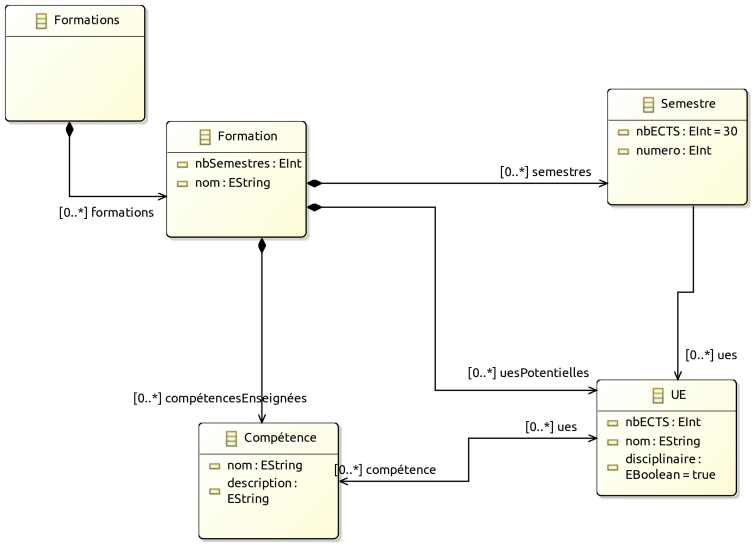
# Définition d'une syntaxe graphique avec Sirius

Nous avons vu comment créer un métamodèle, et manipuler un modèle conforme à ce nouveau métamodèle. En revanche, pour saisir les modèles à manipuler, nous avons utilisé l'éditeur réflexif qui permet de saisir un modèle via l'arbre formé par les liens de composition, à partir de la racine. Cela revient à saisir le modèle en syntaxe abstraite, ce qui est finalement assez logique puisque l'on ne connaît pas de syntaxe concrète. Nous allons ici définir une syntaxe concrète (et graphique, par opposition à une syntaxe textuelle) pour un petit métamodèle, et générer le modeleur correspondant, grâce à l'outil Sirius.

## 1 Un petit métamodèle pour la mise en place de formations universitaires

On vous propose de travailler sur le petit métamodèle de formations présenté ci-dessous. Chaque formation a un pool d'UEs potentielles, qui peuvent être attachées à des semestres. On vous fournit le métamodèle (son fichier ecore). Les UEs permettent d'acquérir des compétences. La formation se doit de faire acquérir un certain nombre de compétences.




## 2 Présentation de la syntaxe graphique souhaitée

On souhaite la syntaxe suivante :

- une formation est représentée par une ellipse orange avec le nom de la formation dedans,
- les UEs sont représentées par des carrés avec le nom de l'UE, de taille proportionnelle au nombre d'ECTS de l'UE (les carrés seront bleus pour les UEs disciplinaires et gris pour les non disciplinaires),
- les compétences sont représentés par des rectangles marron clair avec le nom de la compétence,
- les semestres sont représentés par des rectangles de la couleur de votre choix, avec le numéro du semestre précédé d'un S.
- les UEs d'un semestre lui sont liées avec un trait un peu large
- les compétences d'une UE lui sont liées avec une ligne fine
- les semestres d'une formation lui sont liées par une flèche

# 3 Sirius, installation, présentation, et préparation

## 3.1 Installation

Installez Sirius via le marketplace d'Eclipse.

## 3.2 Présentation succincte

La documentation de Sirius se trouve ici : `https://www.eclipse.org/sirius/doc/`. Sirius va permettre d'associer des éléments graphiques à des éléments du métamodèle : essentiellement à des instances de métaclasses et des liens issus de méta-associations. Sirius permet également de créer des palettes d'outils permettant de créer un modeleur, et des règles de vérification. Quand on crée une syntaxe avec Sirius, on fait évidemment souvent référence au métamodèle pour lequel on construit la syntaxe. On a besoin de se référer à des métaclasses par exemple, ou de naviguer le métamodèle. Quand vous trouverez des champs verts dans Sirius, vous devez y saisir des méta-classes (préfixées par le nom du métamodèle, par exemple ici : maquette.UE ou maquette.Formation). Quand vous trouverez des champs jaunes, cela veut dire que le contenu du champ sera interprété. Il s'agit en général d'une navigation dans le métamodèle, à partir d'un élément contexte. Vous pouvez y placer des noms de caractéristiques comme : feature :nom (dans le contexte d'une métaclasse ayant une caractéristique nom) ou des expressions Acceleo. Acceleo est un environnement de transformation Model2Text permettant notamment la génération de code à partir de modèle. La documentation qui devrait être suffisante pour ce TP concerne AQL (Acceleo Query Language) [1] pour écrire des expressions Acceleo. Vous verrez que la philosophie et la syntaxe sont très proches de celles d'OCL. Nous ne verrons pas en détail toutes les fonctionnalités de Sirius.

## 3.3 Préparation du métamodèle

- Créez un projet de modélisation (Modelling project).
- Déposez-y maquette.ecore (dans un répertoire model).
- Associez-y un fichier de génération de modèle (genmodel) en procédant comme vous l'aviez fait pour l'exemple de cartes.
- Ouvrez le genmodel et demandez la génération de tout.
- Il apparaît 3 nouveaux projets dans votre workspace.
- Cliquez droit sur votre projet initial, puis run as Eclipse Application.
- Une nouvelle instance d'Eclipse devrait s'ouvrir, cela revient en quelque sorte à disposer d'une instance d'Eclipse où un plug-in maquette aurait été déployé.
- Dans toute la suite du TP, on travaillera uniquement dans cette deuxième instance d'Eclipse.

# 4 Définition de la syntaxe graphique avec Sirius

## 4.1 Préparation d'un projet de test et d'un projet pour la définition de la syntaxe

Nous allons créer un projet le projet de test, c'est-à-dire celui qui contiendra le ou les instances de maquettes, destiné à vérifier que votre définition de syntaxe graphique est opérationnelle. Ce n'est pas toujours mentionné, mais bien sûr vous devez tester au fur et à mesure ce que vous faîtes.

1. https ://www.eclipse.org/acceleo/documentation/

- Créez un projet de type Modelling Project, par exemple nommé maquette.test.
- Dans ce projet, créez un modèle de maquettes : new other maquette model (notez que les maquettes apparaissent dans les menus dans cette seconde instance d'Eclipse), de nom formation1.
- Choisissez comme "Model Object" Formations (la racine du modèle sera une instance de Formations).
- Créez quelques éléments de modèles et quelques liens avec l'éditeur réflexif, par exemple une formation avec 4 semestres, dont vous ne détaillez que le premier avec 2 UE.

Nous allons maintenant créer le projet sirius destiné à mettre au point la syntaxe graphique :

- Créez un projet Sirius ViewPointSpecification Project, par exemple nommé maquette.design.
- Un point de vue (ViewPoint) est normalement déjà créé (sinon créez-en un), qui représente en Sirius la façon d'associer une vue graphique à un modèle. Renommez ce viewpoint si vous le souhaitez.
- Dans le Sirius Specification Editor nous allons créer une représentation graphique, ici par un diagramme (il en existe d'autres que nous ne verrons pas ici) : New Representation, Diagram Description.
- Définissez l'identifiant (qui doit être unique, par exemple formations), et le métaclasse qu'il représente : maquette.Formations. Vous pouvez changer le label (par exemple diagramme de formations).
- Testez : retournez dans le projet de test, cliquez droit sur la racine du projet et sélectionnez New Representation puis other ... puis sous le viewpoint que vous venez de manipuler, la représentation graphique que vous venez de créer.
- Une vue graphique s'ouvre, mais oh, déception, elle est vide. C'est normal car pour l'instant nous n'avons pas défini comment représenter le modèle. Nous avons juste dit que les maquettes se représentaient avec des diagrammes de formations.

## 4.2 Une syntaxe pour les UEs

Revenons dans le projet de design. Pour donner la représentation des UEs, on doit créer un nœud dans le calque Default. Nous verrons utérieurement ce que représente la notion de calque. Sur le calque Default (dans le Sirius Specification Editor), clic droit, new Diagram Element, Node. Dans l'onglet General qui s'ouvre alors :

- on donne un identifiant, par exemple ici UENode
- on saisit la Domain Class, c'est-à-dire la métaclasse pour laquelle on construit la représentation graphique, ici : maquette.UE.

Ensuite, on ajoute à notre nouveau nœud son style graphique : clic droit sur le nœud, new Style, puis choisir un carré et inspecter les properties de l'élément.

- Dans l'onglet Label : on peut régler les paramètres du label. Décocher Show Icon (sinon ça ajout un petit losange représentant les noeuds. Dans Label Expression, on peut définir une expression Acceleo pour définir le label en fonction de caractéristique de la métaclasse représentée (ici : UE). Pour avoir le nom de l'UE, on mettra : [self.nom/].
- Dans l'onglet Color réglez la couleur.
- Dans l'onglet Advanced : empêchez le redimensionnement, et mettez une expression pour calculer la taille en fonction du nombre d'ECTS, avec une expression Acceleo comme par exemple : [2*self.nbECTS/]

Sauvez ensuite maquette.odesign (attention, il faut parfois sortir le contrôle de l'onglet properties avant de sauver, étrange ...). Et retournez dans le projet de test sur le diagramme de votre master préféré : les UEs sont maintenant des carrés bleus.

### 4.3 Une syntaxe pour les compétences, les semestres et la formation

Procédez de même pour les compétences, les semestres et la formation.

### 4.4 Liens semestres-UEs et UEs-compétences

Créez une Relation Based Edge pour les liens entre semestres et UEs.

— Dans l'onglet General mettre un identifiant et positionner source Mapping au nœud des semestres et Target Mapping au nœud des UEs, puis Target Finder Expression à : feature :ues (ou [self.ues/] pour indiquer quelle relation est à prendre en compte dans le métamodèle.

— Puis réglez les propriétés graphiques (sans flèche, et avec ligne épaisse, une couleur si vous le souhaitez)

Procédez de la même façon pour les compétences des UEs. Retournez dans le projet de test sur le diagramme de votre master préféré. C'est chargé !

### 4.5 Un autre layer pour les compétences

Déplacez les compétences (le nœud des compétences et celui de la relation entre UEs et compétences) dans un nouveau layer. Dans le diagramme de test vous pouvez maintenant les faire apparaître ou disparaître facilement.

### 4.6 Représentation graphique conditionnelle

Pour permettre l'affichage des UEs de couleur différente selon que l'UE est disciplinaire ou non, on doit définir un style conditionnel pour les UE. Placez vous sur le nœud des UEs dans maquette.odesign, et ajoutez un style conditionnel, conditionné par [not self.disciplinaire/]. Créez sous ce style conditionnel le nouveau style pour les UEs non disciplinaires.

### 4.7 Restreindre les éléments affichés

Si vous créez un autre modèle de formation, par exemple pour votre licence préférée, vous remarquerez que les éléments créés côté licence s'affichent aussi sur le diagramme de votre master, ce qui dans notre cas n'est pas souhaitable. Ceci est du au fait que dans nos vues graphiques on a juste dit qu'il fallait afficher les éléments d'un certain type, sans restriction. Pour pallier ce problème [2] nous allons restreindre dans un premier temps l'affichage des UEs. Pour cela, retournez dans le nœud correspondant aux UEs (dans maquette.odesign). Dans le champ jaune Semantic Candidate Expression, ajoutez : [self.formations->collect(uesPotentielles)/] pour spécifier qu'on ne veut sélectionner que les UEs potentielles des formations de l'instance courante de Formations. Procédez de même pour les autres éléments.

## 5 Préparation de la palette d'outils

Jusqu'ici, nous avons juste permis d'afficher un modèle avec une syntaxe graphique que nous avons définie. Si nous voulons obtenir un modeleur, il nous faudra disposer dans la palette d'outils de création et de manipulation des éléments de modèle. La palette est, vous l'aurez constaté amèrement, actuellement vite. Nous allons ici remédier à cela [3]

2. vous noterez au passage que pallier est un verbe transitif direct, souvent mal utilisé
3. vous remarquerez que remédier est transitif indirect.

### 5.1 Mise en place d'une section d'outils

Placez vous dans le layer par défaut (dans maquette.odesign). Puis cliquez droit, new Tool, Section. Donnez un identifiant à la section, et un label sympathique pour un utilisateur.

### 5.2 Ajout d'une outil de création d'une formation dans la palette

Dans votre section, ajoutez un outil de création de formation : new Element Creation, node creation.

- Donnez un identifiant et un label sympathique à votre outil
- Associez-y le nœud des formations via le champs Node Mappings.

Il s'est créé sous votre outil 2 variables auxquelles vous ne toucherez pas :

- Node Creation Variable container qui pointe sur l'élément sémantique cliqué lors de l'utilisation de l'outil, ici ce sera donc une Formation, donc cette variable sera de type maquette.Formation.
- Container View Variable containerView qui pointe sur la représentation graphique de l'élément cliqué.

Il se crée aussi un élément Begin, qui va servir à regrouper les actions à effectuer lors de l'utilisation de l'outil. Ajoutons des éléments à ce Begin :

- Clic droit, New operation, Change Context, puis New Operation, create instance. Cela va nous permettre de positionner pour les actions :
  - Reference Name : c'est la référence où l'on peut trouver les éléments de type Formation dans le container, donc ici : [formations]

  Testez en ajoutant à votre maquette une nouvelle formation (une très belle licence par exemple).

### 5.3 Ajout d'un outil de création d'une UE dans une formation

C'est un peu plus compliqué, car il faut lors de la création que l'utilisateur sélectionne la formation dans laquelle ranger l'UE créée. On utilisera pour cela un selection wizard plutôt qu'un node creation.

### 5.4 Ajout d'un outil de lien semestre-UE

On ajoute une new edge creation à notre section d'outils. Quatre variables sont alors créées, notamment source et target (le premier et deuxième élément sémantique cliqué). Ajoutez une action set qui positionne ues à target (en fait qui rajoute target à ues). Pour cela, dans le champ Feature Name mettre ues et dans le champ value mettre [target/].

### 5.5 Terminer la palette

Teminez la palette avec les autres outils nécessaires (y compris dans l'autre layer).

## 6 Ajout d'outil de vérification/validation de modèle

Sirius inclut le moyen de vérifier des propriétés sur le modèle. On va ainsi pouvoir détecter que le modèle viole des règles de validation. Nous allons ici écrire une règle vérifiant qu'une formation a bien défini le bon nombre de semestres. Pour cela, nous allons créer une validation pour le diagramme (clic droit sur le diagramme dans maquette.odesign, new validation). Puis écrivons la règle de validation : new Semantic validation rule. Vous pouvez y définir la cible (les formations), le niveau de gravité, ainsi que le message. Définissez ensuite un audit (avec

une expression acceleo, rappelez vous que la syntaxe est quasiment la même que celle d'OCL). Ajoutez d'autres règles de validation (par exemple : chaque compétence d'une formation doit être couverte par au moins une UE présente dans la formation). Notez que ces règles correspondent à des invariants qui doivent être respectés une fois le modèle terminé, et pas en cours de construction.

# 7 S'il vous reste du temps

— Peaufinez la syntaxe graphique et les outils.
— Ajoutez des outils de modification des éléments, et d'édition.
— Ajoutez des règles de validation.
— Explorez les vues à base de tables plutôt que les vues graphiques (ces vues peuvent être pratiques par exemple pour associer les UEs et les compétences).