

**CORS GDR** Groupement de recherche **GPL** Génie de la programmation et du logiciel



# The road to green code (with Sonar)

## The limits to (software) growth

How it started (2011)



Mark Andreessen founder of Netscape, renowned Venture Capitalist Andreessen-Horowitz Software is eating the world, in all sectors

In the future every company will become a software company How it's going (2024)

## Loi d'erooM

•0

Effort Radicalement Organisé d'Optimisation en Masse



#### Optimiser le logiciel d'un facteur 2 tous les 2 ans

En optimisant le logiciel d'un facteur 2 tous les deux ans, on libère de la puissance informatique avec laquelle on peut inventer de nouveaux usages.

C'est comme la loi de Moore, mais **sans changer le matériel** !

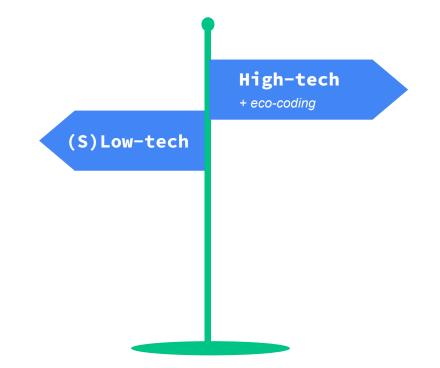
#### ©Tristan Nitot

## The software eco-\*

## SOFTWARE ECO-DESIGN

ECO-CODING ECO-WRITING CODE ECO-PROGRAMMING GREEN CODING "the most responsible software is the one we don't build"

## Fork in the road



## **First law of eco-coding**

# energy = $(more \ code)^2$

 $e = mc^2$ 

## **Energy versus Performance**

Computer/Device	Α	В
Energy (in joules)	30	20
Time (in seconds)	10	20

**Energy-efficiency vs Run-time-efficiency** 

## **Basic eco-coding incentives**

### 💰 Money

The fewer resources SmartContracts\* consume, the lower the costs

contract ERC20 is Context, IERC20, IERC20Metadata {
 mapping(address => uint256) private \_balances;

mapping(address => mapping(address => uint256)) private \_allowances;

uint256 private \_totalSupply;

string private \_name; string private \_symbol;

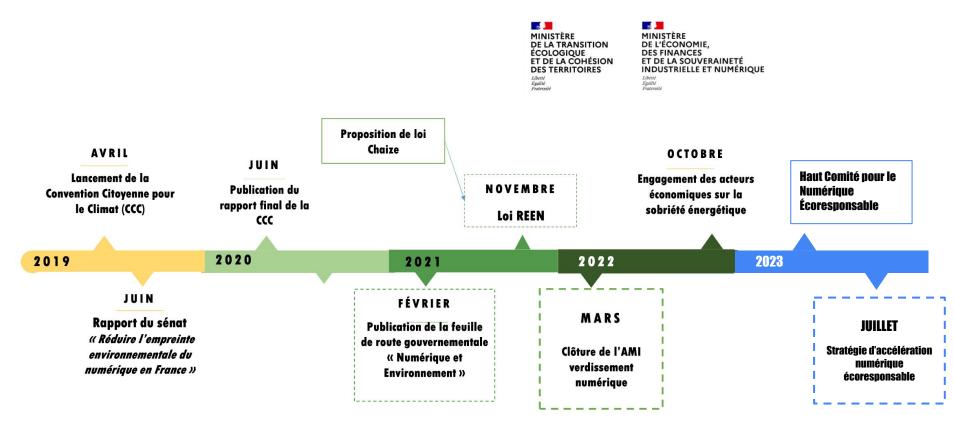


Bad reviews left on app stores can ruin your business



\*programs running within a Blockchain

## **French roadmap**



## Towards an eco-score...

#### cyber-score (effective in oct. 2023)

Laurent Lafon ? @L\_Lafon · 22 oct. 2020 Les Français ont besoin d'une information claire et lisible sur le niveau de protection de leurs données personnelles en ligne.

Ma proposition de loi visant à créer un #CyberScore que toutes les plateformes devront afficher est examinée cet après-midi au Sénat. @UC\_Senat

Afficher cette discussion



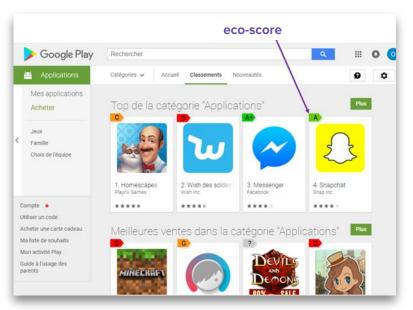
#### eco-score (elusive goal)



## What if an eco-score?

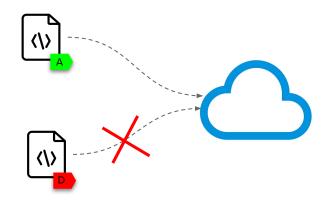
#### Information

App stores display the eco-score to the end-users (and include it in their ranking algorithm)



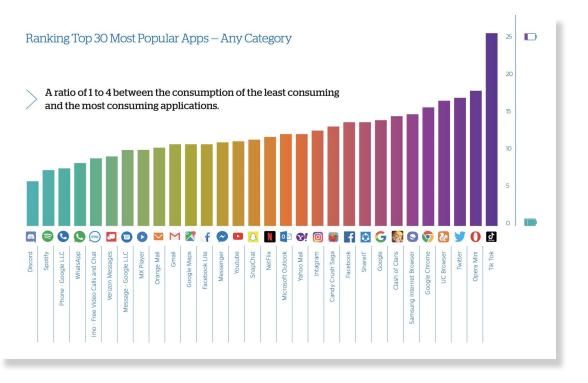
#### Regulation

(Truly sustainable) Cloud providers refuse the deployment of program lower than D



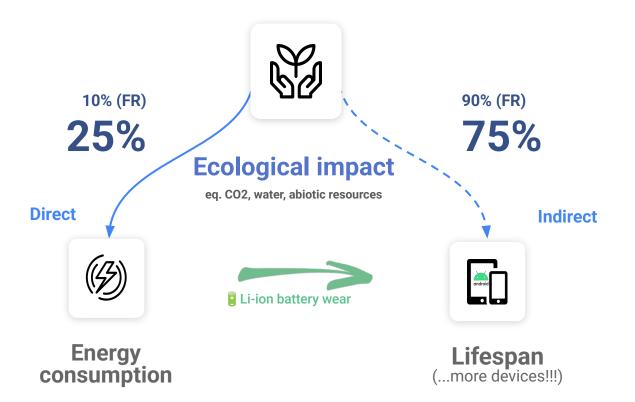
## **Ranking eco-friendly apps**





source: pCloud (2021)

## Think global & mobile-first



# Limited impacts

## **Green programming languages?**

You don't always have a choice!

What about the runtime?

Mobile apps are programs, but rarely algorithms\*

Programming language: there is no silver bullet!

	Energy (J)
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

Rui Pereira *et al.* "Ranking Programming Languages by Energy Efficiency". Science of Computer Programming, volume 205. Elsevier, 2021.

\*mathematically provable object

## **Code smells: The good old classics**

{iohnatan.oli

Code smells are

with the app. Aimin

ing the maintainabi

apply the refactorin

ware resource use,

few studies have eve

Android, This paper

smartphone resource

code smells: God C

this purpose, we se

show that refactoring

appropriate for And

of God Method had

than 47%, while the

memory consumptio

respectively, in one

munity in conductin

new tools. Also, it g

thus improving the

keywords: Code S

tion of Smartphone

1 Introduction

one of the most pop

society, especially of

people's lives [10].

platform are mains

80% of the market f

it has far surpassed

the critical factors f

is related to the eas

apps available to mi

ware development

phones and tablets,

battery, and memor

is affected by deadl

desktop, once apps :

Mobile app deve

In recent years,

- Feature Envy
- God Class
- Blob Class

. . .

- Long Method
- Long Parameter List

#### An Empirical Study on the Impact of Android Code Smells on Resource Usage

Johnatan Oliveira<sup>1</sup>, Markos Viggiato<sup>2</sup>, Mateus Santos Eduardo Figueiredo2, Humberto Marques-Neto1

<sup>1</sup>Department of Computer Science, Pontifical Catholic University of Mina <sup>2</sup>Department of Computer Science, Federal University of Minas G

Danny Dig

ABSTRACT

Understanding Code Smells in Android Applications

Umme Ayda Mannan Iftekhar Ahmed Oregon State University Oregon State University Corvallis, OR, USA Corvallis, OR, USA mannanu@oregonstate.edu ahmedi@oregonstate.edu

Carlos Jensen Oregon State University Oregon State University Corvallis, OR, USA Corvallis OR USA digd@eecs.oregonstate.edu cjensen@eecs.oregonstate.edu

violated, which may lead to long-term maintainability problems and technical debt [5, 34]. Researchers have shown that a large

Code smells are associated with poor coding practices that cause g-term maintainability problems and mask bugs. Despite mobile being a fast prowing software sector, code smells in mobile applications have been understudied. We do not know how code smells in mobile applications compare to those in deskton applications, and how code smells are affecting the design of mobile applications. Without such knowledge, application developers, tool builders, and researchers cannot improve the

We first reviewed the literature on code smells in Android applications and found that there is a significant gap between the most studied code smells in literature and most frequently occurring code smells in real world applications. Inspired by this ling, we conducted a large scale empirical study to compare th type, density, and distribution of code smells in mobile vs. desktop applications. We analyze an open-source corpus of 500 Android pplications (total of 6.7M LOC) and 750 desktop Java applications (total of 16M LOC), and compare 14,553 instances of code smells in Android applications to 117,557 instances of code smells in desktop applications. We find that, despite mobile applications wing different structure and workflow than desktop applicat the variety and density of code smells is similar. However, the distribution of code smells is different - some code smells occur more frequently in mobile applications. We also found that different categories of Android applications have different code smell distributions. We highlight several implications of our study for application developers, tool builders, and researchers.

1. INTRODUCTION

Code smells [12] identify bad design or coding practices. Code smells are not the same as bues and do not mean that the code deviates from the expected execution, rather that design rules were

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To con otherwise, or republish, to post on servers or to redistribute to lists, MORILESoft 16 May 16-17 2016 Austin TX USA Copyright 2016 ACM. ISBN 978-1-4503-4178-3...\$15.00. DOI: http://dx.doi.org/10.1145/12345.67890

http://www.eartner.com/newsroom/id/2153215

2 http://www.eartner.com/newsroom/id/3061917

Swing, nor JavarA. Many APIs are specific to mobile (Contacts, Power Management, Graphics, etc.). GUIs on Android are declared via XML. Do these differences in structure and workflow of Android applications affect the distribution of code smells? If code smells are the same, it means that all the tools and research

on code smells of desktop application applies to mobile applications. But if they are different, then application developers. ol builders, and researchers can make wrong assumptions about code smells in mobile applications. Novel tools and approaches might be needed, and the priority in developing software engineering tools might need to be revised.

Rana Abdullah M Almurshed

almurshr@oregonstate.edu

Oregon State University

Corvallis, OR, USA

number of code smells correlate with bugs [23, 30] and

maintainability problems [12]. However, according to Yamashita

et al. [42], around 32% of the developers are not aware of code

smells and their pitfalls. Moreover, on the research side, Ahmed et

al. [2] found that there is significant gap between the code smells

that receive a lot of attention in the literature and those that annear

In recent years, mobile applications have grown to become a large

part of the software industry. According to Gartner<sup>1</sup>, in 2016 more than 300 billion apps will be downloaded. Another Gartner<sup>2</sup> report

shows that in 2015, Android had more than 78% of the world

market share of smartphones. Thus, in this paper we focus on

Researchers [40] have shown that code accrues code smells during high-intensity, frequent code changes performed under time and

market demand. Android applications often have more frequent

updates and releases than desktop applications [28]. Does this mean

that Android applications exhibit more or different code smells than

Moreover, there are other important differences between mobile

and desktop applications<sup>3</sup>. Mobile applications have limited resources (e.g., memory, CPU, network, battery). The

reactive event-driven programming. Android applications have a

special structure: there is no main function, the entry points are

given by event-handlers4 such as onCreate, onResume, etc. Also,

Swing, nor JavaFX. Many APIs are specific to mobile (Contacts,

a libraries are different: Android does not have all J2SE APIs, nor

ming paradigm is also different: mobile applications use

most frequently in real-world applications

Android applications.

desktop applications

http://gamedev.stackexchange.com/questions/4288/howdifferent-is-java-for-jre-vs-java-for-android 4 http://developer.android.com/guide/components/activities. html

#### Anti-patterns and the energy efficiency of Android applications

IEEE, Rubén Saborido, Member, IEEE, Foutse Khomh, Member, IEEE, o. Member, IEEE, and Giuliano Antoniol, Senior Member, IEEE

as changed the traditional landscape of software development by introducing new challenges. e devices e.g. memory CPU network handwidth and hattery. The energy consumption of mobile archers are actively investigating the role of coding practices on energy efficiency. Recent in conflict with energy efficiency. Therefore, it is important to take into account energy efficiency app. The research community has proposed approaches to detect and remove anti-patterns (i.e., tation problems) in software systems but, to the best of our knowledge, none of these rns that are specific to mobile apps and-or considered the energy efficiency of apps. In this paper, lyzing the impact of eight type of anti-patterns on a testbed of 59 android apps extracted from act of anti-patterns in mobile apps with respect to energy efficiency; then (2) we study the impact energy efficiency. We found that then energy consumption of apps containing anti-patterns and not Int. Moreover, we find that the impact of refactoring anti-patterns can be positive (7 type of ti-patterns). Therefore, developers should consider the impact on energy efficiency of refactoring

e: Refactoring: Anti-patterns: Mobile apps: Energy consumption

in our life today. We

quality is critical. Similar

ruban cabarida infantas

An Energy-Aware Refactoring

d with the exponential is comprised of methods with low complexity and is the nobile apps [1], software result of speculation in the design and-or implementation hange in the landscape stage. Another common anti-pattern is the Blob, a.k.a., God design point of view, class, which is a large and complex class that centralizes ced in the development most of the responsibilities of an app, while using the nts related to internal rest of the classes merely as data holders. A Blob class has battery: as well as ex- low cohesion, and hinders software maintenance, making s. Moreover, traditional code hard to reuse and understand. Resource management functionality and reliabil- is critical for mobile apps. Developers should avoid antiective visual attributes, patterns that cause battery drain. An example of such antipattern is Binding resources too early class [5]. This anti-

pattern occurs when a class switches on energy-intensive time and for everything; components of a mobile device (e.g., Wi-fi, GPS) when they e the Internet, and even cannot interact with the user. Another example is the use of anking and health moni- private getters and setters to access class attributes in a class. instead of accessing directly the attributes. The Android mobile apps age as a documentation [6] strongly recommends to avoid this antifunctionality, bug-fixing, pattern as virtual method calls are up to seven times more which sometimes lead to expensive than using direct field access [6].

n [3]. This phenomenon Previous studies have pointed out the negative impact of nifested in the form of anti-patterns on change-proneness [7], fault-proneness [8], example of anti-pattern and maintenance effort [9]. In the context of mobile apps, n a class does too little, Hecht et al. [10] found that anti-patterns are prevalent pp. A Lazy class typically along the evolution of mobile apps. They also confirmed the observation made by Chatzigeorgiou and Manakos [11] and G Antoniol are with that anti-patterns tend to remain in systems through several releases, unless a major change is performed on the system.

Recently, researchers and practitioners have proposed approaches and tools to detect [12], [13] and correct [14] Málaga, Spain, E-mail: chianti-patterns. However, these approaches only focus on object-oriented anti-patterns and do not consider mobile development concerns. One critical concern of mobile apps

DOI reference number: 10.18293/SEKE2018-157

## **Code smells: The new challengers**

On the Impact of Code Smells on the Energy Consumption of Mobile Applications Fabio Palomba<sup>a</sup>, Dario Di Nucci<sup>b</sup>, Annibale Panichella<sup>c</sup>, Andy Zaidman<sup>c</sup>, Andrea De Lucia<sup>d</sup>

Abstract

1. Introduction

\*Corresponding author

Preprint submitted to Elsevier

Email addresses: sarra.habchi@uni.lu (Sarra Habchi), moha.naouel@uqam.ca

(Naouel Moha), romain.rouvoy@inria.fr (Romain Rouvoy)

<sup>a</sup>University of Zurich - Binzmuhlestre <sup>b</sup>Vrije Universiteit Brussel - Ple <sup>c</sup>Delft University of Technology - Mekel <sup>d</sup>University of Salerno - Via Giovann

#### Abstract

Context. The demand for green softwar pecially in the context of mobile devices. by battery life. Previous studies found l a strong impact on the energy consumpt Objective. Despite the efforts spent so fluence of code smells, i.e., symptoms of on the energy consumption of mobile apr Method. To provide a wider overview energy efficiency, in this paper we condu the influence of 9 Android-specific code : Android apps. In particular, we focus ou theoretically supposed to be related to no such as performance and energy consum Results. The results of the study highlight smell types, i.e., Internal Setter, Leaking Slow Loop, consume up to 87 times more smells. Moreover, we found that refacto consumption in all of the situations. Conclusions. Based on our findings.

designing automatic refactoring approact

Keywords: Code Smells, Refactoring, I

#### 1. Introduction

Energy efficiency is becoming a majo as applications performing their activiti though the problem is mainly concerned past researchers have successfully demon

Preprint submitted to Information and Softwa

Energy Refactorings for Android in the Large and in the Wild Marco Couto João Saraiva João Paulo Fernandes CISUC Universidade de Coimbra, Portugal ho. Portugal nho nt jpf@dei.uc.pt Android Code Smells: prove energy consumption has already presented promising From Introduction to Refactoring search results [4]-[13]. These results, however, have essenially been validated by testing code patterns individually and often in a small set of applications (sometimes only in one), Sarra Habchi<sup>a,1,\*</sup>, Naouel Moha<sup>b,1</sup>, Romain Rouvou<sup>c,1</sup> In this naper, we consider 11 energy-greedy code natterns btained from the literature, described in detail in Section III <sup>a</sup>University Of Luxembourg Ve conduct a study over a large-scale repository of 600+ An-<sup>b</sup>Université du Québec À Montréal roid applications to understand the frequency of occurrence <sup>c</sup>University of Lille such patterns. Within the 200+ applications where the patns were detected, we studied the impact that replacing them. dividually and combined, by their documented alternatives is on the energy consumption. Moreover, as we consider the possible combinations of the individual patterns, this sulted in 400+ refactored applications under analysis. To perform our study, we developed an extensible, fully auomated framework called Chimera, which is able to detect and Object-oriented code smells are well-known concepts in software engineering factor the patterns. Each pattern is considered individually that refer to bad design and development practices commonly observed in nd is also combined with all the other patterns. Chimera also software systems. With the emergence of mobile apps, new classes of code easures the energy consumed by an application in different mulated usage scenarios, before and after refactoring. smells have been identified by the research community as mobile-specific In summary, the main contributions of this work are: code smells. These code smells are presented as symptoms of important An analysis of how energy-greedy patterns proposed in performance issues or bottlenecks. Despite the multiple empirical studies e literature are distributed over a large-scale repository of indroid applications. This is described in Sections IV and V; about these new code smells, their diffuseness and evolution along change A reusable prototype of a pattern-oriented testing framehistories remains unclear. work (Chimera), described in Section VI-B, for the detection We present in this article a large-scale empirical study that inspects the tering, and refactoring of patterns in Android applications; can also run a set of usage scenarios on such applications, introduction, evolution, and removal of Android code smells. This study rewhile collecting metrics such as energy consumption; lies on data extracted from 324 apps, a manual analysis of 561 smell-removing ) An empirical study, described in Section VI, to assess the commits, and discussions with 25 Android developers. Our findings reveal arry impact of applying refactorings. We analyze, for each le pattern and combination of patterns, the test results for that the high diffuseness of mobile-specific code smells is not a result of re-Android applications on which they occur, and compare leasing pressure. We also found that the removal of these code smells is e obtained gains between each pattern/combination. generally a side effect of maintenance activities as developers do not refactor Using the results of the empirical study referred in 3), we im at answering the following research questions: smell instances even when they are aware of them. RQ1: Do refactorings consistently lead to energy savings? RQ2: Do all individual refactorings lead to energy savings of the same magnitude? RO3: What are the refactorings that, individually or when combined, produce the higher energy savings? Mobile apps have established themselves as mainstream software systems RQ4: When refactoring for energy efficiency, what approach should developers follow? deployed at scale. Over the last few years, they successfully invaded the

October 15 2020

Internal Setter

- Leaking Thread
- Leaking Inner Class
- Member Ignoring Method
- No Low Memory Resolver
- Hashmap Usage
- Init OnDraw

. . .

# Greater impacts

## **Android-specific matters**

Battery-killers are nestled at the platform-level, not the language-level

Every Android project has a well-defined, meaningful structure

There are lots of interesting things to inspect:



## **Energy-greedy components**

Hardware-related Component	Avg. energy consumption (J)
display	139.784567875382
camera	84.1856142588254
microphone	81.8998646885348
gravity	71.3078291080087
magnetic_field	69.6877663025097
gyroscope	69.3777997221
accelerometer	67.9535327322522
сри	66.6925401713931
room_database	66.0762976599094
speaker	65.6659164078901
gps	65.6478179873468
local_storage	64.5536233840085
ambientlight	63.0030057575923
networking	62.6477966616013

#### A Framework for the Automatic Execution of Measurement-based Experiments on Android Devices

Ivano Malavolta<sup>1</sup>, Eoin Martino Grua<sup>1</sup>, Cheng-Yu Lam<sup>1</sup>, Randy de Vries<sup>1</sup>, Franky Tan<sup>1</sup>, Eric Zielinski<sup>1</sup>, Michael Peters<sup>3</sup>, Luuk Kaandorp<sup>1</sup>
<sup>1</sup> Vrije Universiteit Amsterdam, The Netherlands. inaukoNta@vunl, e.m.graa@vunl, e.2.lam@student.vu.nl, randy.de.vries@student.vu.nl, kk.tum@vu.nl, e.a.riclinski@student.vu.nl, lcamdorp@student.vu.nl <sup>2</sup> Mzmobi. The Netherlands. mpeters@mzmobi.com

#### ABSTRACT

Conducting measurement-based experiments is fundamental for sumption, CPU, and memory usage. However, orchestrating such experiments is not trivial as it requires large balorphate code, careful setup of measurement tools, and the adoption of various empirical bet practices scattered across the literature. All together, those factors are also wing down the scientific advancement and haming experiments' replicability in the mobile offware empirical stup of measurement tools. And the adoption of the scientific advancement and the scientific advancement and haming experiments' replicability in the mobile offware empirication on native and finde duppene in a file originate distribute of the scientific and scientific advancements and the scientific science of the individual science of the scientific science of the scientific and science of the scientific science of the scientific science of individual science of the scientific science of the science in Python and it can be extended with hird-party profilers. AR has here used in more than 25 scientific science to implemented in Python and it can be extended with hird-party profilers.

AR has been used in more than 25 scientific studies primar targeting performance and energy efficiency.

#### 1 INTRODUCTION

Android is the leading mobile platform today and the majority of scientific contributions on mobile software engineering focuses on Android [1]. When dealing with quality properties like energy efficiency and performance, practitioners and researchers rely on the measurement of run-time metrics such as battery discharge. CPU and memory usage, number and type of network requests, etc. [7, 9, 10]. In this context, considerable effort and time are spent on setting up infratructures and safetaware pipelines for conducting measurement based experiments. Moreover, when available, esits induced to case regific quality moperty (e.g., energy community). This paper presents Android Rumer (AQ): a framework to asmunically accent deperiments involving Android Accent. In AR.

<sup>1</sup>https://github.com/S2-group/android-runner

Premission to make digital or hand copyon of all or gard of this work for percensal or discumme using inguest diversifier percendent languages are not made or distributed and the strength of the strength of the strength or strength of the percent strength of the strengt experiments are defined in a descriptive fashion, and then their execution is fully *automatic*, *customizable*, and *replicable*. We designed AR with the following design drivers in mind:

- Automation: after an initial configuration, the experiment can be executed without any interaction from the user;
- Incremental experiments: AR always persists the intermediate results of the experiment and, if interrupted, it is able to resume it and continue with the remaining runs;
- Usability: users define the experiment in a descriptive manner, without writing boilerplate code or knowing the internals of AR;
- Customizability: users have the possibility to include their own business logic and automated testing tools [6] at specific points within the experiment execution (e.g., before the whole experiment begins, before or after each run, etc.);
- Profiler independence: in AR, run-time measures can be collected both via hardware (e.g., the Monssoon power monitor<sup>2</sup>) and software (e.g., Trepi). Profilers can produce different data points and can interact with apps and the Android device in their own way, moreover, AR makes straightforward to use multiple profilers within a single experiment. Experiment: reollicability: evine the confluention. sub-
- Experiments replicability: given the configuration, subject apps, and available Android devices, AR can execute an already-performed experiment with low effort, even if the experiment has been performed by a third party.

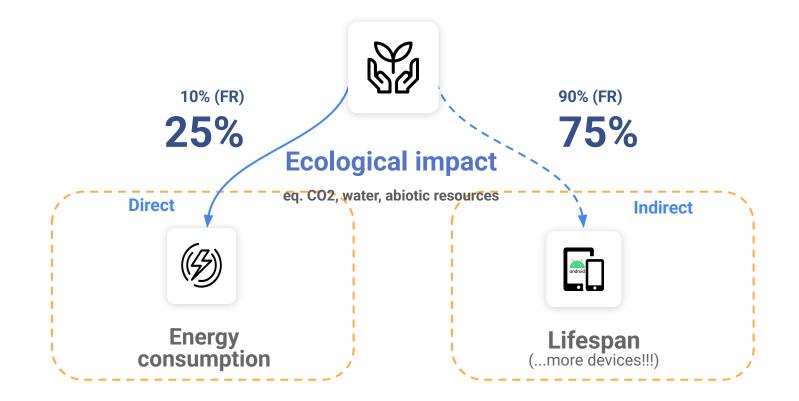
We are avance that frameworks like AR must be accurate as possible and that their accuracy must incleopendently verifiable. In order to facilitate the vialdation of AR, we created a set of 27 benchmarking apps, each of them stressing a specific hardware component of an Android mobile device, such as its accelerometer, marra, CPU, diapley (CF), etc). We enhow appon an explain basis for vialdating the accuracy of AR. The full set of benchmarking papes in sphildy syndigies (CF) sets. We enhouse appont an engular basis to independently of AR.

The target audience of AR includes (i) researchers who need to conduct empirical evaluations of software engineering methods and techniques involving Android apps. (ii) researchers developing new run-time profilers for Android devices, and (iii) practitioners needing to quantitatively assess the quality of their own apps.

The remainder of the paper is organized as follows. Section 2 provides an overview of the proposed framework, Section 3 presents its

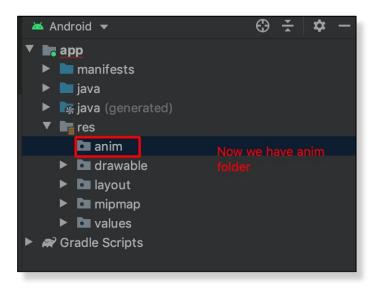
<sup>2</sup>https://www.msoon.com/high-voltage-power-monitor <sup>3</sup>https://developer.qualcomm.com/forums/software/treps-power-profiler <sup>4</sup>https://github.com/S2-group/android-apps-benchmark

## **Back to the 2 scopes**



## **Scope #1: energy consumption**

### 🔀 Avoid extraneous animation



### <sup>2</sup><sup>z</sup> Avoid keep screen on

#### •••

public class MainActivity extends Activity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity\_main);
 getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);
}





## Scope #2: device lifespan

1 Fight software obesity

### Support aging devices

#### •••

#### android { defaultConfig {

minSdkVersion 15 targetSdkVersion 33 multiDexEnabled true

## dependencies implementation "androidx.multidex:multidex:2.0.1"

### •••





# **Green code smells**

## **Open source catalog**

License CC BY-NC-ND

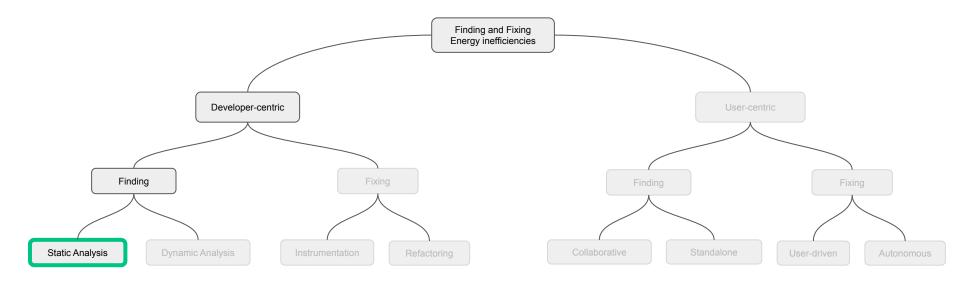
40+ code smells arranged in 9 categories, crosscutting scope 1 & 2

Description based on Java (but Kotlin-ready)

https://github.com/cnumr/best-practices-mobile

⊙ Issues 2 13 Pull requests © Discussions ⊙ Actions ⊞ Projects © Security 🗠 Insights (\$ Settings		n/cnumr/best-practices-mobile	Ba Q Å ☆	** 🗆 🖬
Insure 1       IN partneyees	arch or jump to	Pull requests Issues Codespaces Market	place Explore	¢ +
I takend     I tak				<ul> <li>★ Starred 12</li> </ul>
elements of start developers for mobile platforms who want to make their native apps more sustainable, that is at developers for mobile platforms who want to make their native apps more sustainable, that is at developers for mobile platforms who want to make their native apps more sustainable, that is at developers for mobile platforms who want to make their native apps more sustainable, that is at developers for mobile platforms who want to make their native apps more sustainable, that is at developers for mobile platforms who want to make their native apps more sustainable, that is at developers for mobile platforms who want to make their native apps more sustainable, that is the environmentally and socially acceptable, Unlike greeneral dues of thumb, this guide is focused on code smells.          Mobile       Medical developers for mobile platforms who want to make their native apps more sustainable, that is the appediem, or that them might be abate where the sustainable is that make their native apps more sustainable, that is the subject where their here might be abate where the sustainable of the appediem, or that the might be abate where the subject where their might be abate where the subject where the subject where their might be abate where the subject were their advectable by program malypic to kines (and the work, as where a could be abate where the subject were their advectable by the subject were the s	P main - P 1 bi		Go to file Add file - Code - About	۲
Contributions Under Contribution and Under Services which comparison and other Services which comparison and other services. It is difficulty commanded to maximize battery files, such as executions providers is one of the location Affeits in Google Play services which comparison providers in an other location provider is one of the location of	olegoaer Create 0	ITATION.cff		st Practices (for
LICENSE.md type in filename tate and the second second to make their native appendixes being detectable by program malping tools, such are concluded. Models (fromerly, tools, such are concluded. Mo	CITATION.cff	Create CITATION.cff	4 days ago android ios code-am	nell
Image: Contraction of the formation of the location of the loc	CONTRIBUTING.	d Update CONTRIBUTING.md	last month D Readme	
	LICENSE.md	typo in filename	Hads Internet	
RAME_md   Image: Control <ul> <li>Control</li> <li>Control</li></ul>	README.md	fixed missing Android code smells	5 days app	
Whobile-specific Best Practices         This guide is aimed at developers for mobile platforms who want to make their native apps more sustainable, that is update is aimed at developers for mobile platforms who want to make their native apps more sustainable, that is update is surface symptoms that suggest three might be a potterm, or that there might be a better way of writing the code. Therefore, these locate of the tigeral standards being detectable by program analysis tools, such as exeCode Mobile (formerly hosted by Crum).       V tage: ************************************	:= README.md		© 2 watching	
This guide is almed at developers for mobile platforms who want to make their native apps more sustainabile, that is during the could be excluded with the might be a batter word of the support of the problem. That the might be a batter word of the support of the problem. That the might be a batter word of the support of the might be a batter word of the support of the might be a batter word of the support of the might be a batter word of the batter word of the might be a batter word word of the might be a batter word of the might be a bat	i_ neromenta		₽ ¥ 3 forks	
Name         Detailed Description         Contributors 3           Optimized API         The fused location provider is one of the location APIs in Google Play services which combines signals from OPS, WF-1, and cell networks, are well as accelerometer, groscope, magnetometer and other sensors. It is dificulty recommended to maximize battery life.         State of the optimized API o	is, both environme smells, that is sur writing the code.	ntally and socially acceptable. Unlike general rules of t face symptoms that suggest there might be a problem, "herefore, these low-level best practices offer the great	their native apps more sustainable, that humb, this guide is focused on code or that there might be a better way of a daviantage of being detectable by + 1 release	
Name         Detailed Description         Optimized API           Optimized API         The fused location provider is one of the location APIs in Google Play services which combines signals from GPS, Wi-FL, and cell networks, as well as accelerometer, sproscope, magnetometer and other sensors. It is dificultly recommended to maximize battery life.         Center Seminal PRIOL           Fused Location         Trust, developer MPS software in the gravity of the Seminal PRIOL         Declemes Remuid PRIOL	is, both environme smells, that is sur writing the code. <sup>2</sup> program analysis Android	ntally and socially acceptable. Unlike general rules of t face symptoms that suggest there might be a problem, herefore, these low-level best practices offer the great ools, such as ecoCode Mobile (formerly hosted by Cnu Platform	their native apps more sustainable, that mmb, this guide is focused on code or that them right to a botter wyork advantage of being detectable by mp). Packages to packages ublished	
Optimized API  The fused location provider is one of the location APIs in Google Play services which combines signals from GPS, Wi-FL and cell networks, as well as accelerometer, sproscope, magnetometer and other sensors. It is dificulty recommended to maximize battery life. Fused Location Thus, developer also set up Google Play Service in the gradel field with a dependenty to	is, both environme smells, that is sur writing the code. <sup>2</sup> program analysis Android	ntally and socially acceptable. Unlike general rules of t face symptoms that suggest there might be a problem, herefore, these low-level best practices offer the great ools, such as ecoCode Mobile (formerly hosted by Cnu Platform	their native apps more sustainable, that numb, this guide is focused on code or that there might be a better way of advantage of being detectable by mr). Packages No packages published Pathating your first packages	
The fused location provider is one of the location API in Google Pily services which combines signation GMPS, WFL-api and entervolts, as well as accelerometric prosoche, magnetometer and other sensors. It is dificially recommended to maximize battery life. Fused location Thus, developer has to set up Google Pily Service in the gradefile with a dependency to the sensor of the sensor of th	is, both environme smells, that is sur writing the code. program analysis : Android I	Intally and accellar acceptable. Unlike general rules of the a problem, the symptom that any upget there inplies the a problem, herefore, these low-level best practices offer the great mode, such as ecoCode Mobile (formerly hosted by Cnu Platform mental Code Smells	their ratios apps more sustainable, that how how how how how how how how how how	
com.google.android.gms.location instead of the android.location package of the SDK.	is, both environme smells, that is sur writing the code. program analysis : Android I & Environn Name	Intally and accellar acceptable. Unlike general rules of the a problem, the symptom that any upget there inplies the a problem, herefore, these low-level best practices offer the great mode, such as ecoCode Mobile (formerly hosted by Cnu Platform mental Code Smells	their native apps more sustainable, that numb, this guide is focused on code of that there right has a better way of advantage of being detectable by mp). Packages Ho packages published Pathot your first package Contributors 3 Spin on Spin of the sp	
In contrast to classic Bluetooth, Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption. Its purpose is to save energy on both paired devices but very for developes are area aread of this alternature API. From the Android client side, It means append and regist, bluetosth, Lee, imports to android.bluetosth, a imports in order to benefits from Overnergy features.	is, both environme smells, that is sur writing the code. program analysis t Android I Churche and Aname Optimized API	Intelly and accellar acceptable. Unlike general rules of the a problem, there symptoms that any upget there inplies the a problem. Therefore, these low-level best practices offer the grant only, such as accoccide Mobile (formerly hosted by Cru- <b>Platform</b> <b>Intellight Code Smells</b> Detailed Description of the location A structure of the location	their native apps more sustainable, that humb, this guide is focused on code of advantage of being detectable by my). Status wells The message mathematicable packages Detection Status wells Status Wells	ppolo
Leakege	is, both environme smells, that is sur writing the code program analysis t Android I & Environn Name Optimized API Fused Location Bluetooth Low-	ntally and socially acceptable. Unlike general rules of the a problem, there syngment that suggest there night be a problem, herefore, these low-level best practices offer the great ook, such as ecoCode Mobile (former/) hosted by Cru- <b>Platform</b> <b>Internal Code Smells</b> <b>Detailed Descript</b> The fused location provider is one of the location Al combines signals from OPS, Wi-Fi, and cell network magnetometer and other sensors. It is officially rece Thus, developer has to set up Google Play Service cost, agoids, and other sensors. It is officially rece Thus, developer has to set up Google Play Service Spole. In contrast to classic Bluetooth, Bluetooth Low Erson. In contrast to classic Bluetooth, Bluetooth Low Erson significantly lower gover consumption. Its purpose devices but very few developers are aware of this a doit, Treman sequer and artiscial. Subschol, Tex Her and the treme and the sensors. It is purpose devices but very few developers are aware of this a doit, Treman sequer and artiscia. Subschol, Tex Her	their native apps more sustainable, that humb, this guide is focused on code of advantage of being detectable by my). Diameter of the sector of the sector of the sector of the sector packages Diameter of the sector of the sector of the sector of the sector packages and the sector of the sector of the sector of the sector packages and the sector of the sector of the sector of the sector packages and the sector of the sector of the sector of the sector packages and the sector of	ppolo

## **R&D** taxonomy



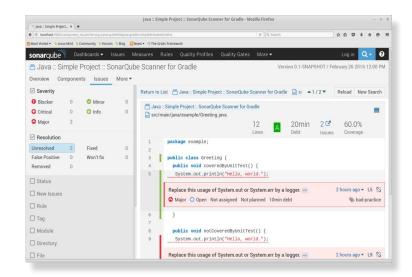
Adapted from Marimuthu C. et al., "Energy Diagnosis of Android Applications: A Thematic Taxonomy and Survey". ACM Comput. Surv. 53, 6, Article 117 (February 2021)

# Introducing ecoCode



🕏 uptodown - [C:\Users\Carlos\AndroidStudioProjects\app-uptodown-android\uptodown\app] - [app]\src\main\java\com\mediain	gea\uptodown\activities\InstallerActivityjava - Android Stu	dio 22 - 🗗 🗙
Ele Edit Yiew Mavigate Gode Analyze Befactor Build Ryn Iools VCS Window Help		
		ର 🖸
🕞 uptodown 🕞 app 🗈 sec 🗈 main 🗈 java 🗈 com 🗈 mediaingea 🗈 uptodown 🗈 activities 🥝 InstallerActivity		
🛛 Project 🔹 🕄 🕂 🖉 MainActivityjava × 🕲 Constantesjava × 🔮 Facebookjava × 🤮 InstallerActi		0
		∑radi V
R Dides Bisport		· · · · · · · · · · · · · · · · · · ·
V Dapp		
* Created by Carlos on 22/03/2015.		
Didea     Created by Carles on 22/03/2015.     Disud     Digade     Digade     Digade		
A ► D guide public class InstallerActivity extends Activity ( ▼ ► D projectifiesBackup		
80vezzide		
bit public void onCreate(Bundle savedInstanceState, Fersistabled super.onCreate(Barella tanceState, persistentState);		
Dissets tryi		
<pre>v Diwa Intent intentini = getIntent(); v Commediainge Uri uri = null;</pre>		
V El commensarige Vil VII - Halle V El activities if(intentini)=mill) (		
<pre>uri = intentIni.getData();</pre>		1
🙆 în Base		
O Facel Intent intent = new Intent(Intent.&CTION_VIEW);		
B Goog intent.setDataAndType(uri, "application/vmd.android.     B Go h Intent.setFlags(Intent.FLAG_ACTIVITY_NEW_FASH);		
G a Insta     Intelect.sectrops(Intelecting activity as Insta),     G a Man     if (android.or.Build.VERSION.SON_INT >= Build.VERSIO		
intent.putExtra(Intent.EXTRA INSTALLER FACKAGE )		
Andreid Mentor		0-1 <u>x</u>
n Motorole XT1052 Android S.1, API 22 🔻 com.mediainges.sptodown (15179) 🔻		And
A lapat Monton -		Regex Show only selected application 🔻
		Mode
8 B		1
🚳 👷 Debug  🔮 TODO 📕 🔮 Andreid Maniter 🔛 🖳 Messages 📧 Terminal		📭 Event Log 🔳 Gradie Console
Gsadle build finished in 14s ferrs (38 minutes age)		29:1 CRLF: UTF-8: Context <no context=""> 🚡 🖗</no>
🖷 🔎 🗔 🏮 👧 🜒		







## "Green as You Code" sounds good

## 2-step implementation

It takes 2 simple steps to implement Clean as You Code.



#### **Quality Gate on New Code**

A Quality Gate focused only on metrics for New Code – added or changed – prevents new issues from creeping in. Sonar sets this by default and aligns developers across the organization to deliver to that standard.



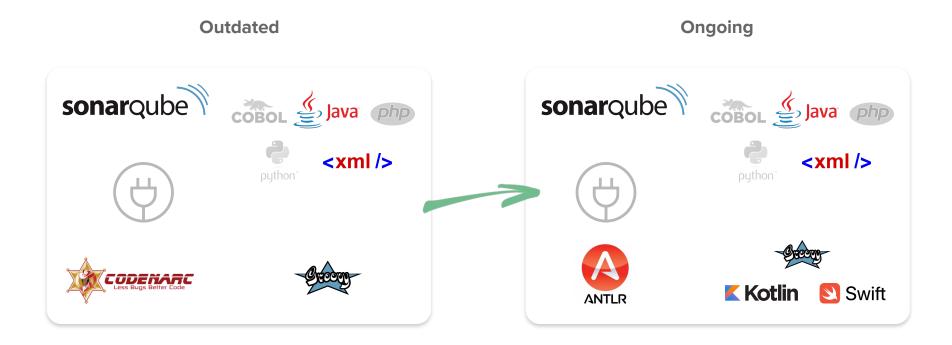
#### don't release unless it's green

The only rule that needs to be applied is the common organizational understanding that no project will be released to production if it's failing its Quality Gate.

## Screenshots\* 📀 🖉

Issues	× + ~	
$\leftrightarrow$ $\rightarrow$ C $\triangleq$ sonarqube.ed	ocode.io/project/issues?id=MyApp&resolved 🔤 ର୍ 🏠 🔲 🍮 Navigation privée 🗄 🤅	
>ecoCode Projects Issues F	ules Quality Profiles Quality Gates • Q Q Search for projects + SS	
□ MyApp ☆ 🤌 master ☺	May 12, 2022, 11:58 AM Version unspecified $ {\rm \widehat{m}}$	Projects X + V
Overview Issues Security Hotspots	Measures Code Activity	🔒 sonarqube.ecocode.io/projects 🔤 🔍 🕁 🔲 🌧 Navigation privée (2) 🚦
All My Issues	□ Bulk Change	S Projects Issues Rules Quality Profiles Quality Gates Q Q Search for projects + SS
FILTERS Clear All Filters	🕒 src/main/AndroidManifest.xml	My Favorites Perspective: Overall Status 🔹 Sort by: Name 🔹 🚊 🔍 Search I 1 projects 🏫
Yype CODE SMELL     Clear     Severity	□ Battery optimization should not be ignored. See Rule       21 hours ago ~ L8 % ~ ~	to 12 MyApp → Last analysis: 5 minutes ago
Blocker     O     Minor     1	src//univpau/uppamaps/Screens/City/CityListener.java	● I Failed (*) (A) 0 (2) (A) 6 (2) (S) 1.5k
<ul> <li>o Critical</li> <li>0</li> <li>info</li> <li>0</li> <li>∞ Major</li> <li>5</li> </ul>	Use com.google.android.gms.location instead of android.location to maximize battery life.       21 hours ago ▼ L5 % ▼         See Rule       Image: See Rule       See Rule         Code Smell ▼        Major ▼        Open ▼ Not assigned ▼ 20min effort Comment       See code environment.optimized-api ▼	1 Social Environment Java, Xml
> Scope > Resolution	Use com.google.android.gms.location instead of android.location to maximize battery life. 21 hours ago ~ L6 % ♥~ See Rule ♦ Code Smell ~ ♦ Major * O Open * Not assigned * 20min effort Comment ♦ ecocode, environment, optimized-api *	
> Status	src/fr/univpau/uppamaps/Screens/ListBuilding.java	0
<ul> <li>&gt; Security Category</li> <li>&gt; Creation Date</li> <li>&gt; Language</li> </ul>	Use com.google.android.gms.location instead of android.location to maximize battery life. 21 hours ago v L17 % 🟹 v See Rule Code Smell v 💊 Major v 🔿 Open v Not assigned v 20min effort Comment 🗞 eccode, environment, optimized-api v	əllity 1
> Rule	src//univpau/uppamaps/SplashScreen/SplashScreen.java	
> Tag > Directory	Use com.google.android.gms.location instead of android.location to maximize battery life. 21 hours ago - L8 % T-	
	Size (8) <1k	This application is based on SonarOube™ but is not an official version provided by SonarSource SA. 0

## **Technical hurdles**



## **Related works**

#### Academic

- EcoAndroid [Ribeiro et al., 2021]
- S E-Debitum [*Maia et al., 2020*]
- 🗎 xAL [Fatimaa et al., 2020]
- aDoctor [lannone et al., 2020]
- Green Android Lint [Le Goaer, 2019]

#### **Non-Academic**

- Green Software Insights [CAST, 2023]
- S EcoSonar [Accenture, 2022]
- Greensight Sonar [Capgemini, 2022]\*
- 🕥 Ecoscan [Enedis, 2020]

## **Digital commons**

Avoid reinventing the wheel every time

Open Source improves IT sustainability. ecoCode cannot but be OSS

Build a community first (e.g., through hackathons). The lines of code will follow

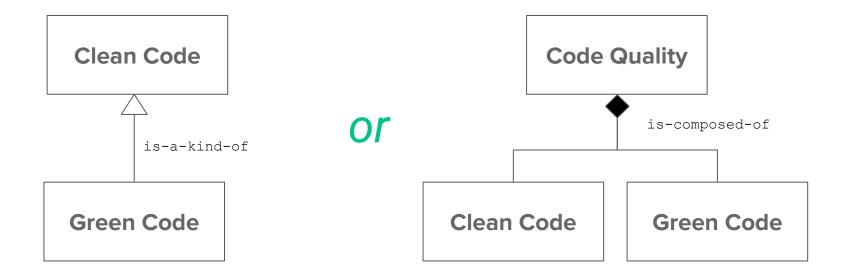
Many to watch, few to make



Green Code Initiative (GCI) https://github.com/green-code-initiative

# Food for thought

## How clean green code relate?



## **Green software supply chain**

Greening the software is noble, but greening the software supply chain too

Motto: "Wherever there's code, eco-coding is possible !"

The "Everything-as-code" is a huge potential reservoir of green code smells : infrastructure-as-code, configuration-as-code, platform-as-code, ...

## **Static analysis: The great filter**

Very few general rules of thumb withstand the filter of static code analysis ("use cache", "not too much videos", etc.)

Pro Tips: Must be rooted at syntax-level

Bottom-up approach is the preferred way to find new rules/patterns/best-practices

Energy Patterns: This is an open catalogue of energy-relate share the knowledge across all developers a We'd love to count on you to make this a tho development community. Help us spread th A visualization with prevalence and co-occu Nuw: This catalog has been accepted to th	d patterns in mobile applicat nd make mobile apps more e rough catalogue and available e word. Tweet rence of patterns can be found	tions. Our goal is to nergy efficient. to the mobile		
out the preprint.	developers 📥 🛛 🛄	Plus - Q. Search	⊕ Language ▼	Android Studio Connexion
	DOCUMENTATION			
Dark UI Colors Provide a dark UI color theme to save ba	Overview Guides UI Gr	uide Reference Samples	Design & Quality	
Dynamic Retry Delay Whenever an attempt to access a resour before asking access to that same resou	Improving performance     How to improve     performance     Baseline Profiles	Android Developers > Docs > Guides		u vous e-t-il été utile? 应 GI
Avoid Extraneous Work Avoid performing tasks that are not visib obsolete.	<ul> <li>App startup</li> <li>Guides</li> <li>Solving common problems</li> <li>App Startup</li> </ul>	Standby Sur cette page ~ Understanding Doze Doze restrictions		
Race-to-idle Release resources or services as soon a	Rendering     Memory     Battery and power	Adapt your app to Doze Understanding App Standby Using FCM to interact with your app while Support for other use cases	the device is idle	
	Optimize for doze and app standby	Testing with Doze and App Standby Testing your app with Doze		
Open Only When Necessary Open/start resources/services only when	Monitor the battery level and charging state	Starting from Android 6.0 (API level	23), Android introduces tw	o power-saving features
Push Over Poll	Monitor connectivity status and connection metering	that extend battery life for users by connected to a power source. Doze background CPU and network activi	reduces battery consumpti	ion by deferring
source	Determing and monitor docking state and type with Batterystats and Battery Historian Analyze power use with Battery Historian	periods of time. App Standby defers user has not recently interacted. While the device is in Doze, apps' a deferred until maintenance window Management Restrictions. Doze and App Standby manage the	background network activi cess to certain battery-inte s. The specific restrictions i	ity for apps with which the msive resources is are listed in Power g on Android 6.0 or higher,

experience for users, test your app in Doze and App Standby modes and make any

## **Static analysis: A noble art**

Challenges



₽ Cross-scanning



#### **Opportunities**



### 🕸 Call Graph/Control Flow Graph



## Pain point: the evaluation. But...

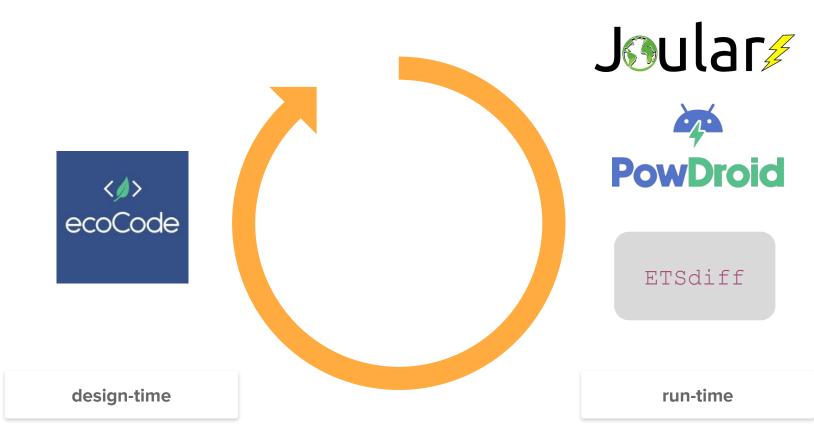
Unlike 90's OOP code smells, green code smells are still in their infancy

Do not expect green code to do what clean code has barely done

Sometimes common sense is enough

Ever-evolving mobile platforms makes things even more challenging

## **Round-trip engineering**





"Wow effect" is important to attract early-adopters

Our revamped UI was hard-coded. Tailoring the SonarQube UI to green-specific concepts would require diving deep

Developers can find green code burdensome. Gamification can help (to engage and reward)

# The end.