Green Pauware

Eco-conception logicielle mobile et IoT

23 Octobre 2020

Olivier Le Goaër (Université de Pau/E2S)

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

e2s UPPA energy environment solutions

Snapp'

Plan de la présentation

- Le contexte
- Le projet de recherche
- L’industrialisation
- Conclusion

Le contexte

8%

Part des émissions carbonées mondiales du numérique en **2025**

(production et utilisation)

Source : “Déployer la sobriété numérique”, The Shift Project, octobre 2020

Ça bouge dans les think tank

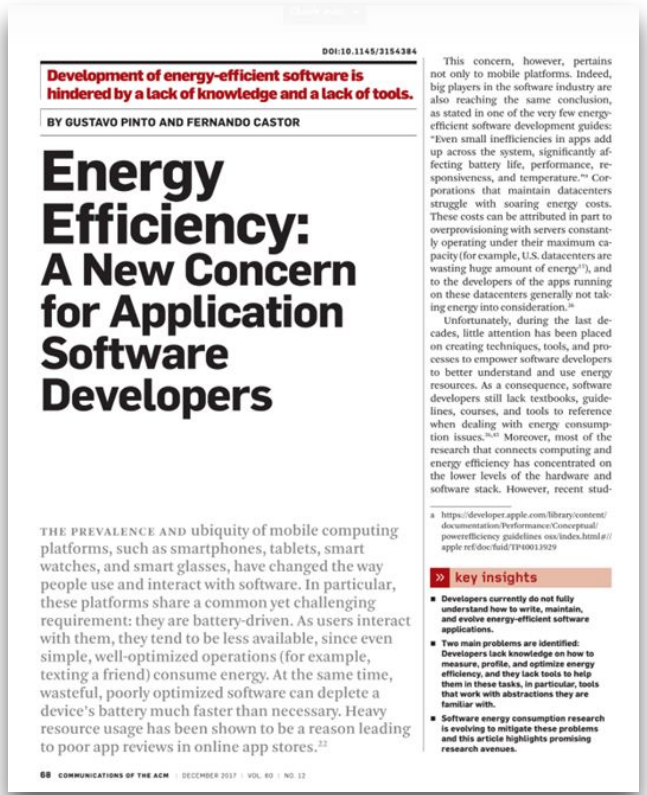
Ça bouge chez les politiques

Fabrice Brun (Député LR), a demandé au gouvernement d'obliger les éditeurs de logiciels à « pratiquer une écriture plus vertueuse, en terme environnemental, du code informatique. » (Question N° 19741, 21/05/2019)

La commission de l'aménagement du territoire et du développement durable du Sénat a lancé ce 29 janvier 2020 une mission d'information pour évaluer les impacts environnementaux du digital en France

A partir du 1er janvier 2021, les services de l'État et les collectivités territoriales devront, lorsqu'ils acquièrent ou font développer un logiciel, privilégier les logiciels à la consommation énergétique limitée (article 6 bis de la loi "anti-gaspillage" 2020)

Ça bouge chez les scientifiques



DOI:10.1145/3154384

Development of energy-efficient software is hindered by a lack of knowledge and a lack of tools.

BY GUSTAVO PINTO AND FERNANDO CASTOR

Energy Efficiency: A New Concern for Application Software Developers

THE PREVALENCE AND ubiquity of mobile computing platforms, such as smartphones, tablets, smart watches, and smart glasses, have changed the way people use and interact with software. In particular, these platforms share a common yet challenging requirement: they are battery-driven. As users interact with them, they tend to be less available, since even simple, well-optimized operations (for example, texting a friend) consume energy. At the same time, wasteful, poorly optimized software can deplete a device's battery much faster than necessary. Heavy resource usage has been shown to be a reason leading to poor app reviews in online app stores.[22]

This concern, however, pertains not only to mobile platforms. Indeed, big players in the software industry are also reaching the same conclusion, as stated in one of the very few energy-efficient software development guides: "Even small inefficiencies in apps add up across the system, significantly affecting battery life, performance, responsiveness, and temperature."[a] Corporations that maintain datacenters struggle with soaring energy costs. These costs can be attributed in part to overprovisioning with servers constantly operating under their maximum capacity (for example, U.S. datacenters are wasting huge amount of energy[43]), and to the developers of the apps running on these datacenters generally not taking energy into consideration.[36]

Unfortunately, during the last decades, little attention has been placed on creating techniques, tools, and processes to empower software developers to better understand and use energy resources. As a consequence, software developers still lack textbooks, guidelines, courses, and tools to reference when dealing with energy consumption issues.[36,43] Moreover, most of the research that connects computing and energy efficiency has concentrated on the lower levels of the hardware and software stack. However, recent stud-

a https://developer.apple.com/library/content/documentation/Performance/Conceptual/powerefficiency guidelines osx/index.html#//apple ref/doc/fuid/TP40013929

» key insights

- **Developers currently do not fully understand how to write, maintain, and evolve energy-efficient software applications.**
- **Two main problems are identified: Developers lack knowledge on how to measure, profile, and optimize energy efficiency, and they lack tools to help them in these tasks, in particular, tools that work with abstractions they are familiar with.**
- **Software energy consumption research is evolving to mitigate these problems and this article highlights promising research avenues.**

Communications of the ACM, December 2017

Romain Rouvoy "Favoriser l'émergence de logiciels plus efficients"

Date : 01 juil. 2019

Accueil > Actualités et événements > Romain Rouvoy "Favoriser l'émergence de logiciels plus efficients"

Mis à jour le 23/12/2019

À l'heure où les data centers représentent désormais plus de 10% de la dépense énergétique mondiale, la question de la consommation propre aux logiciels est plus que jamais d'actualité. C'est précisément le sujet sur lequel travaille Romain Rouvoy, enseignant chercheur au sein de l'équipe Spirals.

© Inria / Photo C. Morel

Une fois son bac en poche, Romain Rouvoy ne voulait pas faire de longues études... Passionné d'informatique, il s'est donc engagé dans un DUT, puis dans un master professionnalisant.

C'est là qu'il a été mordu par le virus de la recherche qui ne l'a plus lâché. Après une thèse et un postdoctorat de deux ans à l'université d'Oslo, il rejoint l'université de Lille en 2008, où il sera d'abord maître de conférences puis professeur.

La même année, il intègre l'équipe-projet ADAM du centre Inria Lille – Nord Europe qui deviendra Spirals *(Self-adaptation for distributed services and large software systems)* en 2012 et dont les domaines d'expertise sont les systèmes répartis et le génie logiciel.

Logiciels trop gourmands

C'est en 2010 que le jeune chercheur commence à s'intéresser à la problématique du logiciel durable. « *Le point de départ est le projet* EcoHome, *mené en collaboration avec* Orange *et* ST Microelectronics, se souvient Romain Rouvoy. *L'enjeu était de mieux comprendre la consommation des box domestiques pour envisager des pistes pour mieux la maîtriser.* » Cette première expérience sert de révélateur : l'équipe et le chercheur prennent conscience de l'importance de la part imputable au logiciel dans les dépenses énergétiques globales des systèmes informatiques.

« La consommation liée au matériel en lui-même est un sujet de préoccupation qui n'a fait que croître ces dernières années et les fabricants ont su se mettre en mouvement, par exemple en concevant des processeurs beaucoup moins gourmands que leurs prédécesseurs. Mais il n'en va pas de même pour les couches logicielles, en particulier dans l'univers des hébergeurs et du cloud computing où la consommation peut être l'une des composantes du business model *des entreprises : plus elles consomment et plus la facture augmente ! »*

INRIA, Juillet 2019

Etre prêt...

Injonction de plus en plus forte de la société

Evolutions règlementaires inévitables

L’industrie logicielle, l’une des plus grosse du monde,

va devoir faire sa part dans le “green deal”

Le projet de recherche

Eco-conception logicielle

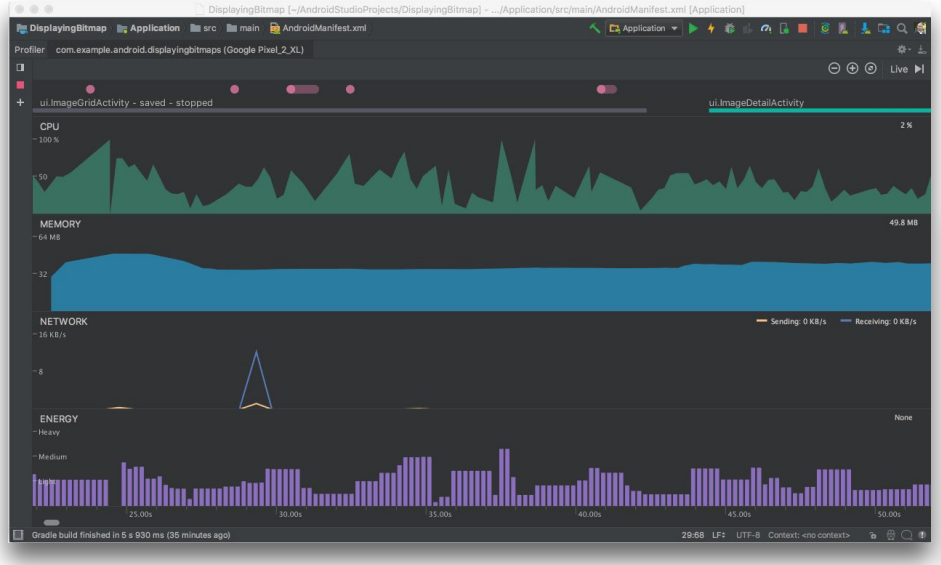
Appliquer le concept du **green-by-design** en favorisant l'écriture de code plus efficient énergétiquement : le **green code**

Les **plateformes mobiles** (Android/iOS) sont une cible de choix compte tenu du volume des applications déployées à travers le monde

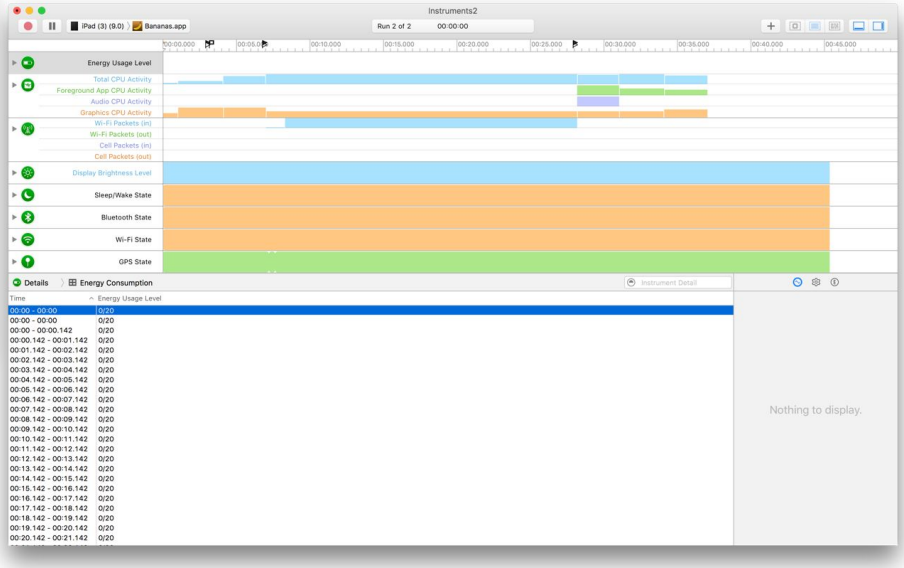
Fournir une suite d'outils facilement intégrables dans les projets **réels**

Et aussi : conseil et accompagnement des éditeurs de logiciels, sensibilisation des étudiant·e·s et des utilisateurs finaux, programme d'éco-labellisation, etc.

Trouver les racines du mal n’est pas simple...



Android Energy Profiler (Android Studio)



iOS Energy Profiler (Xcode)

Agir à son niveau avec le green code

Ecriture vertueuse de programmes dans le framework Android (APIs)

Empreinte énergétique des langages de programmation (Java*/Kotlin) et de leur runtime (JIT compiler)

Optimisations de Android OS (Doze Mode, Adaptive Battery)

* Rui Pereira et al., Energy efficiency across programming languages: how do energy, time, and memory relate? 10th ACM SIGPLAN International Conference, 2017

Odeurs de code pour l’efficience énergétique

Les odeurs de code (**code smells**) sont les symptômes de choix de conception qui peuvent impacter la qualité du logiciel

L’efficience énergétique est un **attribut de qualité** des apps mobiles

L’impact peut être négatif (bad smell) ou au contraire positif (good smell)

On peut les détecter* automatiquement avec des outils d’**analyse statique de code** (a.k.a linters)

*et éventuellement les corriger

Catalogue d'odeurs énergétique Android

Consultable sur http://green.pauware.com/android-energy-smells/

13 au départ; 26 aujourd'hui → **Il est donc possible d'en trouver d'autres !**

Introduction d'une classification des odeurs

Optimized API (2)

Leakage (3)

Sobriety (4)

Batch (3)

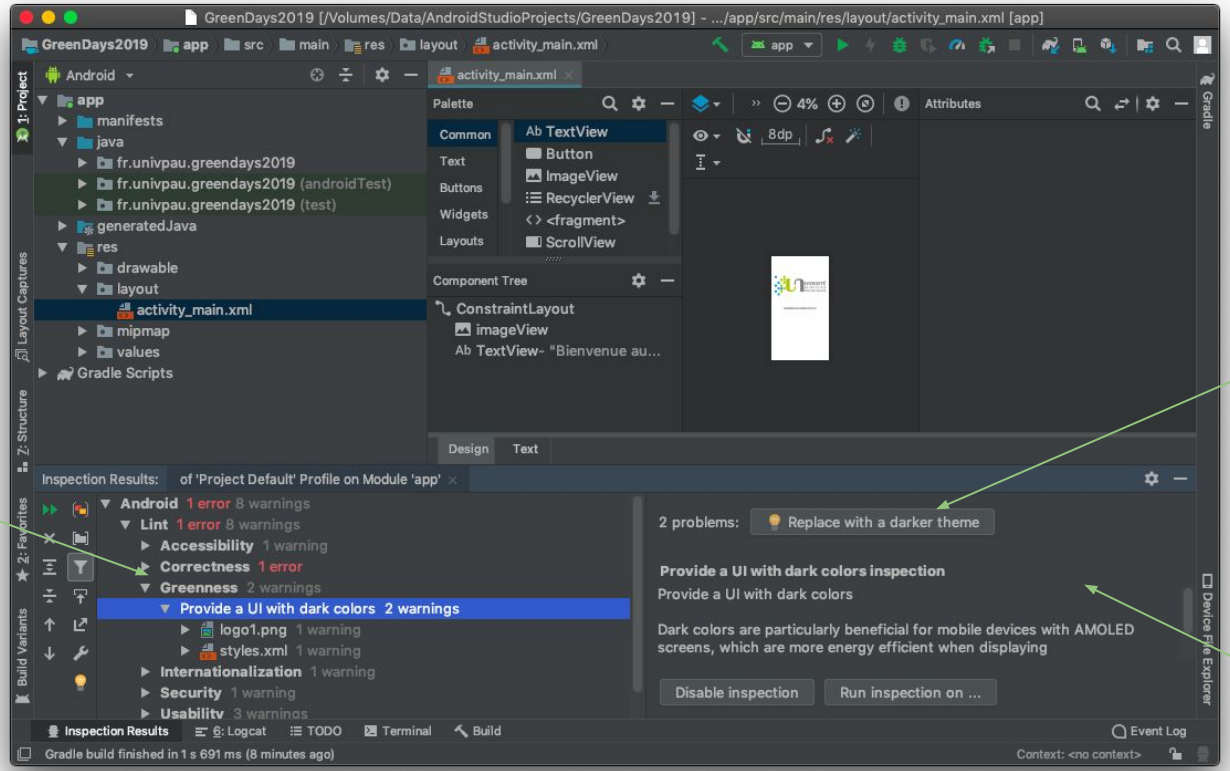
Bottleneck (4)

Power (3)

Idleness (6)

Postprocessing (1)

Preuve de concept (extension Android Lint)



Nouvelle catégorie d'inspections dédiée à l'efficience énergétique

Correctif rapide

Rapport d'inspection

Actions déjà réalisées

O. Le Goaer. *Linter Vert pour l'éco-conception logicielle*, Green Days 2019

O. Le Goaer. *Enforcing Green Code With Android Lint*, A-Mobile@ASE 2020

O. Le Goaer & F. Barbier, *Projet CleanApps (Creedengo)* en 2019

O. Le Goaer, A. Nourredine, R. Rouvoy, F. Barbier. *Vers des Logiciels Éco-responsables*, Nouveau Groupe de Travail au CNRS GDR-GPL 2020

O. Le Goaer, A. Nourredine, *Identifier des patterns logiciels de consommation énergétique mobile et serveur en utilisant les techniques de data mining* Sujet de stage M2 Recherche 2020

L’industrialisation

Creedengo™

Breaking News En janvier 2020, Microsoft a lancé *Application Inspector* pour analyser statiquement le code source et détecter les menaces

Tous les voyants sont au vert pour que Creedengo lance un outil d'analyse statique de code pour détecter les apps mobiles voraces en énergie

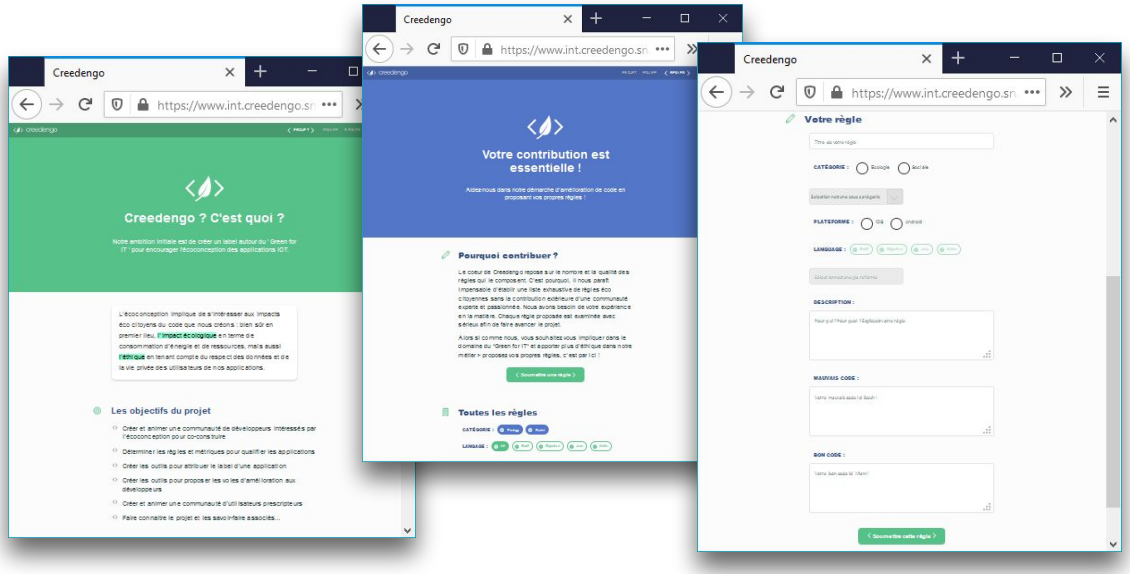
Partenariat **LIUPPA** (Génie Logiciel) + **Snapp'** (dev mobile / program analysis*)

*suite au rachat de Neomades en 2017

Jalon #1 : collecte des code smells

Plateforme Web **collaborative** : https://www.creedengo.com/

Base de connaissance ouverte (mais modérée) à n'importe quel dev mobile expérimenté



Jalon #2 : suite d’outils

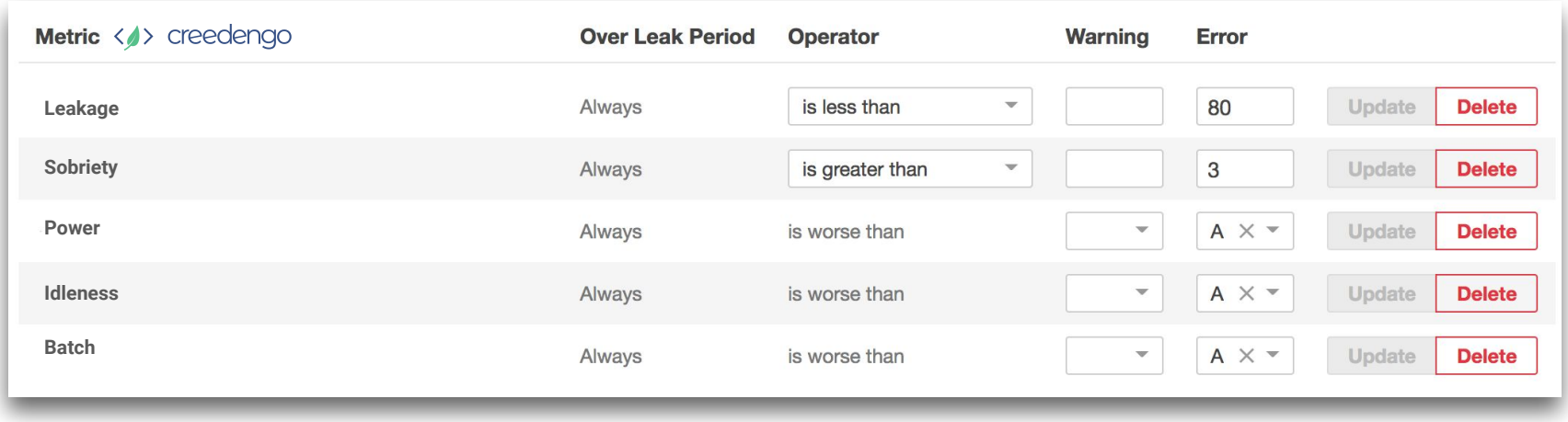
	Inspection-time	Scope	Reporting	Support
IDE-based	• on-the-fly • manual	developer-level	• code highlighting • HTML files	IDE-dependent
SaaS	• commit • pull request	project-level (collaborative)	• code highlighting • dashboard	cross-platform

Modèle économique : l’inspection est **gratuite** mais le reporting et les correctifs sont **monétisés** ?

Jalon #3 : engager les équipes de dev.

Pas facile d'imposer un outil qui "importune" les équipes en faisant remonter des alertes, sur un sujet pas **bankable** (pour l'instant !)

Laisser chaque chef de projet définir ses objectifs de qualité : *Green Quality Gates*



Conclusion

Élargissement au logiciel éco-responsable

Eco-responsabilité

=

eco-conception

+

inclusion

+

vie privée

+

...

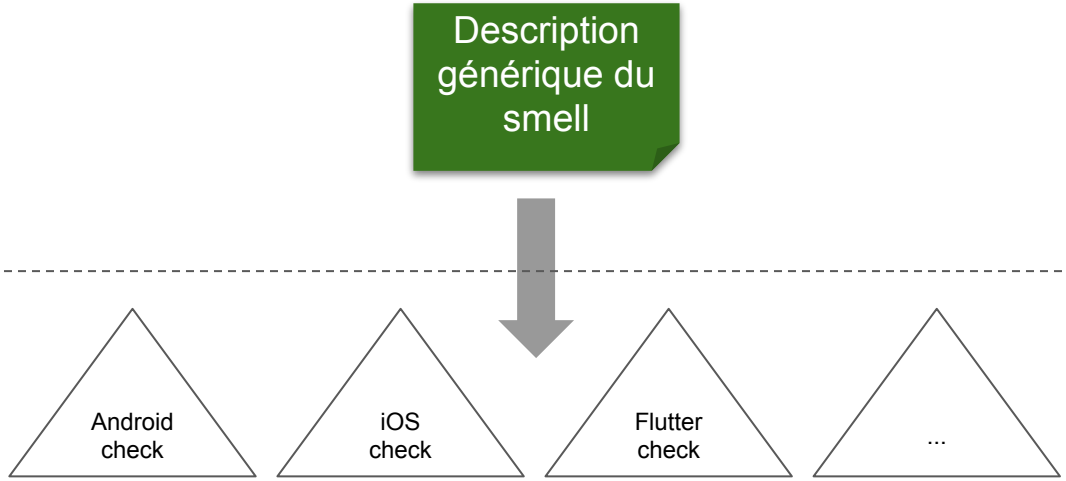
Génération automatique des détecteurs de smells

Constat : Certains smells sont communs à toutes les plateformes et technos

Question : Peut-on les décrire 1 seule fois, et générer les détecteurs pour *n* technologies (dont celles à venir) ?

platform-independant

platform-specific



Questions ?