Description de Graffiti

# Intro

Graffiti est basé sur le framework EMF(Eclipse Modeling Framework) avec un métamodèle au format Ecore représentant les concepts et les relations du langage. Il existe un éditeur graphique réalisé à l’aide du composant open source Sirius permettant d'éditer un diagramme de comportement de code.

# Méta-modèle

Graffiti permet de décrit le comportement d’un code via la description de système (System). Un système est constitué de trois entités : des acteurs (Actor), des événements (Event) et des connections entre les acteurs (Connection). Un acteur est constituer de ports (Port) permettant de crée des liens entre acteur et un comportement (behavior). Le comportement d’un acteur est la représentation graphique du comportement du code de tout ou partie d’un système. Il se compose d’un état initial (InitialState), d’un état final (FinalStatement) et de déclarations (Statement). La déclaration est l’élément principal du comportement d’un code, elle peut être une simple séquence (Sequence), une boucle (Loop), une ouverture de branche (BrancheIn), une fermeture de branche (BrancheOut) ou une communication (Communication). Quel que soit son type une déclaration est accompagnée d’une configuration (Configuration) qui contient les caractéristiques physiques de la déclaration (consommation énergétique, dégagement de chaleur, charge réseaux, …).

Finalement Graffiti est un outil permettant la description graphique du comportement d’un code ce via les notions de boucle, de branche et de communication. De plus Graffiti prend en compte les caractéristiques physiques relatives au comportement d’un code (temps d’exécution, consommation énergétique, dégagement de chaleur, charge réseaux, …).

FIGURE 1

# Modèles

Graffiti permet de généré des modèles de système construit en plusieurs couches : une couche système et une succession de couches de comportement. La couche système permet de définir les évènements, les acteurs et les connections entre les acteurs.

FIGURE 2

La représentation graphique des acteurs appartement au système est un carré blanc au bord noir, les acteurs du système modèlisé dans la figure 2 sont donc CPUMachine, GPUMachine et PlatformMachine. Les ports des acteurs (port d’entré ou de sortie) sont représenté par des éclipses de couleur (rouge pour sortie et bleu pour sortie). De plus les évènements sont représentés sur la couche système vie des rectangles de couleur jaune.

La couche système regroupe des informations comparable avec un model PimCa, on peut alors faire une entre la notion d’acteur Graffiti et cette de machinerie PimCa. Cette analogie servira de base à un système de synchronisation entre PimCa et Graffiti **(voir partie XXX)**.

FIGURE 3

Les couches comportemental constituent l’élément principal d’un model Graffiti, on y retrouve la schématisation du comportement du code et la caractérisation physique de ce code (temps d’exécution, dégagement énergétique, …)

Par exemple un algorithme de traitement d’image peut être modélisé via MorphoseMachine avec une boucle de courte durée ce répétant un grand nombre de fois. De plus des indications comportementales peuvent être données à une séquence (un morceau de code). Ces indications permettent par exemple de modéliser la montée en température d’un FPGA effectuant un calcul complexe ou encore la consommation d’un processeur en pleine charge.

Dans notre cas d’utilisation nous instancions de métamodèle de la figure 1 afin de créer le modèle de comportement du code d’un système de détections de forme. Ce système est composé de deux acteurs communiquant ensemble, un FPGA et un processeur (voir figure 2). Chaque acteur ayant un comportement particulier représenté dans les figures 3 à 5.

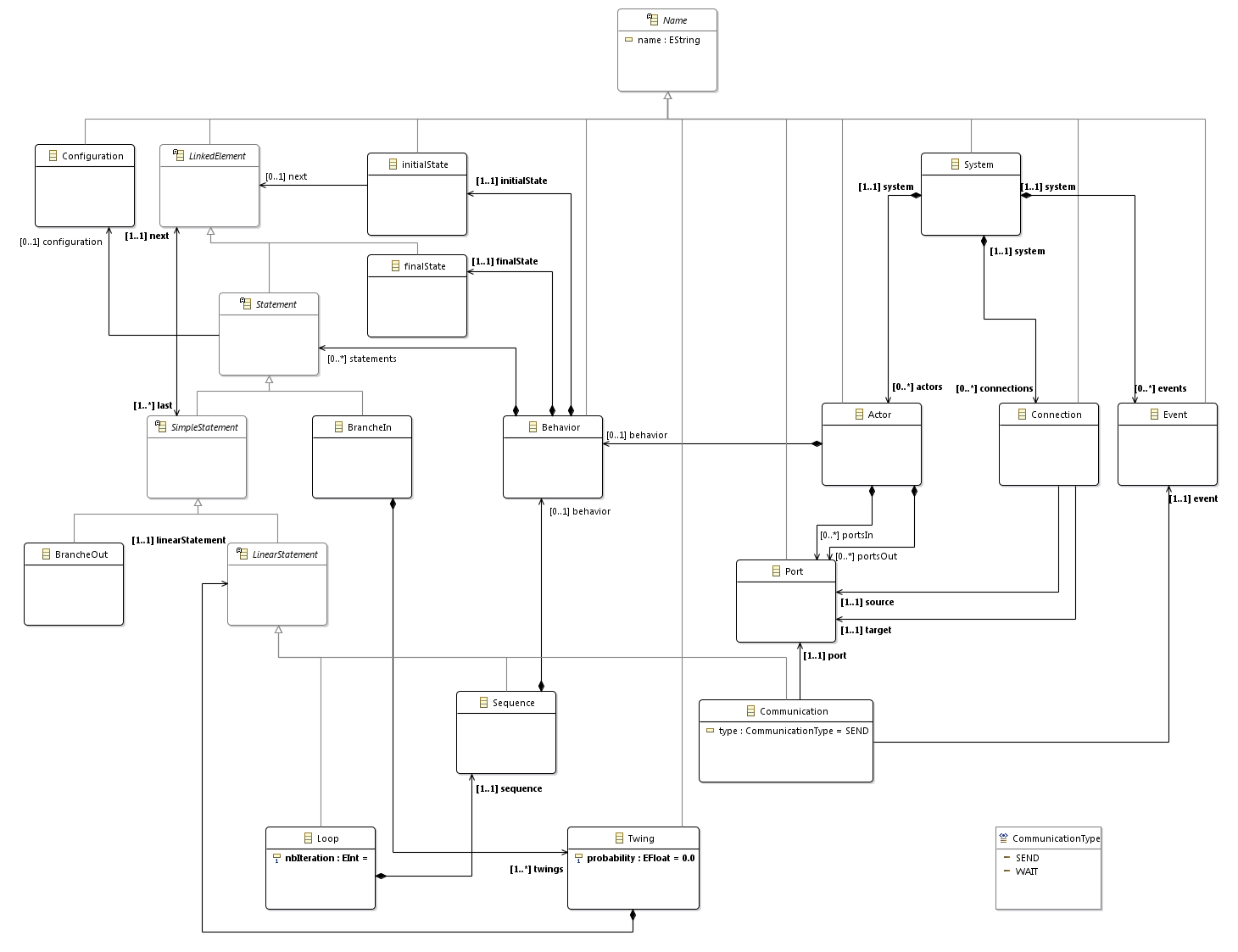


Figure 1: Méta-modèle de Graffiti

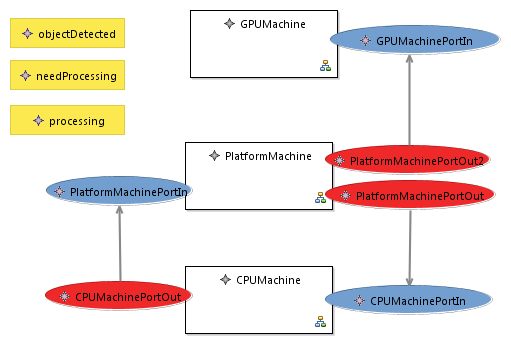


Figure 2: Couche système du modèle d'un système de détection de forme simplifié sous Graffiti

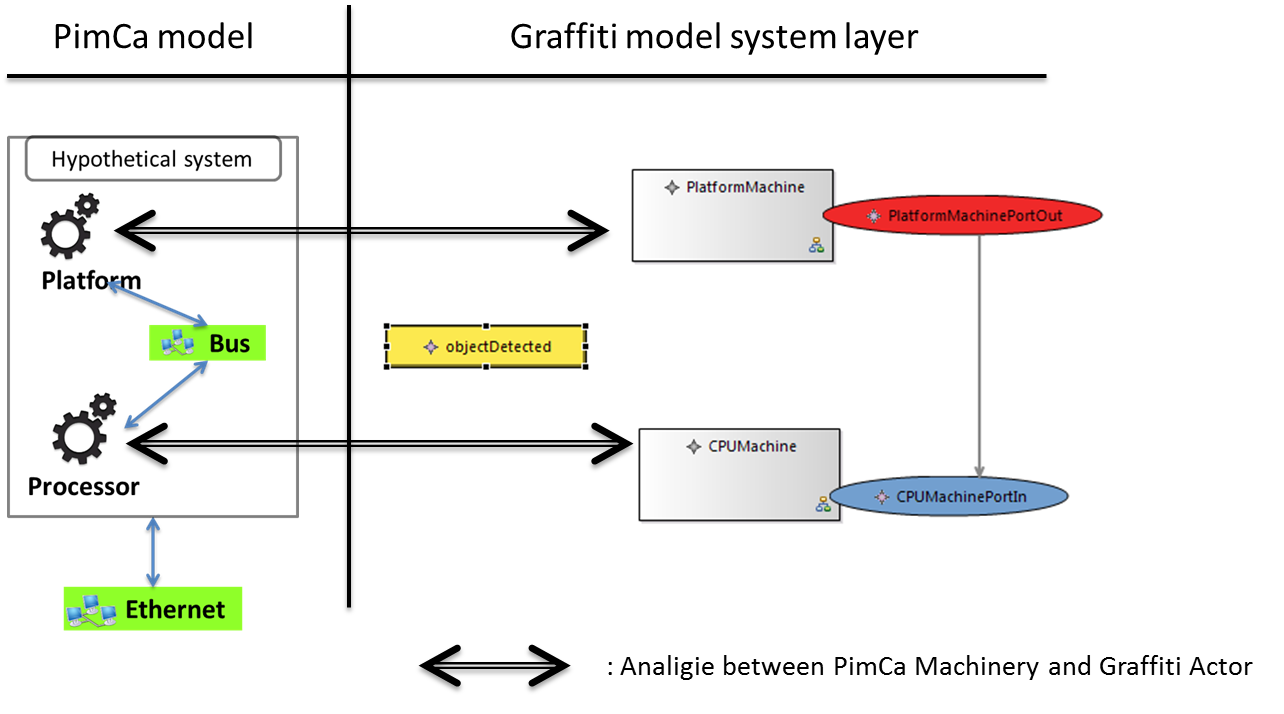


Figure 3: Analogie entre un modèle PimCa et la couche système d’un modèle Graffiti

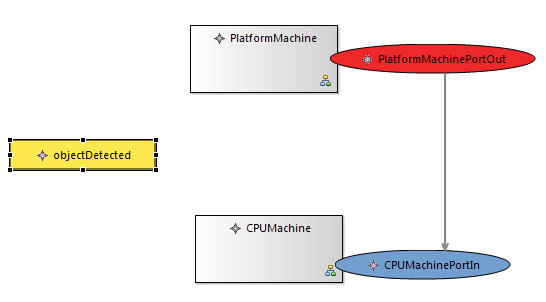


Figure 4: Modélisation d'un système de détection de forme simplifié sous MorphoseMachine

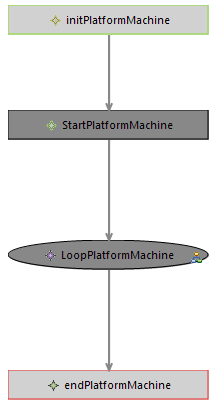
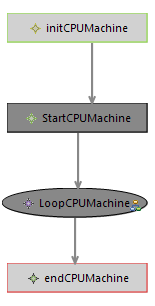


Figure 3: Comportement du code global d'un CPU et d'une plateforme (FPGA) utilisé pour la détection de forme

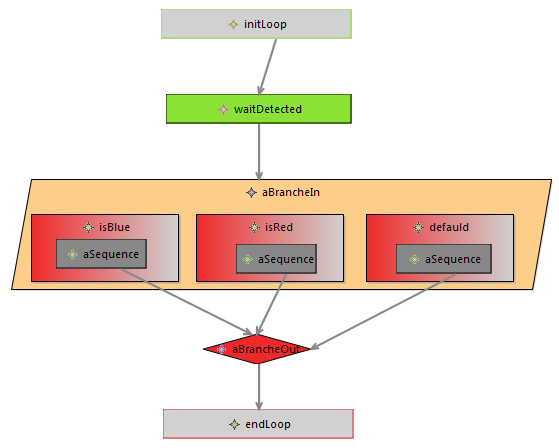


Figure 4: Comportement de la boucle principal du code implémenté sur le CPU utilisé lors de la détection de forme

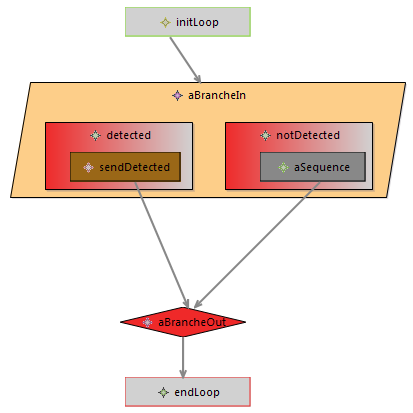


Figure 5: Comportement de la boucle principal du code implémenté sur le FPGA utilisé lors de la détection de forme