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 modélisé 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, …). Elle peut être linéaire ou décisionnel, contenir des boucles, des communications ou d’autres séquences.

FIGURE 4

La **figure 4** présente le comportement de l’acteur PlatformMachine, il est composé d’un état initial d’un état final et de deux déclarations : une séquence et une boucle. La séquence représente le comportement de la plateforme lors de l’initialisation, ce comportement n’est pas détaillé dans cet exemple car il ne fait pas partie du mode de fonctionnement standard de la plateforme. La boucle quant à elle représente le mode de fonctionnement standard de la plateforme (ici une détection de forme), son comportement a donc était détaillé (**figure 5**). Pour ce faire nous avons ajouté un comportement (Behavior) à la boucle, ce comportement est représenté sur la **figure 5**.

FIGURE 5

La **figure 5** est la représentation graphique du comportement de la boucle principal de l’acteur représentant une plateforme utilisé lors de la détection d’objets. Comme dans tout comportement représenté sous Graffiti on retrouve un état initial et un état final. On retrouve aussi ici une branche décisionnel, soit l’objet est détecté (branche de gauche) soit il ne l’est pas (branche de droite). Si l’objet est détecté l’évènement sendDetected est envoyé sur l’un des ports de sortie de l’acteur PlatformMachine, puis le processus reste bloqué jusqu’à recevoir l’évènement needProcessing sur l’un des ports d’entré de l’acteur. Finalement l’évènement processing est envoyé sur un autre port de sortie de l’acteur. Par contre si l’objet n’est pas détecté alors la branche ce ferme sans envoyer d’événement. Ce comportement ce répète en boucle un nombre de fois défini (nombre fini ou non).

# Conclusion

Graffiti est un outil de représentation graphique du comportement du code d’un système. Les modèles créés sous Graffiti sont constitués de couches superposées incluant une couche système, décrivant le système global via des acteurs et des connections entre acteur, et des couches comportementales. Ces dernières constituent le cœur de l’outil est sont défini grâce à un nombre restreint de concepts (branche, boucle, envois/réception de message et séquence). Les éléments d’une couche comportemental peuvent eux même inclure des comportements ce qui permet de représenté le comportement d’un code de manière plus ou moins fine selon les besoins. Les éléments constitutif d’un model Graffiti sont rappelés dans la **figure 6**

FIGURE 6

# NB

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.

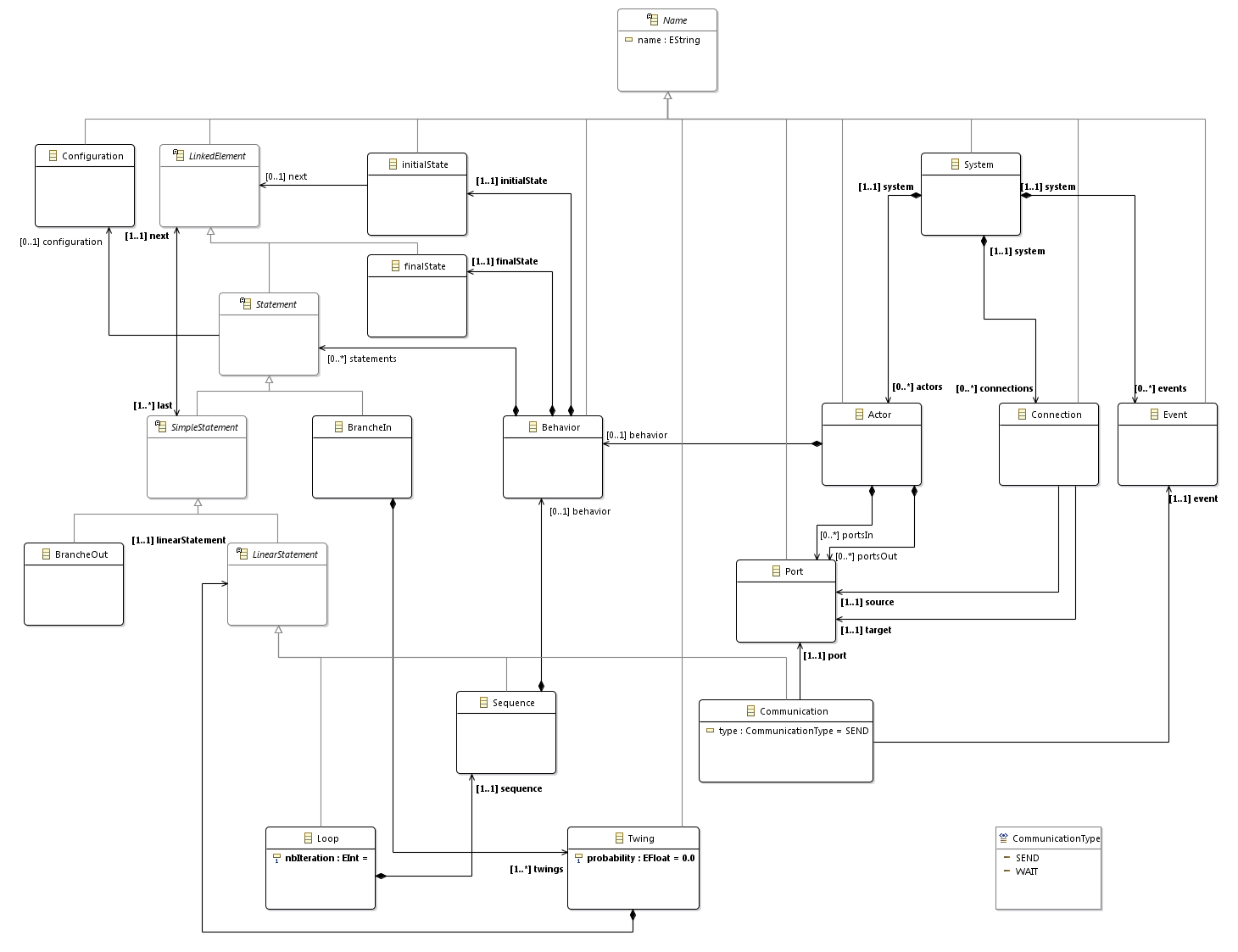


Figure 1: Méta-modèle de Graffiti

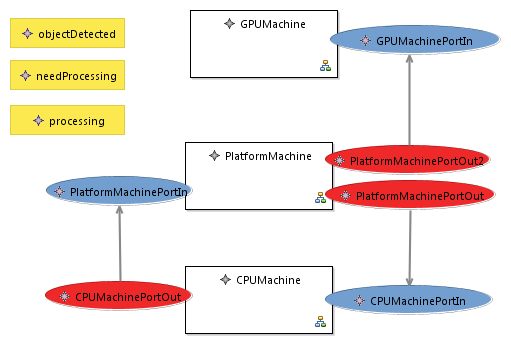


Figure 2: Couche système du modèle d'un système de détection de forme simplifié sous Graffiti

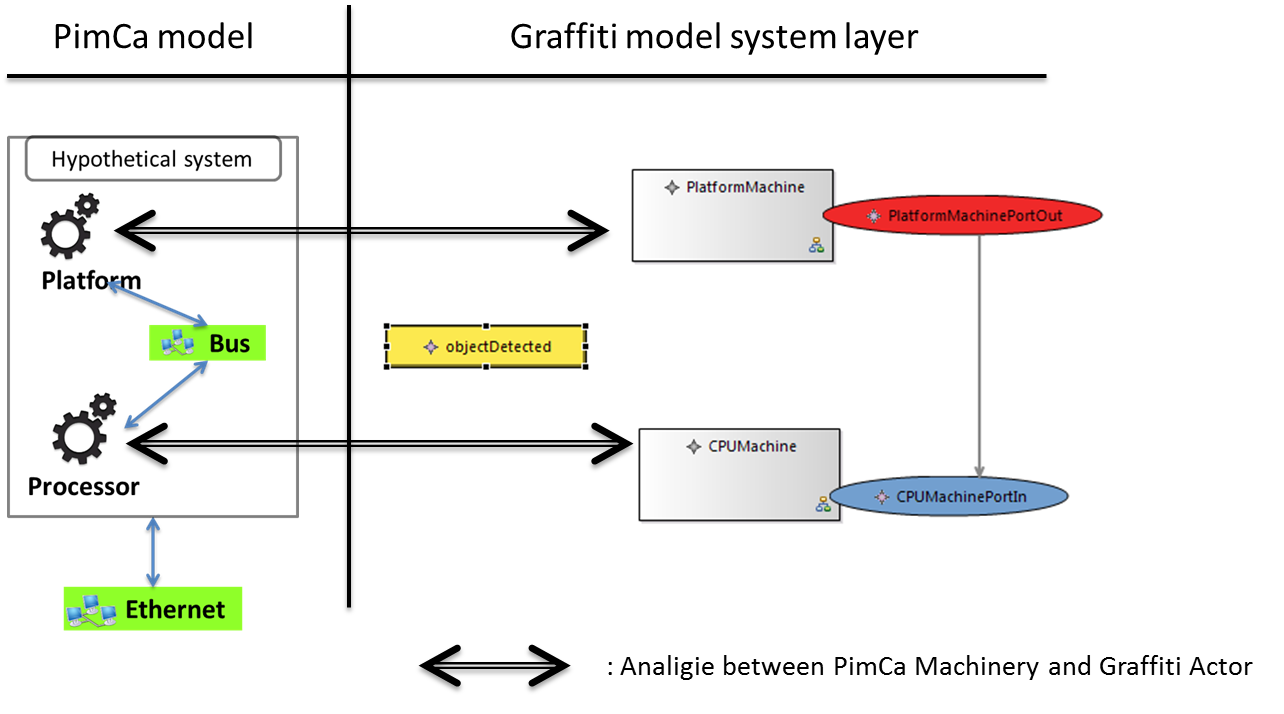


Figure 3: Analogie entre un modèle PimCa et la couche système d’un modèle Graffiti

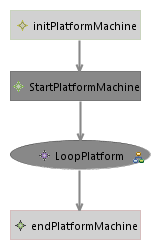


Figure 4: Couche comportemental de 1er niveau représentant le comportement de la plateforme

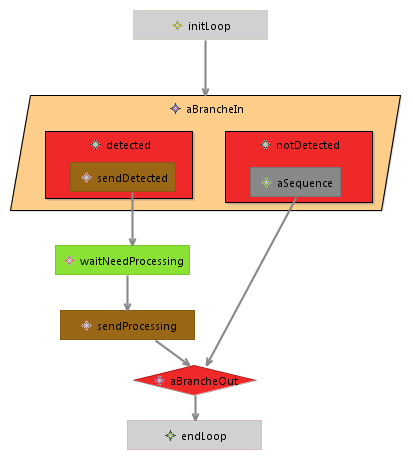


Figure 5: Couche comportemental de 2nd niveau représentant le comportement de la plateforme

|  |  |
| --- | --- |
| **Icône** | Description |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\actor.PNG** | **Acteur** : Représentation d’une partie du système réel ou logiciel interagissant avec d’autre acteur ou l’environnement |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\event.PNG** | **Evènement** : Evènement utilisé dans le système. Il peut être envoyé (send) ou attendu (wait). |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\portIn.PNG** | **Port d’entrée** : Interface d’échange entre acteur permettant la réception d’évènement |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\portOut.PNG** | **Port de sortie** : Interface d’échange entre acteur permettant l’envoi d’évènement. |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\connection_or_edge.PNG** | **Connection** : Connecteur entre un port de sortie d’un acteur et un port d’entré d’un autre acteur. |

|  |  |
| --- | --- |
| **Icône** | Description |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\init.PNG** | **Etat d’initialisation** Etat débutant tous schémas de comportement de code. Il est nécessaire à la définition d’un comportement. |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\end.PNG** | **Etat final**: Etat terminant tous schémas de comportement de code. Il est nécessaire à la définition d’un comportement. |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\sequence.PNG** | **Séquence** : Représentation d’une partie d’un comportement (appelle de fonction, initialisation d’un appareil, …). |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\loop.PNG** | **Boucle** : Itération successive d’une séquence. |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\send.PNG** | **Envois d’évènement** : Séquence dédié à l’émission d’un évènement sur un port de sortie d’un acteur. |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\wait.PNG** | **Réception d’évènement** : Séquence dédié à l’attente d’un évènement sur un port d’entré d’un acteur. Une réception d’évènement est une action bloquante. . |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\brancheIn.PNG** | **Ouverture de branche** : Conteneur permettant de crée des alternatives. |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\twig.PNG** | **Alternative** : Conteneur probabiliste marquant le début d’une alternative. |
| **C:\Users\drouotba\Documents\GitHub\idm_modsoc\documentation\docTravail\Synthesis\data\Models\Graffiti\elements\brancheOut.PNG** | **Fermeture de branche**: Nœud de fusion de plusieurs alternatives. |

Figure 6: List des éléments utilisables dans Graffiti