Travaux Pratiques
Ingénierie Dirigée par les Modèles
Tâche 3:
Syntaxe Graphique

Frédéric Fondement
frederic.fondement@uha.fr

1. Description du problème

Vous vous êtes jusqu'à présent concentrés sur la description d'une syntaxe abstraite pour le langage IACA. Cette syntaxe abstraite a été formalisée par un métamodèle et des règles de cohérence. Une spécification IACA correspond par conséquent à un ou plusieurs modèles instances du métamodèle IACA, modèles qui doivent répondre à des critères de cohérence que vous avez formalisé en langage `Check`. Pour fabriquer ces modèles, vous avez utilisé un éditeur réflexif sous forme d'arbre de données, ou des programmes manipulant les données de la spécification. Si cette approche est fonctionnelle, elle n'est pas très ergonomique. Pour faire en sorte que la spécification utilise le savoir-faire du domaine, IACA a également besoin d'une syntaxe concrète. Vous verrez en théorie du langage comment supporter des syntaxes concrètes textuelles; le but de cette tâche est de supporter une syntaxe textuelle graphique.

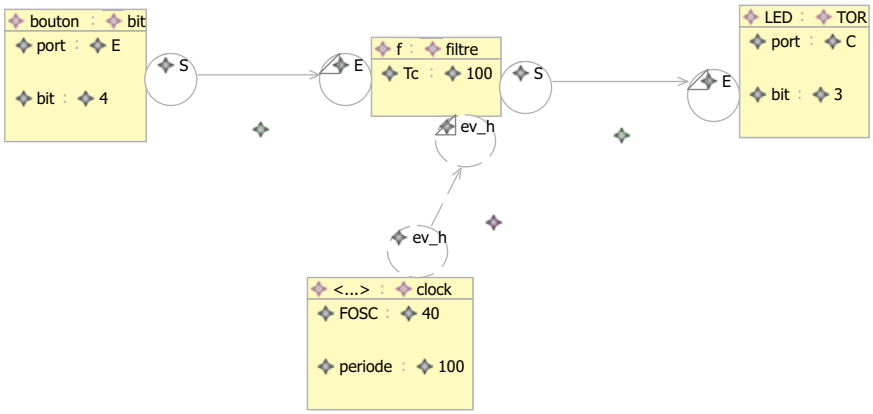
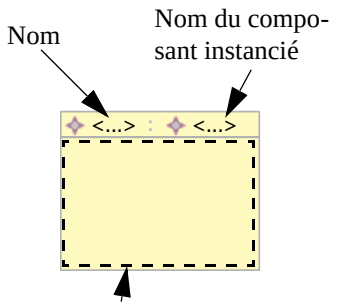
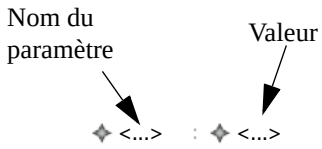
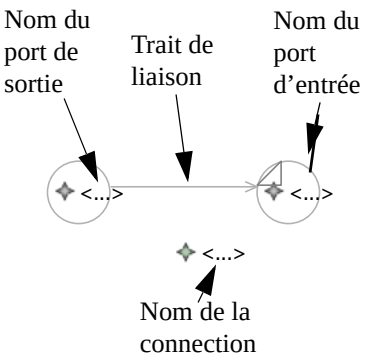


FIGURE 1. Représentation d'un modèle d'utilisation de composants IACA

Un exemple de spécification IACA représenté à l'aide d'une syntaxe graphique est donné en Figure 1. Les diverses icônes sont données dans le Tableau 1. Chaque icône est la représentation d'un concept décrit de manière informelle dans la description de la tâche 1, et formellement dans votre métamodèle. Les icônes définissent un certain nombre de points de variation, c'est à dire des éléments variables en fonction des données du modèle. L'emplacement d'une icône peut également être important.

TABLEAU 1. Syntaxe graphique informelle pour IACA

Concept	Icône	Points de variation	Emplacement
Instance de composant		Le nom de l'instance du composant. Le nom du composant instancié.	N'importe où sur le diagramme. La taille de la représentation peut varier, mais l'icône doit contenir tous ses éléments.
Valeur de paramètre		Le nom du paramètre correspondant à la valeur. La valeur proprement dite.	Dans la partie réceptacle dédiée de la représentation de l'instance de composant propriétaire.
Connection		Le nom des ports de sortie et d'entrée qui relient les instances de composants. L'éventuel nom de la connection. S'il s'agit d'une connection de données, les traits des cercles et du trait de liaison sont pleins; s'il s'agit d'une connection d'événement, ces traits sont pointillés.	Le trait de liaison peut suivre n'importe quel chemin. Les cercles de départ et d'arrivée doivent être affixés aux représentations des instances de composants connectés.

2. Travail demandé

Vous utiliserez l'outil GMF pour créer un éditeur graphique pour le langage IACA, en suivant les règles de la syntaxe abstraite décrites ci-dessus. Il est possible que vous ayez à faire évoluer votre métamodèle (ou le code généré), mais limitez ces évolutions au maximum.

Il sera possible de créer et détruire des instances de composants, des connections de données, et des connections d'événements. Vous rendrez la création de valeurs d'attributs automatique dès que le type de composant instancié sera choisi.

Votre éditeur se basera sur le modèle de la librairie standard IACA que vous avez développé lors de la tâche 1.

Vous intégrerez également à l'éditeur graphique le moteur de vérification de règles de cohérence Check, de manière à ce que l'utilisateur soit au courant de ses erreurs. Un exemple se trouve en Figure 2

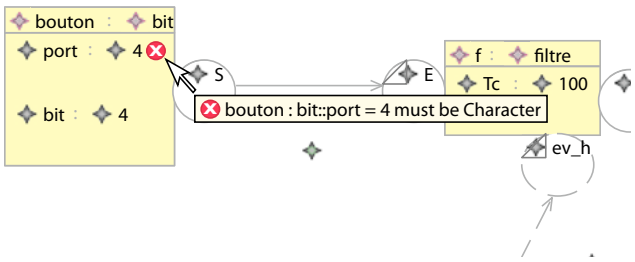


FIGURE 2. Intégration du moteur de vérification `Check`

3. Ressources et astuces

Le `site institutionnel` de GMF est hébergé par Eclipse. Cependant, je vous suggère de suivre le tutoriel intitulé "`Learn Eclipse GMF in 15 minutes`" (en utilisant bien sûr votre propre métamodèle. Malheureusement, ce tutoriel s'appuie sur une version précédente de GMF. La différence majeure est qu'il est nécessaire de définir des définitions de figure (`FigureDescriptor`) au lieu de spécifier directement des figures (§ `The GMF graphical definition model`). Une autre différence est la possibilité d'affixer des noeuds (même paragraphe). Pour vous aider dans votre découverte de GMF, je vous ai préparé une brève description du métamodèle GMF dans le `document` que vous pourrez trouver sur le site du cours. Essayez de créer vos modèles vous-même en suivant ce document au lieu de faire confiance aux assistants de création. La génération d'un modèle `GMFGEN` à partir d'un `GMFMAP` se fait par le menu contextuel, tout comme la génération du code pour l'éditeur graphique. Il n'y a pas (à ma connaissance) de moyen de mettre à jour un modèle `GMFGEN`, il faut donc le détruire à chaque mise à jour.

L'intégration du moteur de contraintes GMF est décrite dans le `document openArchitectureWare` au chapitre II.10.4. Il est par avant nécessaire de faire l'action décrite dans le document en section II.10.3. Attention: il existe une erreur dans

les patrons de génération du projet importé. Vous voudrez donc bien remplacer le fichier `org.openarchitectureware.adapter.gmf2/templates/aspects/xpt/editor/ValidateAction.xpt` par celui fourni `sur le site du cours`.

Je vous conseille d'adopter une démarche itérative dans laquelle vous réaliserez en priorité une version partielle et minimale de la syntaxe graphique que vous ferez évoluer pas à pas par la suite.

Bien sûr, je vous encourage à vous inspirer de l'exemple des `diagrammes d'états`.

4. Délivrables

- Vos métamodèle (`.ecore`), modèle de génération (`.genmodel`), modèle graphique (`.gmfgraph`), modèle d'outils (`.gmftool`), modèle de mapping (`.gmfmap`) et modèle de génération graphique (`.gmfgen`).
- Les spécifications `Check` (et `Xtend` si vous en avez).
- Des modèles exemples, bien formés et mal formés du regard des contraintes de cohérence, et les diagrammes correspondants.
- Votre librairie standard IACA.

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