Passerelles UML/Signal

Plan

1. **Première passerelle UML/signal : Y A T U S (1)**
   - **Avancement depuis décembre 2002**
   - **Conférence Neptune du 22/01/2003**
   - **Reste à faire**

2. **Deuxième passerelle UML/Signal**
   - **Avancement depuis décembre 2002**
   - **Reste à faire**

3. **Remarque sur les deux passerelles**

(1) YATUS : Yet Another Translator from UML to Signal

Avancement sur YATUS depuis décembre 2002

- **Signal « Top » (unité de temps) rajouté automatiquement dans la génération de code Signal**

- **Possibilité de ne pas systématiquement déclarer synchrones les entrées/sorties d'un process associé à une méthode de classe :**
  - **tagged value {nosynch} pour une opération**

- **Tentative de traitement des états composites**

- **Système de priorité entre les transitions pour gérer la simultanéité des signaux :**
  - **tagged value {priority(i)} pour une transition**

- **Option de génération qui permet de rajouter automatiquement en sortie un signal de trace pour l'état d'une classe active :**
  - **tagged value {stateOutput} pour une classe**
  - **tagged value {stateValue(i)} pour un état**

Conférence Neptune du 22/01/2003

- **Présentation des principes de la traduction UML/Signal dans YATUS**

- **Démonstration « Ballon Vapeur » :**
  - **Modèle UML du régulateur**
  - **Génération de code Signal**
  - **Génération de code Java**
  - **Simulation**

Installation chez SITIA du 06/03/2003

Installation sur station SUN :

- **Module Objecteering de YATUS**
- **Polychrony (version téléchargeable)**
- **Java (jdk 1.4)**
- **Démonstration Ballon-vapeur**
  (Modèles UML, Interfaces utilisateur java, compléments bibliothèque Polychrony, ...)

Reste à faire sur YATUS

- **Prise en compte des remarques futures formulées par SITIA**

- **Adaptations pour Application Traitement d'image ?**
(prise en compte des tableaux, ...)

Deuxième passerelle UML/Signal

Etat d'avancement :

- **Exemple Ping-pong avec appels de services sans paramètres, avec signaux d'entrée sans attributs, sans appels périodiques d'opération**

Reste à faire :

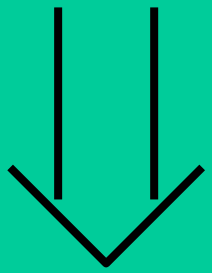
- **Traiter les paramètres d'entrée et de retour dans les appels de services**
- **Traiter les signaux UML d'entrée/sortie**
- **Traiter les appels périodiques d'opération**

(une première version pour fin mars 2003)

- **Amélioration pour les applications de référence**
- **Installation d'un module Objecteering chez SITIA et MBDA pour évaluation ?**

Remarque sur les passerelles UML/Signal

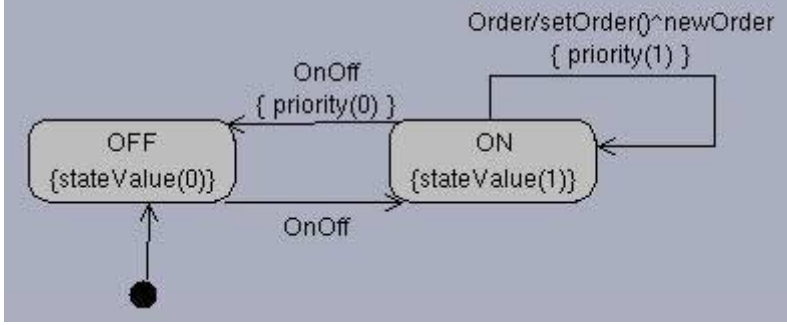
Deux passerelles UML/Signal



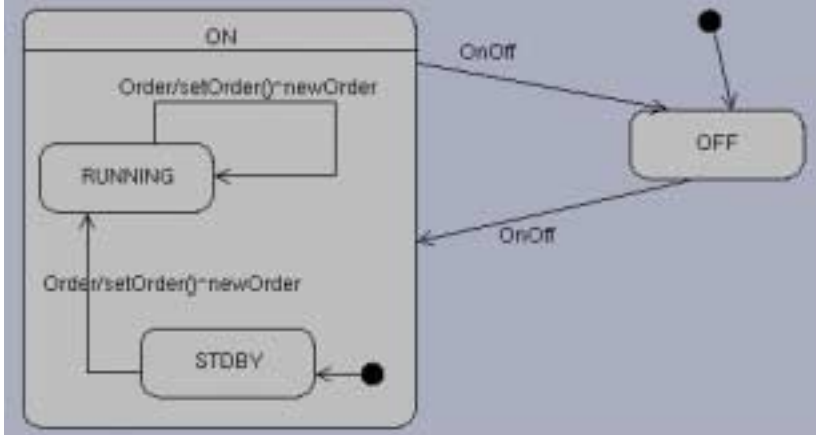
Deux modèles UML pour une même application

Annexe

Utilisation des tagged values {stateValue(i)} et {priority(i)}



Etat composite



Déclaration des entrées/sorties synchrones du process associé à une méthode

```
process ControlEquation_calculateControlValue =
{ real KP, TI, KCOR, SEUIL_QV, Te;
 }
( ? event clk;
real steamFlow;
real level;
real order;
! real output;
 )
spec(| clk ^= output ^= steamFlow ^= level ^= order |)
(| Err := order - level
| X := ((Te/TI)*Err) + (X$1 init 0.0)
| REG3E := steamFlow > SEUIL_QV
| QEA1E := ((KP/KCOR) * Err) + (X / KCOR)
| QEA3E := ((KP/KCOR) * Err) + (X / KCOR) + steamFlow
| output := if REG3E then QEA3E else QEA1E
|)
where
real Err, X;
real QEA1E, QEA3E;
boolean REG3E;
end;
```