

Green Is the New **Clean**

Energy Code Smells for Android

Olivier Le Goaër

GT Logiciel Éco-Responsable @ GDR GPL, Juin 2021

Qualité logicielle et dette technique

Clean Code

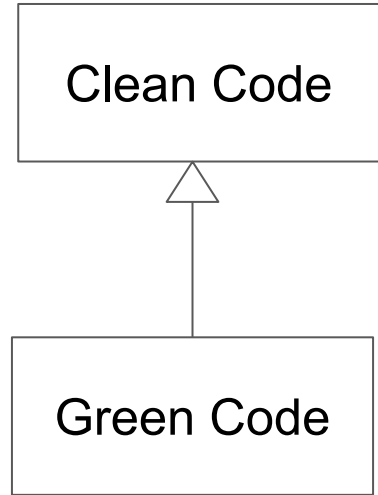
Reporter l'écriture **propre** d'un code engendre un **coût de maintenance** qui augmentera avec le temps

Green Code

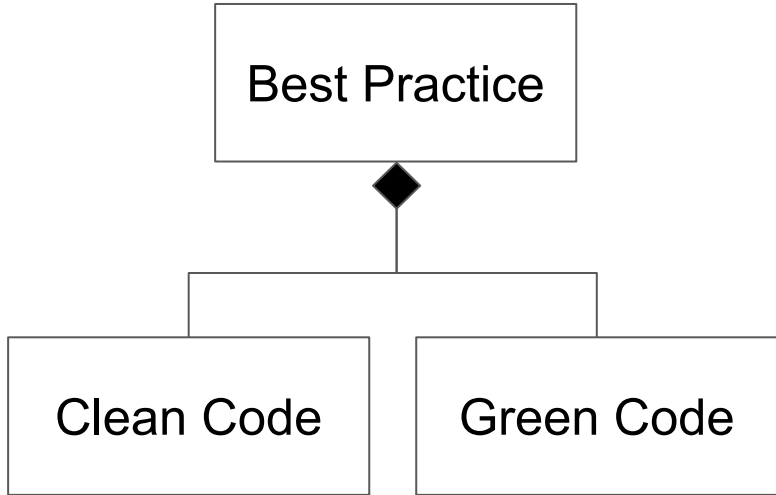
Reporter l'écriture **écologique** d'un code engendre un **coût environnemental** qui augmentera avec le temps

Dette technique & écologique

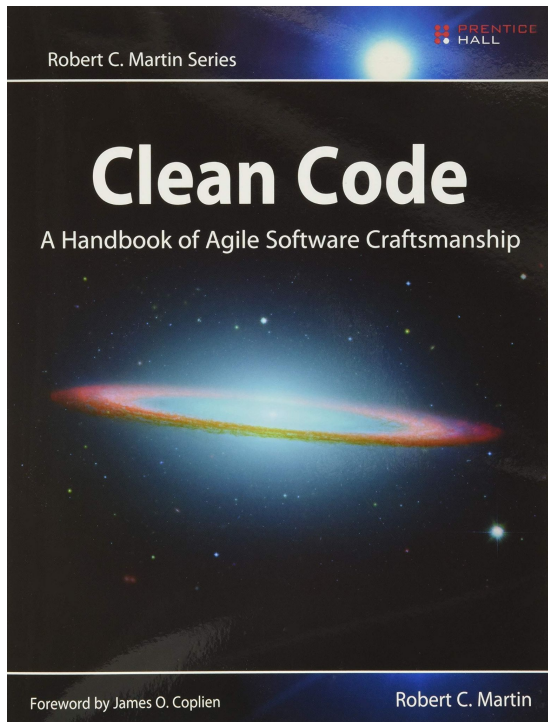
Question conceptuelle...



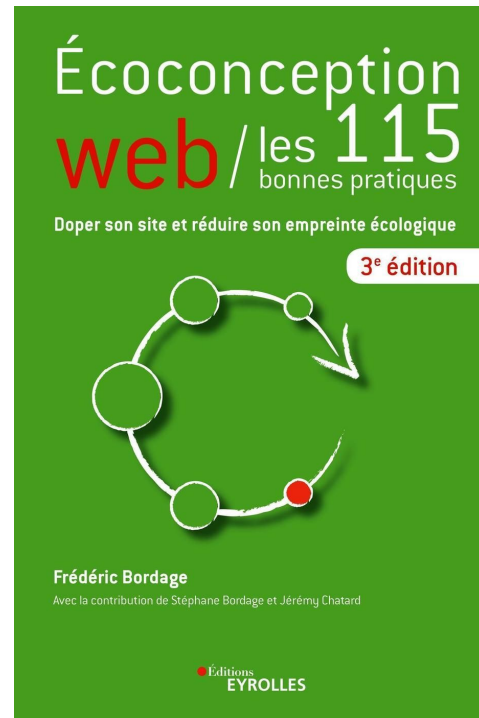
ou



Guides de bonnes pratiques



(2006)



(2015)

Enseigner le Green Code

“ Les mesures relatives à l'écoconception des services numériques sont très faibles aujourd'hui. Quand on est formé au développement, on est peu sensibilisé à « l'écologie du code » ; il est nécessaire d'avancer sur ce sujet-là. ”

Audition au Sénat de M. Cédric O, secrétaire d'État chargé de la transition numérique et des communications électroniques (Mercredi 2 décembre 2020) dans le cadre de la proposition de loi visant à réduire l'empreinte environnementale du numérique en France

[Voir le compte rendu](#)

Code Smell

“ a code smell is any characteristic in the source code of a program that possibly indicates a deeper problem ”

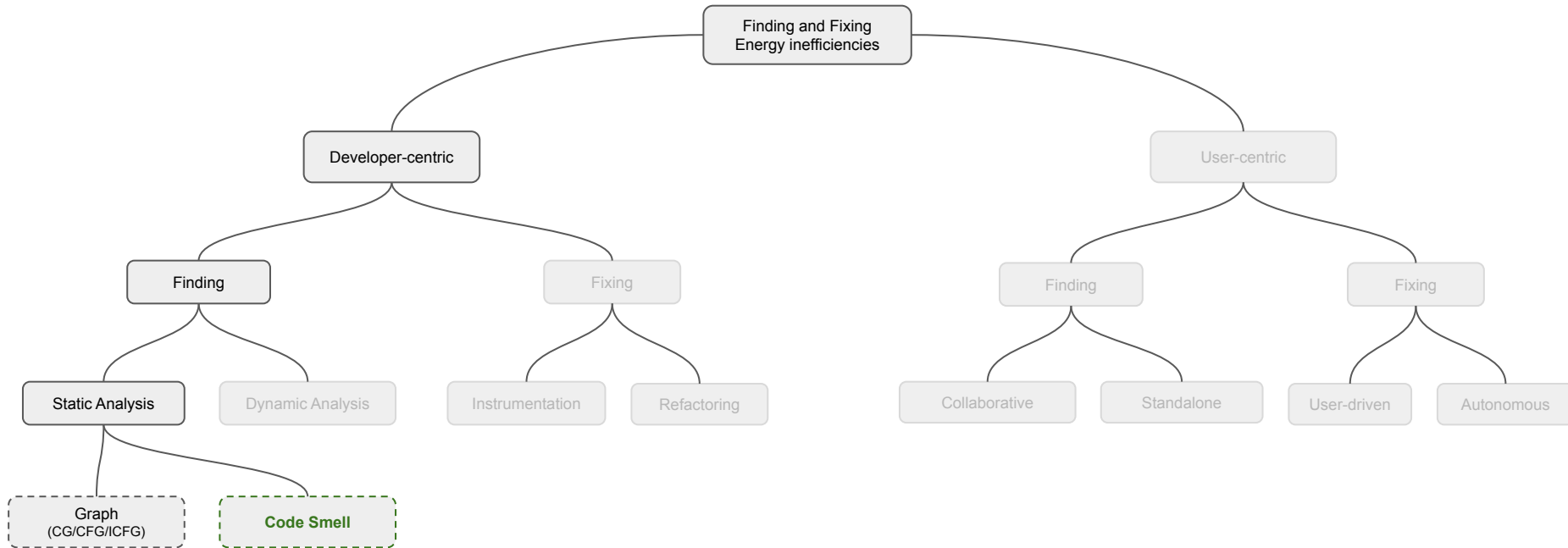
Terme popularisé par Kent Beck (1990) puis Martin Fowler (1999)

Les smells n'empêchent pas le programme de fonctionner (≠ bugs)

Les bad smells contribuent à la dette technique

(Energy) Code Smells

Research taxonomy



Green Code Checklist (Mobile)

OPTIMIZED API

Take a look to battery-efficient APIs that have been specifically designed to substitute regular APIs

LEAKAGE

Check that an acquired resource is always released, to avoid unnecessary battery drain

BOTTLENECK

Avoid accumulation of data or operations that will require an energy peak to be processed

SOBRIETY

If acceptable from a user experience viewpoint, choose the less energy-greedy alternative

IDLENESS

When the app enter in a idle state, please adapt the energy consumption accordingly

POWER

Battery-driven operations help prolong the battery life

BATCH

Grouping individually costly operations allows to save energy globally

RELEASE

Favour the compile-time tasks that improve the energy efficiency of the final app

Energy smells catalog (Android)

OPTIMIZED API (2)

Fused Location, Bluetooth Low-Energy

LEAKAGE (3)

Media Leak, Sensor Leak, Everlasting Service

BOTTLENECK (4)

Internet In The Loop, Wifi Multicast Lock, Uncompressed Data Transmission, Uncached Data Reception

SOBRIETY (6)

Dark UI, Day Night Mode, Brightness Override, Thrifty Geolocation, Thrifty BLE, Thrifty Motion Sensor

IDLENESS (6)

Keep Screen On, Keep CPU On, Durable Wake Lock, Rigid Alarm, Continuous Rendering, Keep Voice Awake

POWER (4)

Ignore Battery Optimizations, Companion in background, Charge Awareness, Save Mode Awareness

BATCH (3)

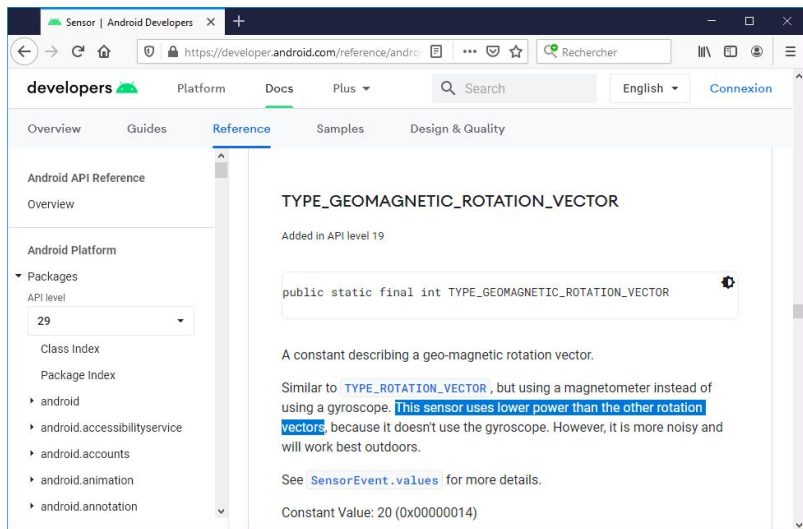
Service@Boot-time, Sensor Coalesce, Job Coalesce

RELEASE (1)

Disable Obfuscation

Sobriety > Thrifty Motion Sensor

```
mSensorManager = (SensorManager) getSystemService(Activity.SENSOR_SERVICE);  
//Accelerometer, Magnetometer, AND (when present) Gyroscope  
mRotationSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);  
mSensorManager.registerListener(this, mRotationSensor, SENSOR_DELAY);
```

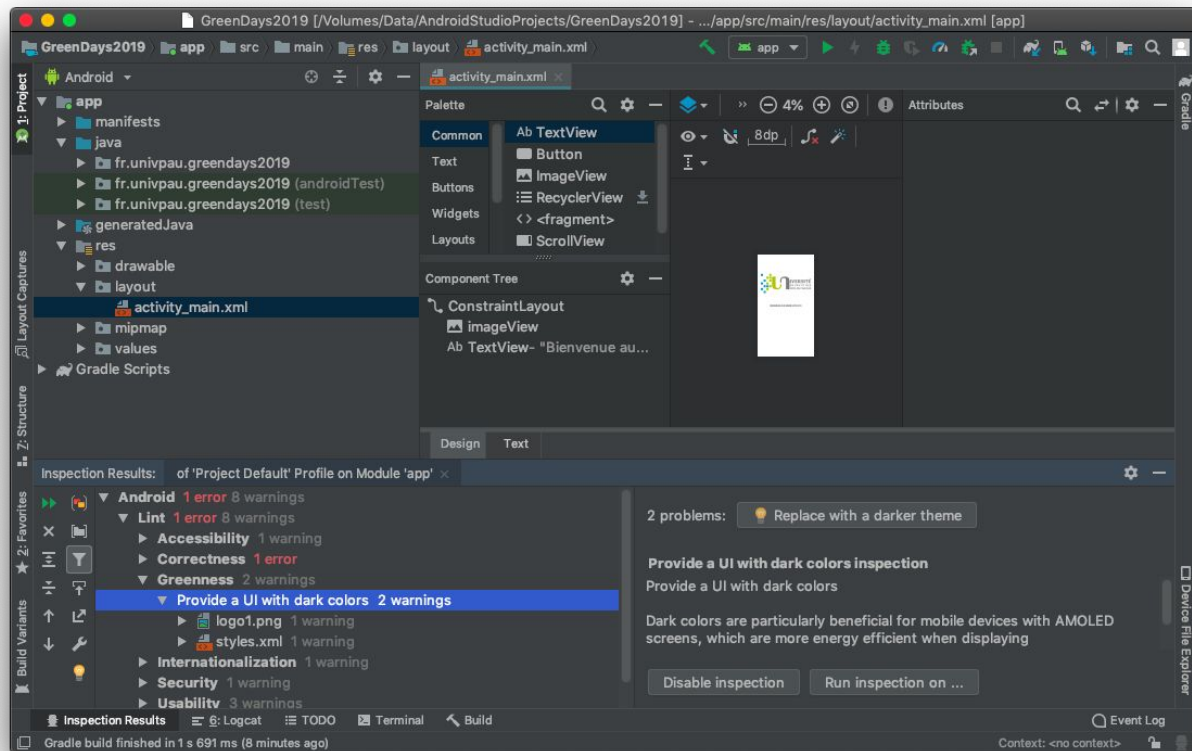


Energy smells detection

Code Scanners for Android

Embold		
Android Lint		
Detekt		
SonarQube		
SonarLint (version IntelliJ)		
Semgrep		
Codacy		
...		

Android Lint Extension (PoC 2019)



SonarQube Plugin (Beta 2021)

The screenshot shows the SonarQube Projects page. The top navigation bar includes 'ecoCode', 'Projects', 'Issues', 'Rules', 'Green Profile', and 'Green Gates'. A search bar is present. The left sidebar contains filters for 'Green Gates' (Passed, Warning, Failed), 'Environment' (A, B, C, D, E), 'Social' (A, B, C, D, E), 'Size' (XS, S, M, L, XL), and 'Languages' (Java, Kotlin). The main content area displays a list of projects with their quality status and metrics:

Project	Status	Environment	Social	Size	Language
ACME	Failed	C 1	D 2	7,3 k	JAVA
COGIP	Warning	A 14	A 8	214 k	JAVA
Cyberdyne	Passed	A 125	B 89	17 k	JAVA
PolarCred Project	Passed	A 11	A 2	512 k	JAVA
Rekall	Passed	A 122	A 7	26 k	KOTLIN
Tricatel	Passed	A 12	B 5	31 k	

The screenshot shows the SonarQube Issues page for the 'PolarCred Project (core)'. The top navigation bar includes 'ecoCode', 'Projects', 'Issues', 'Rules', 'Green Profile', and 'Green Gates'. A search bar is present. The left sidebar contains filters for 'Green Gates' (Passed, Warning, Failed), 'Environment' (A, B, C, D, E), 'Social' (A, B, C, D, E), 'Size' (XS, S, M, L, XL), and 'Languages' (Java, Kotlin). The main content area displays a list of issues with their quality status and metrics:

Issue	Status	Environment	Social	Size	Language
src/com/creed/Orientation.java	Warning	A 14	A 8	214 k	JAVA
src/com/creed/PolarBear.java	Passed	A 125	B 89	17 k	JAVA

The right sidebar shows the details of the selected issue, '1/4 issues'. It includes a description of the issue, a list of rules, and a code snippet. The code snippet shows the use of `TYPE_ROTATION_VECTOR` instead of `TYPE_GEOMAGNETIC_ROTATION_VECTOR`.

```
private void startListening(Listener listener) {
    if (mListener == listener) {
        return;
    }
    public void startListening(Listener listener) {
        if (mListener == listener) {
            return;
        }
    }
}
```

The issue is titled 'Prefer TYPE_GEOMAGNETIC_ROTATION_VECTOR instead of TYPE_ROTATION_VECTOR'. It is a 'Non compliant Code Example' and a 'Compliant Solution' is provided.

Non compliant Code Example

```
mRotationSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
```

Compliant Solution

```
mRotationSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GEOMAGNETIC_ROTATION_VECTOR);
```

Comparaison des solutions

	Audience 📢	Inspection-Time 🔍	Target languages 🛠️	Rule definition 📝	Scope 🎯	Support 🤝	Android-Specific 📱
Android Lint Extension	Intégré par défaut à Android Studio (l'IDE officiel)	on-the-fly manuel	Java + Kotlin (Universal Abstract Syntax Tree)	Java ou Kotlin	.java/.kt, .xml, .gradle, /res/	Pas documenté avant mars 2021 Communauté très limitée (Google Groups)	OUI helpers et constantes bien pratiques
SonarQube Plugin	Outil d'analyse le + populaire du marché	Lors de l'upload du code (intégrable au pipeline CI/CD)	Java (l'AST Kotlin n'est pas open source)	Java	Java quality profile Xml quality profile ...	Documentation à jour Large communauté (SonarSourcers, StackOverflow)	NON les choses sont faites "à la dure"

What's next?

Beaucoup de smells restent à découvrir

29 actuellement. 50 pour être crédible. Tout le monde peut contribuer via la plateforme : <https://www.ecocode.io/rules/new>

Nécessite une expertise Android élevée

Ne pas les confondre avec des smells de performance (pas toujours simple)

Les évolutions incessantes de l'API Android compliquent la tâche

Un travail équivalent est en cours pour identifier des “social code smells”

Beaucoup de choix sont encore empiriques

La sévérité des code smells

Blocker / Critical / Major / Minor / Info

Le coût pour y remédier (dette technique)

Trivial / Easy / Medium / Major / High / Complex
(En minutes, en heure ou en jour)

Les “quality gates” sont à définir librement pour chaque projet

Engager les équipes de développement

Comme pour le clean code, les alertes remontées par l'outil d'analyse risquent d'être perçues négativement par les développeurs...

(planet versus time-to-market)

C'est aux chefs de projet et aux lead devs de l'imposer aux équipes

(c'est la raison pour laquelle nous avons préféré un plugin SonarQube à une extension Android Lint)

On peut peut-être essayer de rendre tout cela plus “amusant” ?

(Gamification)

Beta-test Lab @ Green IT Summer School



<https://sonarqube.ecocode.io/>