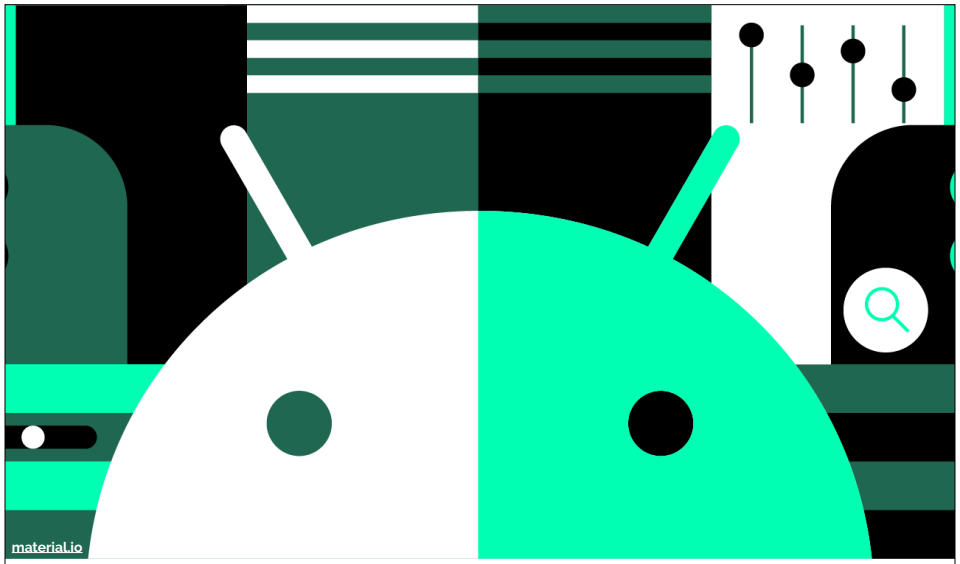


Conception de logiciels Adaptés
Introduction à Android

Sébastien Mosser
INF600G - E20 - Séquence 2 - Partie 4

UQÀM | Département d'informatique

Crédit Images: Pixabay & Pexels

ace

1 Rétrospective L1

2 Architecture App. Mobile

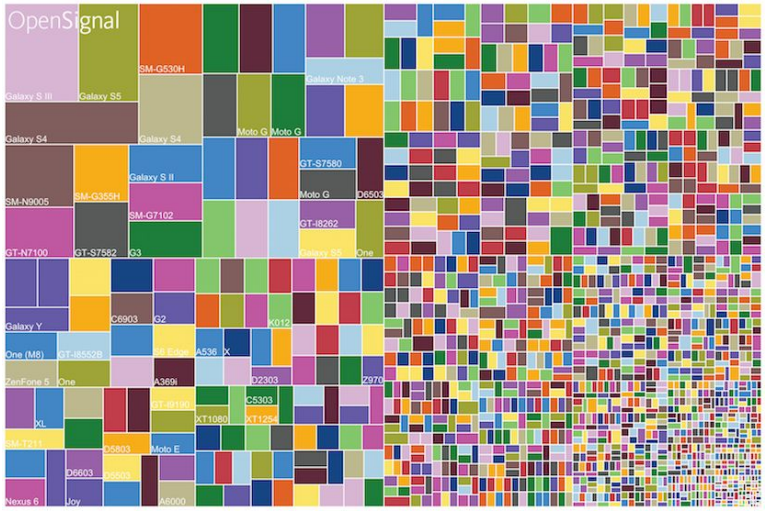
3 Exposition de ressources (REST)

4 Intro. à Android

5 Travail à faire pour L2

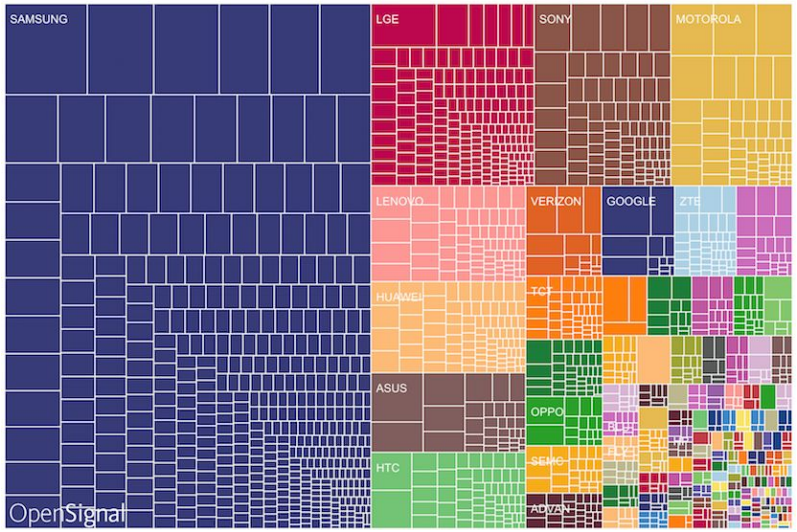
Ceci n'est pas un cours de développement mobile en Android

Un marché extrêmement segmenté



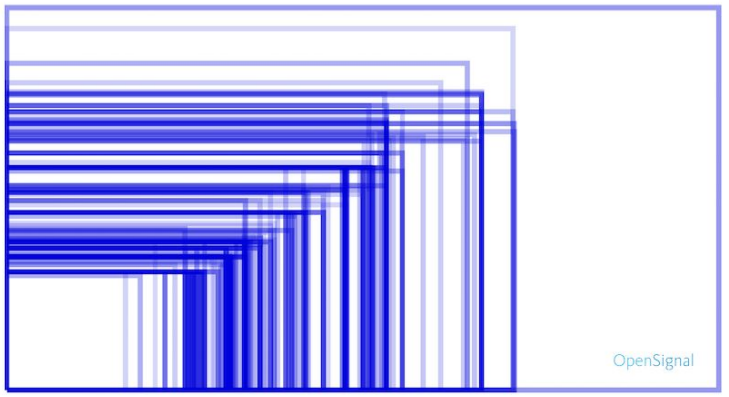
https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf

Répartition par fabricants



https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf

Fragmentation des tailles d'écrans

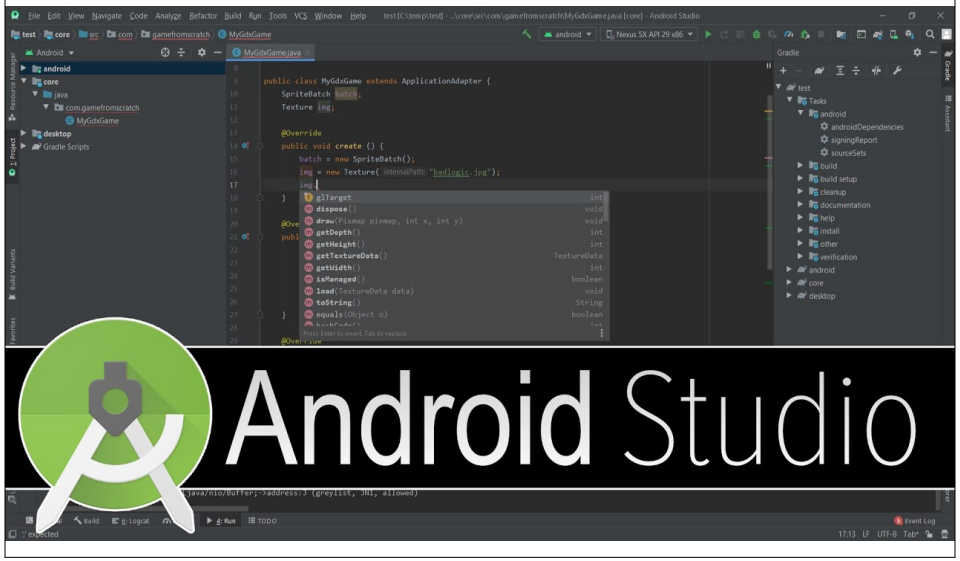


https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf

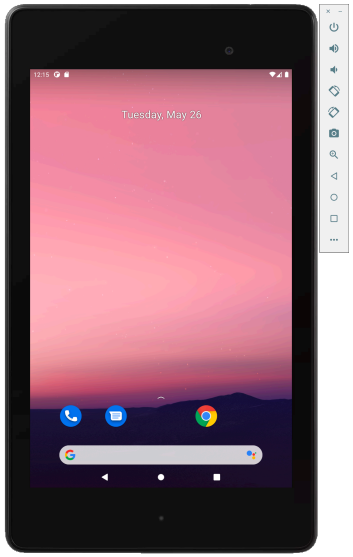
Android versus iOS

- **Philosophie Apple**
  - Maîtrise du materiel, fermeture du système d'exploitation
  - Validation préalable des applications dans l'AppStore
- **Philosophie Android**
  - Matériel ouvert, système d'exploitation Open-source
  - Validation a posteriori si une application est signalée
- Il est possible de développer du multiplateforme (Flutter, Cordova)

AndroidStudio pour le développement



Utilisation d'émulateurs

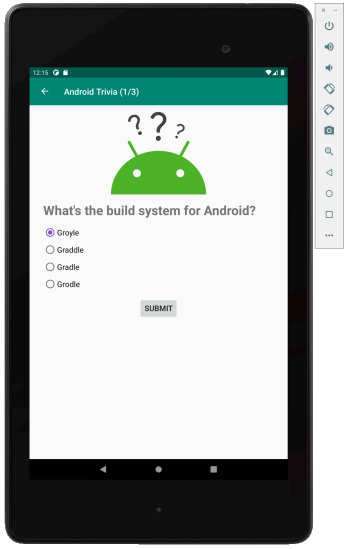


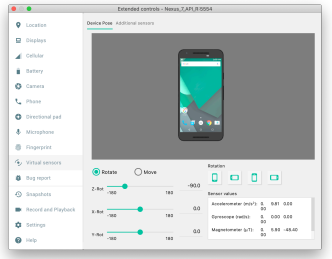
L'émulateur fonctionne comme un "vrai" périphérique.

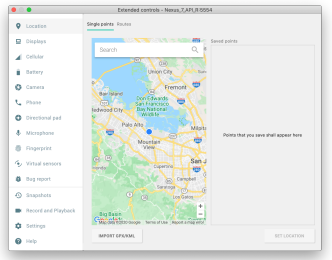
On va téléverser les application développées dans l'émulateur pour les besoins des tests locaux

On peut aussi envoyer l'application sur un via périphérique en USB

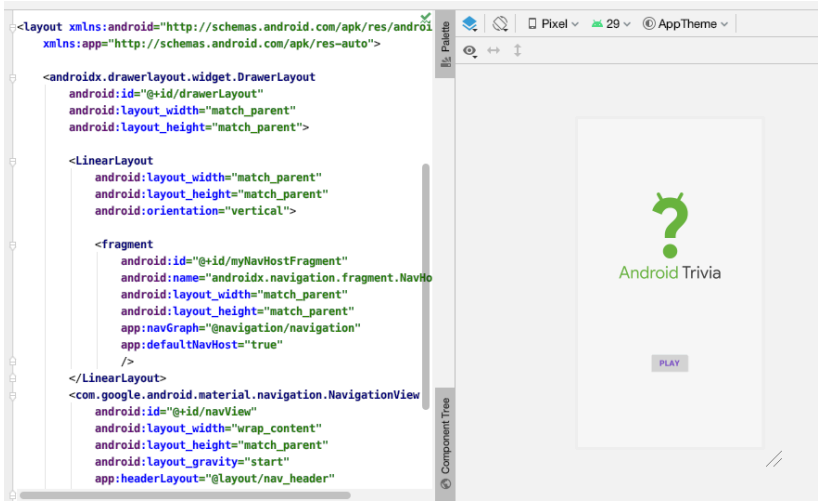
Exemple d'utilisation de l'émulateur







Layout : description XML de l'interface



app/res/layout/activity_main.xml

Activity: Point d'entrée de l'application

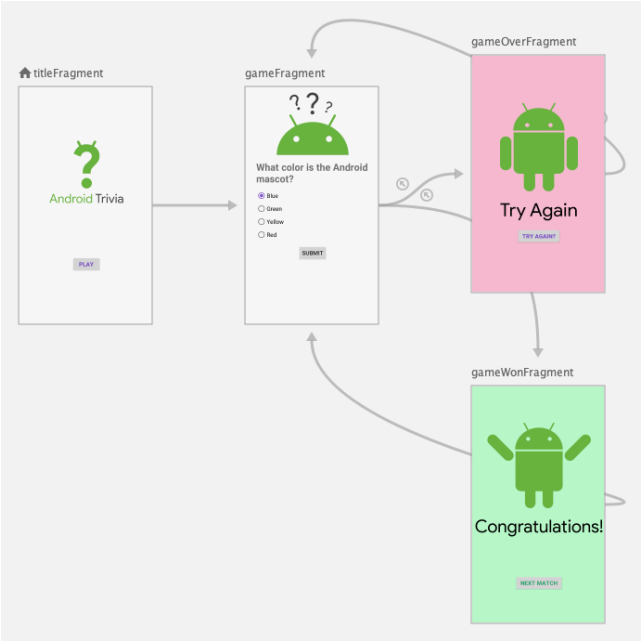
```
class MainActivity : AppCompatActivity() {
    private lateinit var drawerLayout: DrawerLayout

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        @Suppress("UNUSED_VARIABLE")
        val binding = DataBindingUtil.setContentView<ActivityMainBinding>(this, R.layout.activity_main)

        drawerLayout = binding.drawerLayout

        val navController = this.findNavController(R.id.myNavHostFragment)

        NavigationUI.setupActionBarWithNavController(this,navController, drawerLayout)

        NavigationUI.setupWithNavController(binding.navView, navController)

    }

    override fun onSupportNavigateUp(): Boolean {
        val navController = this.findNavController(R.id.myNavHostFragment)
        return NavigationUI.navigateUp(navController, drawerLayout)
    }

}
```

Lien entre activity & layout

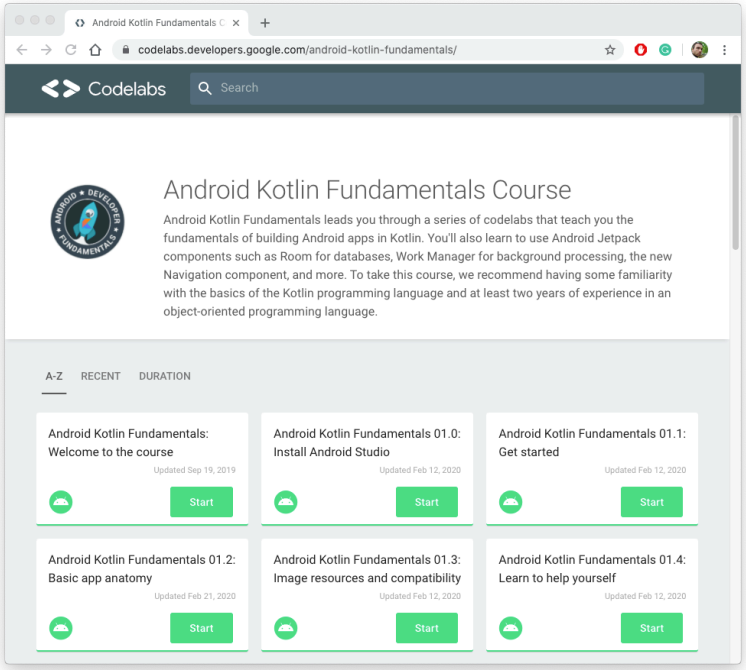
app/java/com.example.android.navigation/MainActivity.kt

Principes de Fragments

- Un fragment est un “morceau” d’interface réutilisable
  - Il est hébergé dans un hôte (une activité)
- On le définit par :
  - Son code (~ comme une activité)
  - Son interface (~ comme une activité)
- On définit la navigation entre fragments au sein de l’hôte



app/res/navigation/navigation.xml



Pendant ce temps là, au labo ...

WILEY Software: Evolution and Process

SPECIAL ISSUE PAPER

A delta-oriented approach to support the safe reuse of black-box code rewriters

Benjamin Benni[1] | Sébastien Mosser[2] | Naouel Moha[2] | Michel Riveill[1]

Abstract

Large-scale corrective and perfective maintenance is often automated thanks to rewriting rules using tools such as `Pythonizer`, `2to3`, or `CoccinelleJ`. Such tools consider these rules as black-boxes and compose multiple rules by chaining them: giving the output of a given rewriting rule as input to the next one. It is up to the developer to identify the right order (if it exists) among all the different rules to yield the right program. In this paper, we define a formal model compatible with the black-box assumption that reifies the modifications (Δs) made by each rule. Leveraging these Δs, we propose a way to safely compose multiple rules when applied to the same program by (a) ensuring the isolated application of the different rules and (b) identifying unexpected behaviors that were silently ignored before. We assess this approach on two large-scale case studies: (a) identifying conflicts in the Linux source-code automated maintenance and (b) fixing energy antipatterns existing in Android applications available on GitHub.

KEYWORDS

code rewriting, conflict detection, rule composition, software reuse

1 | INTRODUCTION

https://ace-design.github.io/

Réparation des anti-patrons

- Pourquoi écrire des programmes ?
  - Quand on peut écrire des programmes qui les écrivent à notre place ?

```java
public class C {

  private String data;

  public String setData(String s) {
    this.data = s;
  }

  public void doSomething() {
    // ...
    setData(newValue)    /* <<<< */
    // ...
  }
}
```

(A) Example of a Java class (`C.java`, p_j)

```java
public class C {

  private String data;

  public String setData(String s) {
    this.data = s;
  }

  public void doSomething() {
    // ...
    this.data = newValue    /* <<<< */
    // ...
  }
}
```

(B) $p_{igs} = R_{igs}(p_j)$

Travail de recherche : Co-localisation

- Comment réparer si deux anti-patrons sont localisé au même endroit dans l'application Android ?

- Le problème se généralise à tout système de réécriture

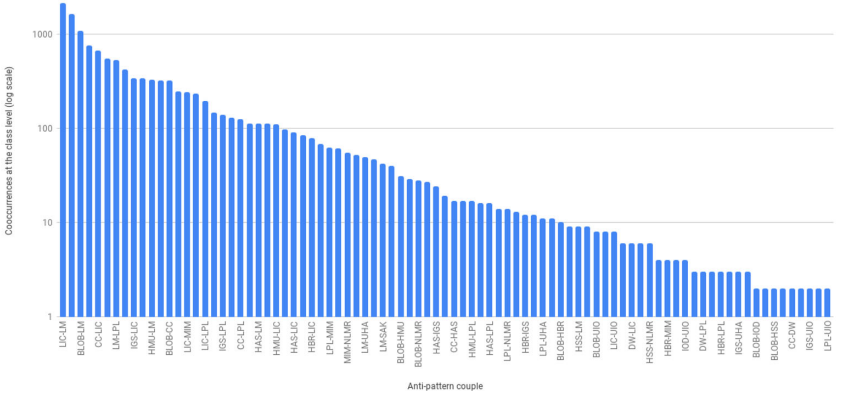
$$\oplus : \mathrm{AST} \times \mathrm{A}^*_< \to \mathrm{AST}$$

$$(p, S) \mapsto \begin{cases} S = \emptyset & \Rightarrow p \\ S = \alpha \vert S' & \Rightarrow \mathrm{exec}(\alpha, p) \oplus S', \quad \text{exec being language specific.} \end{cases}$$

$$\ominus : \mathrm{AST} \times \mathrm{AST} \to \mathrm{A}^*_<$$

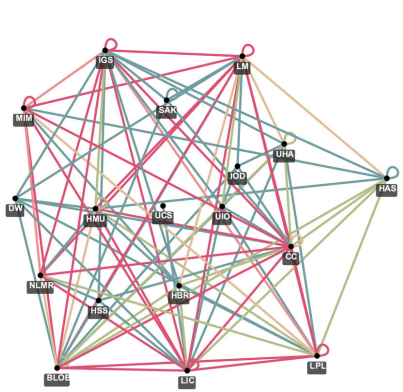
$$(p', p) \mapsto \Delta, \quad \text{where} \quad p' = p \oplus \Delta.$$

Anti-patrons co-localisés

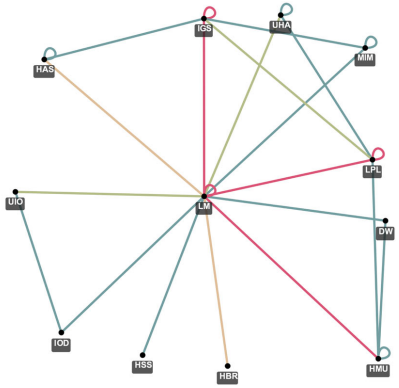


(A) Overlapping anti-patterns detected at the class level

Paires d'anti-patrons



(A) Class level



(B) Method level

Rejoignez l'équipe !

Faculty

Sébastien Mosser
Professor, UQAM

Researchers

Benjamin Benni
Postdoc, Concordia University

Cats

Jinx
European cat (COVID-19 funding)

Ravioli
European cat (COVID-19 funding)

Graduate Students

Sébastien Bonnieux
PhD Student, UCA

Jean-Philippe Caissy
MSc Student, UQAM

Jérémy Fornarino
MSc Student, ESIEE

Pierre Froidevaux
MSc Student, Exia.CESI / UQAM

Rayan Lakhdar
MSc Student, Exia.CESI / UQAM

Lou-Théo Laurent
MSc Student, Exia.CESI / UQAM

Sami Lazreg
PhD Student, UCA

Maxime Mulder
MSc Student, Exia.CESI / UQAM

Juliano Augusto Pereira
MSc.A. Student, UQAM

Florian Vouters
MSc Student, Exia.CESI / UQAM

Avril de Goër de Herve
École Normale Supérieure (ENS) de Lyon

Interns

Alison Lecuyer *B.Sc (IUT), Université Côte d'Azur*
Olivier Levasseur *B.Sc, UQAM (NSERC grant)*

INF6200, Stages, Projets de maîtrise … Venez discuter !