# CR CAO projet Androïd

## Introduction

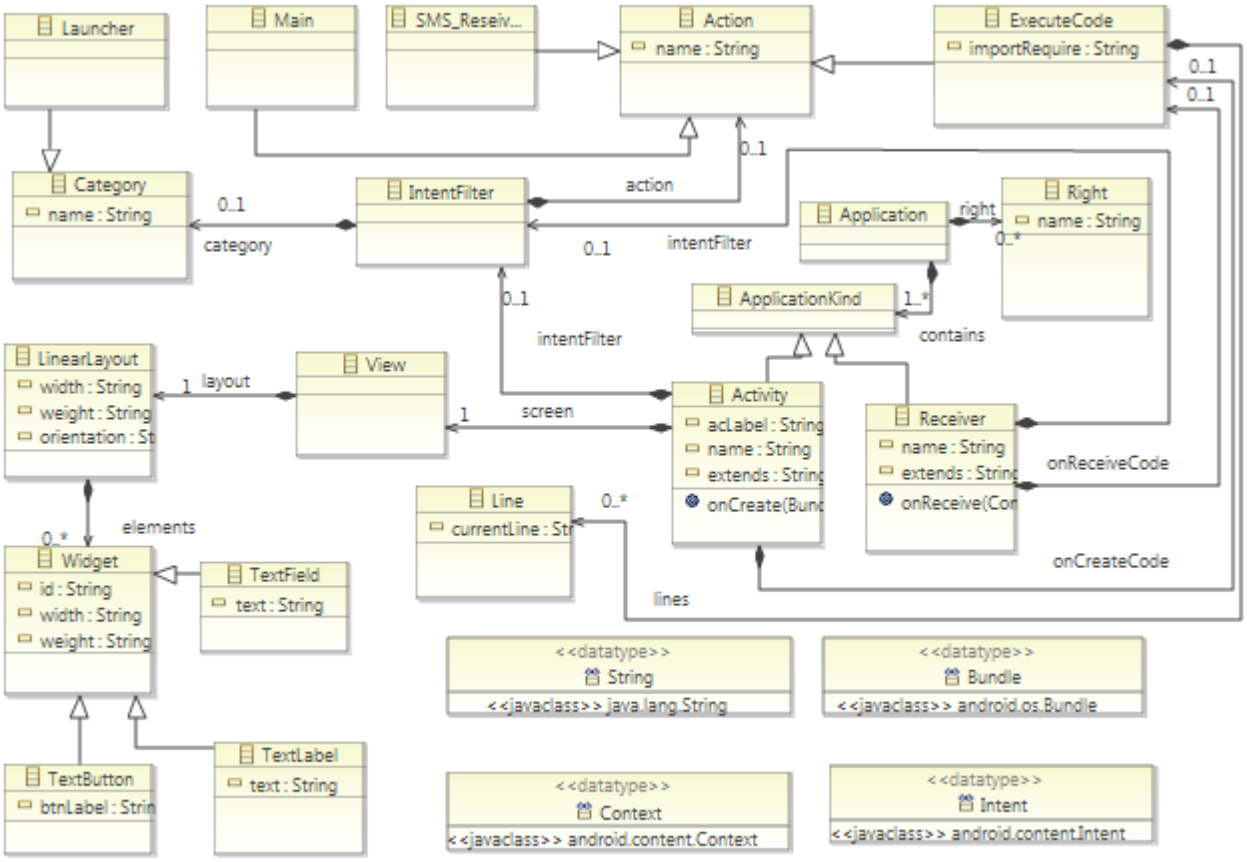
L'ingénierie dirigée par les modèles propose d'automatiser le processus de développement de logiciels. les intérêts sont multiples. Cela réduit le temps de développement d'une application tout en offrant une meilleur maintenabilité ainsi que flexibilité(portage d'un langage vers un autre plus commode).



## 1-Objectif:

Modélisé trois applications "Androïd" et les générer à partir d'un langage de modélisation.
les trois applications à modéliser sont:
-**SayHello**, simple application proposant un bouton à l'utilisateur qui une fois actionné affiche HelloWorld à l'écran.
-**SMSPrinter**, afficher sur la sortie standard le texte ainsi que l'émetteur quand un SMS arrive
-Un application de géolocalisation qui a proximité d'un lieu connu affiche les informations concernant ce lieu (ce cas ne sera pas traité).

# 2-Definition du modèle Android.ecore:




Pour les écran d'une application nous avons définit la classe "LinearLayout" qui contient une collection de "Widget" pouvant représenter des "boutons", des "labels" ainsi que des "textfields".

Notre meta-model offre la possibilité de réaliser deux types d'applications, les "Activity" ainsi que les "Receiver" correspondant aux deux cas d'études (**SayHello** et **SMSPrinter**).

Les compositions "intentFilter" dans "Activity" et "Receiver" sont utilisées pour générer le manifeste de l'application.

# 3-Définition des générateurs avec le langage kermeta

Durant l'analyse il nous est apparu possible d'automatiser; le développement des classes java, des différents layout ainsi que du manifest d'une application "Androïd".
Ainsi avec kermeta nous avons définit 3 générateurs:
-JavaGenerator générateur de code java.
-LayoutGenerator générateur de layout.
-ManifestGenerator générateur de manifest d'application Androïd.
Les trois générateurs sont situés dans le répertoire src/kermeta à la racine du projet.

AndroidAppGenerator (répertoire src) appelle successivement chacun de ces générateurs dans la fonction main pour générer l'ensemble de la partie automatisée d'un développement d'application "Androïd".

# 4-Définition d'instance de modèle

La dernière étape consiste à créer des instances de notre méta-modèle pour valider le processus d'automatisation de développement d'application Androïd. Cette étape est réalisée avec l'éditeur réflexif EMF.

Nous avons définit deux modèles d'applications pour Androïd (disponible dans le répertoire model du projet).
-HelloWorld.xmi modélise l'application sayHello.
-SMSPrinter.xmi modélise l'application SMSPrinter.

# 5-Génération de la partie automatisée du développement:

Le résultat de la génération de l'application "SayHello" est présent dans le répertoire "HelloWorld" situé a la racine du projet. pour l'obtenir nous exécutons "AndroidAppGenerator" avec les paramètres suivants:

-Le chemin de l'instance du modèle de l'application
../model/HelloWorld.xmi

-Le nom du package ou le code java généré est enregistré
cao_project/android/helloworld

-Le path ou les générateurs enregistrent leurs résultats
C:/Users/nicolas/workspace/CAO_Project/HelloWorld

Le résultat de la génération de l'application SMSPrinter est présent dans le répertoire SMSPrinter situé a la racine du projet. pour l'obtenir nous exécutons AndroidAppGenerator avec les paramètres suivants:

-Le chemin de l'instance du modèle de l'application
../model/SMSPrinter.xmi

-Le nom du package ou le code java généré est enregistré
cao_project/android/smsprinter

-Le path ou les générateurs enregistrent leurs résultats
C:/Users/nicolas/workspace/CAO_Project/SMSPrinter

# Conclusion:

Grâce à Kermeta nous sommes en mesure d'effectuer des développements dirigés par les modèles.
Cependant il est possible de pousser le concept plus loin. Nos générateurs pourrait utiliser le mécanisme de template offert par ket (mécanisme utilisé par le frame-work Struts par Velocity pour générer des pages html avec des éléments java).