering structural information of all versions, tool compares metric values of latest and old version of an application and estimates energy change as illustrated by methodology in Section III. At the same time, we also measure the actual energy consumed by the application for the same two versions using PowerTutor [9]. The actual energy change across these versions is then calculated and compared with estimated change in previous step. We observed that our tool correctly estimates the change in energy consumption between different versions of an application 53.13% of the time. Below we explain the detailed procedure used to evaluate our tool for two exemplary versions.

The metric values of QKSMS version 2.1.0 and version 2.5.2 are shown in Table III whereas those for version 1.0.0.2 and 2.0.1 of BeHe ExploreR are shown in Table IV.

TABLE III
QKSMS METRIC VALUES OF TWO VERSIONS

Metric	QKSMS 2.1.0 Value	QKSMS 2.5.2 Value
CBO	15.92708	15.98701
DIT	1.979167	2.118421
NOC	0.364583	0.276316
A	0.126923	0.135652
Ca	26.84615	28.52174
Ce	25.96154	27.52174
D	0.412692	0.393478
I	0.487692	0.486522
AIF	0.5519	0.5560

TABLE IV
BeHe METRICS OF TWO VERSIONS

Metric	BeHe 1.0.0.2 Value	BeHe 2.0.1 Value
CBO	6.5	5.5
DIT	3.5	2.75
NOC	0.083	0
A	0	0.023
Ca	6	12.333
Ce	8	16
D	0.43	0.353
I	0.58	0.7
AIF	93.61	0.818

EnSights uses these values to predict change in energy consumption between respective versions of both applications. At the same time we measure the actual energy consumption of the same versions using PowerTutor [9] and found it to be 0.0918mWh for QKSMS 2.1.0 and 0.0999mWh for QKSMS 2.5.2. This shows an increase from an older version of QKSMS to a newer one, while our tool also predicts an increase. The steps of calculation are are shown in Table V and Table VI for QKSMS and BeHe ExploreR respectively. The actual energy consumed by BeHe 1.0.0.2 is 1.0896mWh while BeHe 2.0.1 consumes 0.9255mWh energy. This shows a decrease in energy consumption between the two versions of BeHe ExploreR, while our tool also predicts a decrease as shown in Table VI.

TABLE V
ENERGY INSIGHTS FOR QKSMS

Energy Insights for QKSMS					
Metric	Change	Correlation	Insights	Change Ratio	ρ
CBO	0.05993	-ve	Inc	-0.15384615	0.7163
DIT	0.13925	+ve	Dec	-0.21428571	0.7028
NOC	-0.08827	-ve	Inc	-1	0.5134
A	0.00872	-ve	Dec	0.02	0.6626
Ca	1.67558	-ve	Dec	1.055	0.6874
Ce	1.56020	-ve	Dec	1	0.2755
D	-0.01921	+ve	Dec	-0.17906977	0.6639
I	-0.00117	+ve	Inc	0.206896552	0.4397
AIF	0.0041	+ve	Dec	-0.99126162	0.2546
Overall Energy		Increase			

TABLE VI
ENERGY INSIGHTS FOR BeHe

Energy Insights for BeHe					
Metric	Change	Correlation	Insights	Change Ratio	ρ
CBO	-1	-ve	Dec	0.003763	0.7163
DIT	-0.5	+ve	Inc	0.07036	0.7028
NOC	-0.083	-ve	Inc	-0.24211	0.5134
A	0.023	-ve	Dec	0.068775	0.6626
Ca	6.333	-ve	Dec	0.062414	0.6874
Ce	8	-ve	Dec	0.060097	0.2755
D	-0.077	+ve	Dec	-0.04656	0.6639
I	0.12	+ve	Dec	-0.0024	0.4397
AIF	-92.927	+ve	Inc	0.007429	0.2546
Overall Energy		Decrease			

The confusion matrix in Table VII shows the performance of our methodology in estimating change in energy consumption due to changes made in code. It is a common way of describing performance of a model on test data for which positives are known. In the matrix Irrelevant column shows those examples where prediction does not matter because no changes occurred in the values of metrics and the change in power reading that was recorded might be due to noise in gathering these readings, we consider these irrelevant examples to be true positive. The confusion matrix shows that 68.75% of the time our tool correctly estimates change in energy on the test set, while it's F-Score is 73.68%.

TABLE VII
CONFUSION MATRIX FOR ENERGY

Actual	Decreasing	Increasing	Irrelevant
Predicted			
Decreasing	8	5	0
Increasing	5	12	0
NoChange	0	0	2

Figure 6 shows F-scores for test data with respect to different metric suites (that are used in the tool and are related to energy consumption) and their combinations. This shows

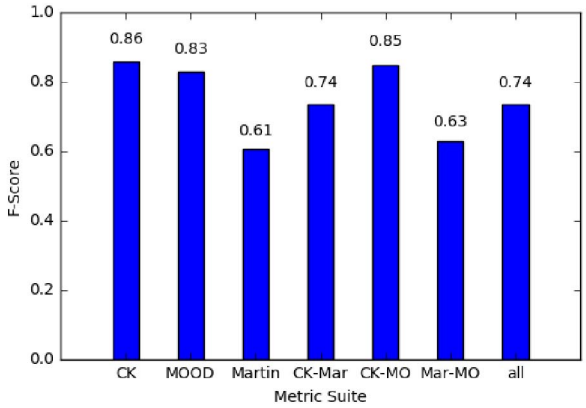

Fig. 6. F-score measure for all test executions

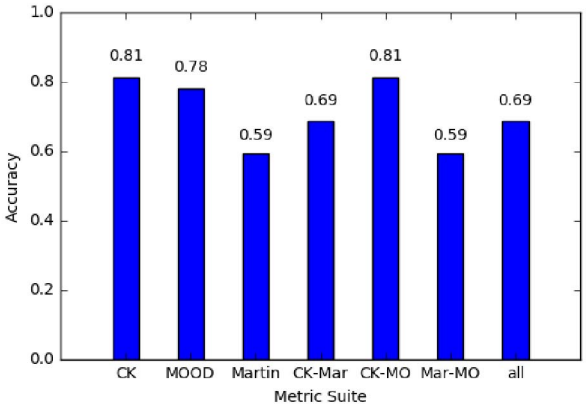

Fig. 7. Accuracy measure for all test executions

the performance of different metric suites in the formation of results.

Similarly Figure 7 shows the accuracy in estimating change in energy consumption for all test data with respect to different metric suites (that are used in the tool and are related to energy consumption) and their combinations. Based on the F-scores we can deduce that CK and MOOD metrics suites are more precise for estimating change in energy, while the Martin metrics suite is less accurate in estimating change. The impact of MOOD metrics on estimation is too small and it is highly affected by other metric suites. The impact of Martin package suite is quite high but it has less precision in estimation, so it highly impacts the results of other metric suites and hence decreases accuracy.

V. Related Work

Energy consumption of software applications has become a serious concern for mobile application developers [20], due to limited battery capacity in devices that run these applications. Several tools have been developed for energy measurement of application. Pathak et al. [19] propose a system call based power modelling approach using finite state machine. Using this technique, they provide a tool [18] that provides information related to energy consumption of the mobile application. Manotas et al. [14] propose a framework that helps developers select energy efficient libraries in their applications. A significant amount of research has been done on identifying factors that affect energy consumption of an application and constructing models to estimate energy consumption [10], [23], [19]. None of these works provide a tool support for developers that can help them to reduce energy consumption of software. Aggarwal et al. [2] analyze the pattern of system calls with energy consumption and proposed model, in which there was rule-of-thumb that any change in system call has an impact on energy consumption profile of application. In later work, they [1] built a tool using the rule of thumb in previous study that predicts changes in energy consumption using system calls of the application. Pinto et al. [22] and Hasan et al. [12] study the behavior of data structures on energy consumption with the former focusing on thread-safe implementations. Whereas our tool focuses on structural metrics of the application, it uses these metrics to predict the behavior of energy consumption.

VI. Discussion

The energy consumption is directly and very largely dependent on depth of inheritance (DIT) because if depth of inheritance of a class increases, the object of the lowest class will take a lot of memory because it will have to instantiate all the variables of above classes. Due to high memory usage it will consume high energy when it is accessed [3].

The energy consumption is indirectly and very largely dependent on coupling between objects (CBO) because if coupling between objects of a class decreases, it means that most of the work is done only by this class. This means number of currently used variables will also increase due to increased functionality and current state will become difficult to maintain, and all of this will lead to high energy consumption during context switching due to a complex state [3].

Number of children (NOC) of a class is also indirectly and largely dependent on energy consumption because lesser number of children means lesser coupling between objects. CBO is indirectly related to energy so NOC is also indirectly related to energy.

Abstractness (A) is a package level metric and effects energy consumption of the application largely. Its value range has an inverse effect on energy. When value of a package decreases, package becomes more concrete and changes related to package will have a large impact on the project.

Instability (I) directly affects energy consumption of the application and has a moderate effect on the energy. If the value of instability increases, this means that changes in package will have more effect on the application.

Distance (D) has a minor and direct effect on energy consumption of the application. If the value of distance increases, it means that package has become unbalanced for the application and changes in package will have a high impact on application.

Attribute inheritance factor (AIF) is project level metric and it affects energy consumption directly but in minor way. Increase in AIF means that the inheritance between classes in the project has increased. We discussed earlier that inheritance directly affects energy, so increase in AIF will also increase the energy consumption.

VII. Conclusion

In this paper, we present a tool EnSights which is a plugin for Android Studio and assists developers during development by providing quick insights about energy consumption of the application. These insights are based on structural changes in code which are measured through change in object oriented structural metrics; CK, Martin and MOOD. We evaluate the effectiveness of EnSights on multiple versions of three open source android applications, and found that 53.13% of the time our tool successfully predicts change in energy consumption between any two versions of applications.

References

[1] Karan Aggarwal, Abram Hindle, and Eleni Stroulia. Greenadvisor: A tool for analyzing the impact of software evolution on energy consumption. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 311–320. IEEE, 2015.

[2] Karan Aggarwal, Chenlei Zhang, Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia. The power of system call traces: Predicting the software energy consumption impact of changes. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, pages 219–233. IBM Corp., 2014.

[3] Abdul Ali Bangash, Hareem Sahar, and Mirza Omer Beg. A methodology for relating software structure with energy consumption. In *Source Code Analysis and Manipulation (SCAM), 2017 IEEE 17th International Working Conference on*, pages 111–120. IEEE, 2017.

[4] Jagdish Bansiya and Carl G. Davis. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on software engineering*, 28(1):4–17, 2002.

[5] Victor R Basili, Lionel C. Briand, and Walcélio L Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on software engineering*, 22(10):751–761, 1996.

[6] Lionel C Briand and Jürgen Wüst. Empirical studies of quality models in object-oriented systems. *Advances in computers*, 56:97–166, 2002.

[7] Lionel C Briand, Jürgen Wüst, John W Daly, and D Victor Porter. Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of systems and software*, 51(3):245–273, 2000.

[8] Shyam R Chidamber and Chris F Kemerer. A metrics suite for object oriented design. *IEEE Transactions on software engineering*, 20(6):476–493, 1994.

[9] Mian Dong and Lin Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 335–348. ACM, 2011.

[10] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane Irwin, Narayanan Vijaykrishnan, and Mahmut Kandemir. Using complete machine simulation for software power estimation: The softwatt approach. In *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pages 141–150. IEEE, 2002.

[11] Tibor Gyimothy, Rudolf Ferenc, and Istvan Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software engineering*, 31(10):897–910, 2005.

[12] Samir Hasan, Zachary King, Munawar Hafiz, Mohammed Sayagh, Bram Adams, and Abram Hindle. Energy profiles of java collections classes. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 225–236. IEEE, 2016.

[13] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E Hassan. What do mobile app users complain about? *IEEE Software*, 32(3):70–77, 2015.

[14] Irene Manotas, Lori Pollock, and James Clause. Seeds: a software engineer's energy-optimization decision support framework. In *Proceedings of the 36th International Conference on Software Engineering*, pages 503–514. ACM, 2014.

[15] Robert C Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[16] Hector M Olague, Letha H Etzkorn, Sampson Gholston, and Stephen Quattlebaum. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. *IEEE Transactions on software Engineering*, 33(6), 2007.

[17] Candy Pang, Abram Hindle, Bram Adams, and Ahmed E Hassan. What do programmers know about software energy consumption? *IEEE Software*, 33(3):83–89, 2016.

[18] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 29–42. ACM, 2012.

[19] Abhinav Pathak, Y Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168. ACM, 2011.

[20] Gustavo Pinto and Fernando Castor. Energy efficiency: a new concern for application software developers. In *Communications of the ACM*, page 60(12). ACM, 2017.

[21] Gustavo Pinto, Fernando Castor, and Yu David Liu. Mining questions about software energy consumption. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 22–31. ACM, 2014.

[22] Gustavo Pinto, Kenan Liu, Fernando Castor, and Yu David Liu. A comprehensive study on the energy efficiency of java's thread-safe collections. In *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*, pages 20–31. IEEE, 2016.

[23] Alex Shye, Benjamin Scholbrock, and Gokhan Memik. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 168–178. IEEE, 2009.