Travaux Pratiques
Ingénierie Dirigée par les Modèles
Tâche 1:
Spécification des concepts IACA

Frédéric Fondement
frederic.fondement@uha.fr

1. Description du problème

IACA est un langage de composants, organisés en bibliothèques. Chaque composant, éventuellement paramétrable par des attributs, expose un certain nombre d'entrées/sorties, qu'elles soient destinées à recevoir/émettre des données ou des événements. Un attribut ou une entrée/sortie est toujours d'un certain type, soient bit, entier, entier long, réel, caractère, chaîne de caractères, etc. Pour définir un processus, il faut instancier des composants (préexistants). Attention: certains composants ne sont instanciables qu'une seule fois. Chaque instance de composant doit définir les valeurs de chacun de ses éventuels paramètres, valeurs qui doivent correspondre au type déclaré desdits paramètres. Des connections peuvent êtres établies entre les instances de composants par mise en relation d'un port de sortie vers un port d'entrée tels que déclarés par les composants. Une connection relie soit des entrées/sorties de type donnée, soit des entrées/sorties de type événement, mais toujours de même type (sortie bit connectée sur une entrée bit, etc.). Au final, toutes les entrées doivent être connectées sur une sortie.

La librairie standard IACA comporte (entre autre) les composants donnés au Tableau 1.

TABLEAU 1. Quelques composants IACA

Nom	Attributs	Données Entrantes	Données Sortantes	Evènements Entrants	Evènements Sortants
TOR	port (caractère) bit (entier)	E (bit)			
bit	port (caractère) bit (entier)		S (bit)		
filtre	Tc (entier)	E (bit)	S (bit)	ev_h (bit)	
clock (singleton)	FOSC (entier long) période (entier long)				ev_h (bit)
front		E (bit)			fm (bit) fd (bit)
diviseur	ratio (entier long)			ev (bit)	ev_h (bit)

TABLEAU 1. Quelques composants IACA

Nom	Attributs	Données Entrantes	Données Sortantes	Evènements Entrants	Evènements Sortants
bascule		E (bit)	Q (bit) Qb (bit)	ev_h (bit)	
cligno	port (caractère) bit (entier) periode (entier long)		Q (bit)	ev_h (bit)	

2. Travail demandé

Je vous demande de modéliser les concepts décrits ci-dessus par un métamodèle `ecore`. Vous ne parviendrez pas à décrire tout, c'est pourquoi vous aurez également à me fournir des contraintes supplémentaires données en langage naturel. Ce métamodèle servira à la génération de classes Java par l'outil EMF. Il vous est possible de modifier et/ou compléter cette génération, par exemple de manière à automatiser la conformité aux contraintes.

Je vous demande également deux modèles, instances du métamodèle `ecore` fourni: le premier est la librairie standard IACA composée (au moins) des éléments donnés ci-dessus. D'autre part, je vous demande de me fournir un modèle de processus faisant intervenir un bouton, une LED, une "clock", et un filtre: selon le modèle, le bouton devra allumer la LED via le filtre comme représenté dans la Figure 1. La créat-

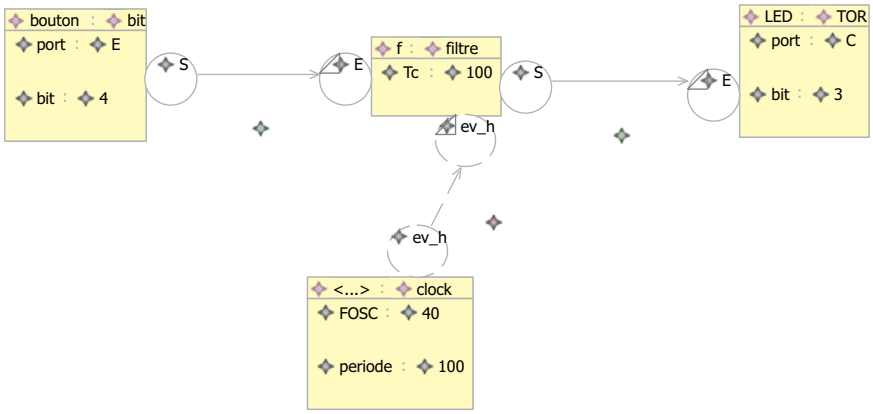

FIGURE 1. Représentation d'un modèle d'utilisation de composants IACA

ion du premier modèle devra se faire par l'éditeur réflexif généré par l'outil EMF. Le second modèle sera généré par un programme java, et s'appuiera sur le premier modèle (chargement et utilisation de la bibliothèque standard IACA sous forme de modèle par code Java).

3. Ressources et astuces

Vous aurez bien sûr besoin de vos supports de cours et ancien TPs sur IACA.

Une introduction à EMF se trouve sur `le site dédié` (`http://www.eclipse.org/emf`). Le chapitre traitant de la manipulation de modèles EMF

par Java est intitulé «Using the Generated EMF Classes». Veillez également à bien comprendre ce que sont des références croisées.

Par ailleurs, un excellent tutoriel se trouve sur `http://moogli/Fondement/2007-8/IMM/openArchitectureWare-42-reference.pdf`, des sections 1.2 à 1.7. Notez qu'il existe des éditeurs graphiques plus simples à utiliser pour décrire des (méta-)modèles `ecore`. Vous créerez un tel diagramme en invoquant la commande New.../Other.../Other/Ecore Diagram à la place de créer simplement un modèle `ecore` (figure 1.3 du document sus-cité). Un diagramme ainsi créé est lié à un fichier `ecore` donné. Le menu contextuel d'un élément vous permet d'en modifier les propriétés («show properties view»).

Les exemples pour les diagrammes d'états sont donnés par les projets `sc.mm.zip` (métamodèle), `sc.java.zip` (manipulation par Java) et `sc.model.zip` (modèle) disponibles sous `http://moogli/Fondement/2007-8/IMM/StateCharts`.

Il existe plusieurs solutions pour placer des références vers un modèle extérieur. Dans le cas de l'éditeur réflexif (par exemple ce qui est montré dans la figure 1.14 du document openArchitectureWare), le menu contextuel vous propose une item «Load resource...». Si vous manipulez votre modèle par Java, vous pouvez tout simplement placer des références dans un modèle sur des éléments d'autres modèles. Pour avoir accès aux primitives de manipulation XMI en Java, vous aurez besoin de référencer le plugin `org.eclipse.emf.ecore.xmi` dans les dépendances de `META-INF/MANIFSEST.MF`.

4. Délivrables

- Votre métamodèle sous forme ecore (fichier `XMI .ecore`), ainsi que son diagramme (fichier `.ecore_diagram`).
- Un modèle décrivant la librairie standard IACA, intégrant au minimum les éléments décrits au Tableau 1.
- Un programme Java pour créer un équivalent du modèle représenté en Figure 1.
- Une liste de modifications apportées à la génération `EMF`.
- Une liste de contraintes en langage naturel.

Le tout sera empaqueté dans un projet Eclipse au format archive `ZIP` et accompagné d'un fichier explicatif (par exemple lisez-moi.pdf).