Documentation succincte sur GMF

Frédéric Fondement
frederic.fondement@uha.fr

GMF est un outil qui réalise un éditeur graphique pour un langage dont la syntaxe abstraite est donnée sous forme de métamodèle ecore. Pour se faire, GMF fournit un langage pour la description de la syntaxe concrète. Cette description se déroule en 3 étapes, chacune étant la description d'un modèle, comme résumé à la Figure 1:

- la description des figures de représentation et des points de variation (gmfgraph),
- la description des outils (gmftool),
- la description des correspondances entre les concepts du métamodèle, les figures, et les outils (gmfmap).

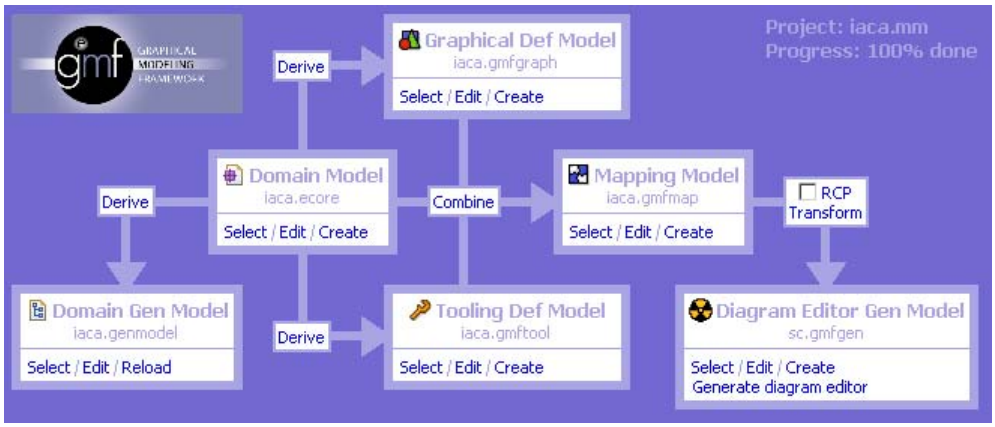

FIGURE 1. Approche GMF pour la spécification de syntaxes concrètes graphiques.

Ces description sont complétées par un modèle de génération graphique (gmfgen) qui, à l'instar du modèle de génération du métamodèle, synthétise les informations provenant de ces différents modèles, en y ajoutant des informations de personnalisation de la génération du code de l'éditeur graphique. GMF intègre une transformation de modèles permettant de générer ce dernier modèle à partir des trois précédents, et un générateur de code générant une extension ("plug-in") à Eclipse permettant d'éditer graphiquement des modèles pour le langage considéré.

Afin de produire ces trois modèles, GMF propose trois langages spécifiques. La description succincte de leur métamodèle fait l'objet du présent document. Ce document se limite cependant à une description des concepts les plus essentiels de ces métamodèles. Pour une description plus complète, veuillez vous référer à la documentation officielle (voir le site http://www.eclipse.org/gmf/). pour une description par la pratique, il existe de nombreux tutoriels, comme sur http://www.ibm.com/developerworks/library/os-ecl-gmf/index.html.

1. Description des figures (`gmfgraph`)

Le métamodèle du langage `gmfgraph` est donné par la Figure 2. La définition des figures est donnée par la classe `Canvas`. Elle contient la description des figures graphiques (`FigureGallery`), des noeuds (`Node`), des connections (`Connection`), des compartiments (`Compartment`) et des labels (`DiagramLabel`).

La galerie des figures décrit les figures (`FigureDescriptor`), et les figures (`Figure`) qui pourront être réutilisées dans diverses descriptions de figures, par exemple les flèches qui seront placées au bout de figures de connection (comme les associations UML).

En plus de décrire la figure à laquelle elle correspond, un descripteur de figure pourra définir un accesseur à un élément particulier de son “anatomie” via un accesseur (`ChildAccess`). Dans l’exemple du langage de diagrammes de classes UML, le descripteur de la figure pour une classe sera un rectangle, contenant notamment un texte dans lequel sera placé le nom de la classe. Le texte, en tant qu’élément de figure sera rendu accessible au reste de la spécification par un accesseur.

Un noeud (`Node`) est la formalisation d’une figure; il se réfère à un descripteur de figure particulier, et sera réutilisé dans la mise en correspondance du modèle `gmfmap`. Une connection (`Connection`) est la même chose qu’un noeud, à ceci près quelle est spécialisée dans la représentation de connections (comme les associations UML, une figure de classe étant elle plutôt un noeud). Il est à noter qu’un noeud peut être affixé à l’élément qui le contiendra. Un compartiment (`Compartment`) est une partie de des-

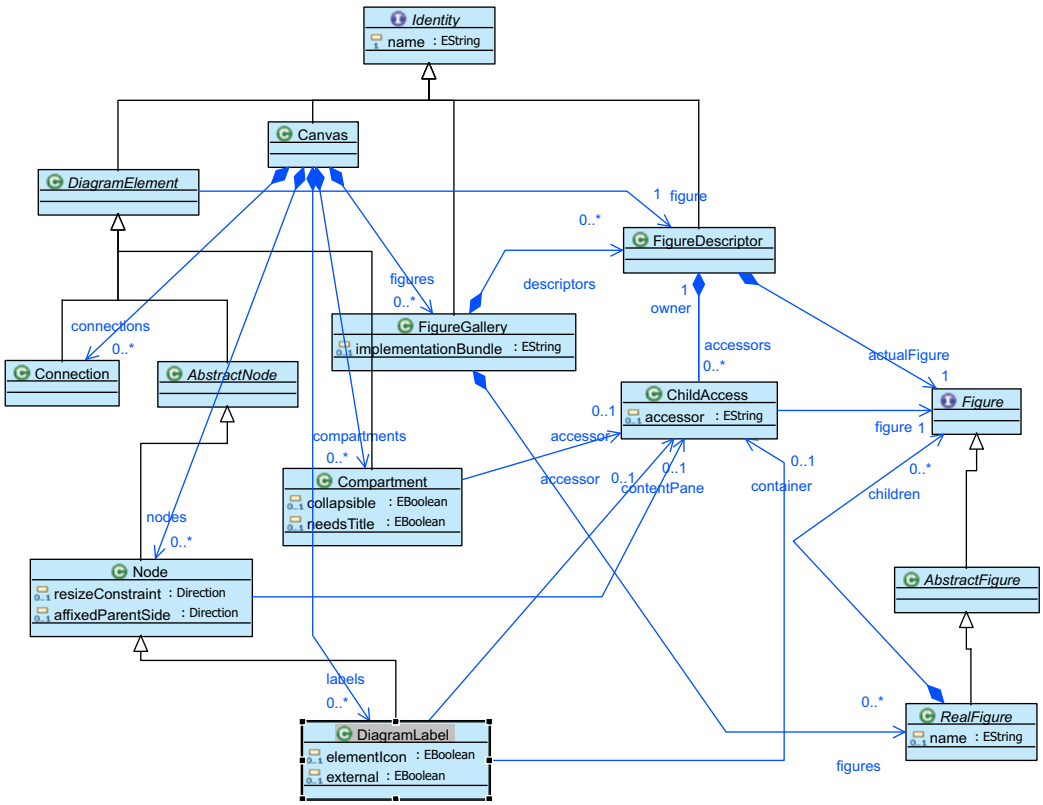

FIGURE 2. GMFGRAPH: main concepts

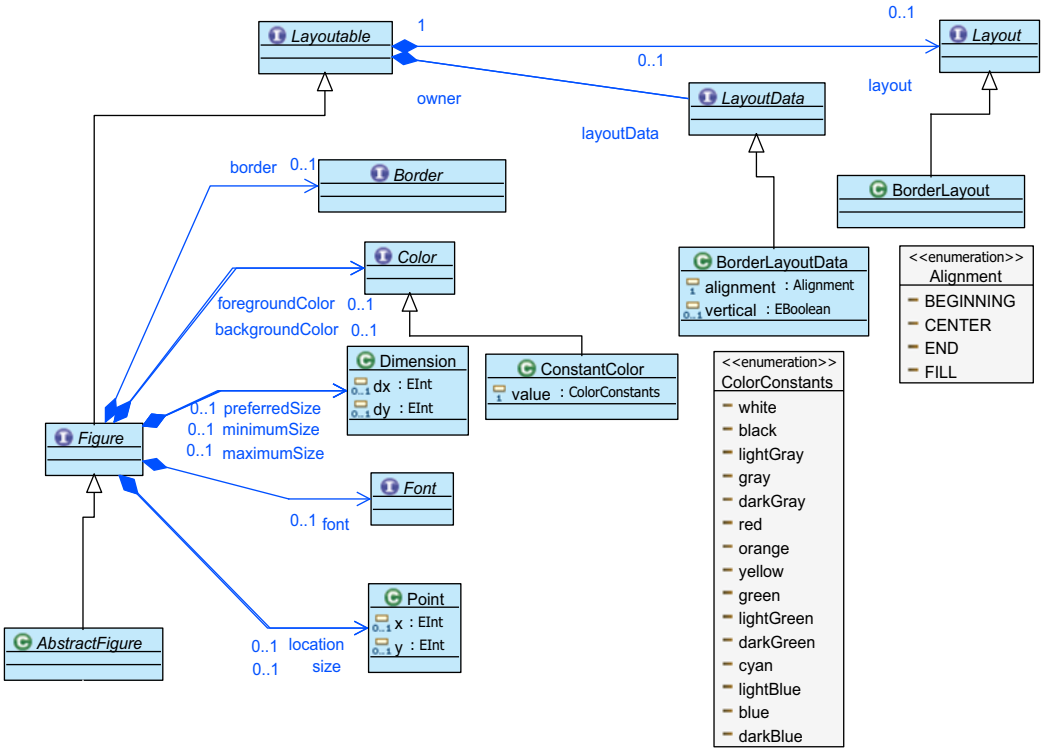

FIGURE 3. GMFGRAPH: Propriétés génériques des figures.

cription de figure qui sera capable de contenir d'autres éléments. Dans l'exemple d'UML, on pourra penser à l'endroit où doivent apparaître les attributs d'une classe. La description de figure référencée sera celle pour la classe, et la partie de la figure sera donnée par un accesseur. Les labels de diagramme (`DiagramLabel`) permettent eux d'accéder à des éléments textuels de certaines figures, toujours via leurs accesseur. Ceci permettra, lors de la mise en correspondance `gmfmap`, de synchroniser une donnée du modèle avec une valeur textuelle représentée.

Les attributs génériques d'une figure sont donnés en Figure 3. Une figure, telle que contenue dans une autre figure (`RealFigure`), est un élément qui doit être placé à un endroit précis. Pour se faire, on peut lui donner certaines règles d'agencement (`LayoutData`), correspondant à l'agencement décidé dans la figure hôte (`Layout`). Diverses stratégies d'agencement existent, comme par exemple la stratégie "de bord" (`BorderLayout`), "de pile" (`StackLayout`), "de flot" (`FlowLayout` - format par défaut) ou de position (`XYLayout`). On peut également préciser (dans une certaine mesure) les dimensions (`Dimension`) préférées, minimales et maximales, pour peu que les stratégies de d'agencement en tienne compte. Une figure peut définir une bordure (comme une ligne - `LineBorder` ou une marge - `MarginBorder`, voir multiples comme `CompoundBorder`), avoir une couleur (`Color` - qu'elle soit constante - `ConstantColor`, ou donnée par ses valeurs RGB - `RGBColor`) et, si utile, une police de caractère (`Font`, comme `BasicFont`).

Les divers types de figures sont donnés en Figure 4. Il est inutile de présenter les zones de texte (`Label`), rectangles (`Rectangle` et `RoundedRectangle`) et ellipses (`Elipse`). Une ligne brisée (`Polyline`) est une suite de segments ouverts, alors

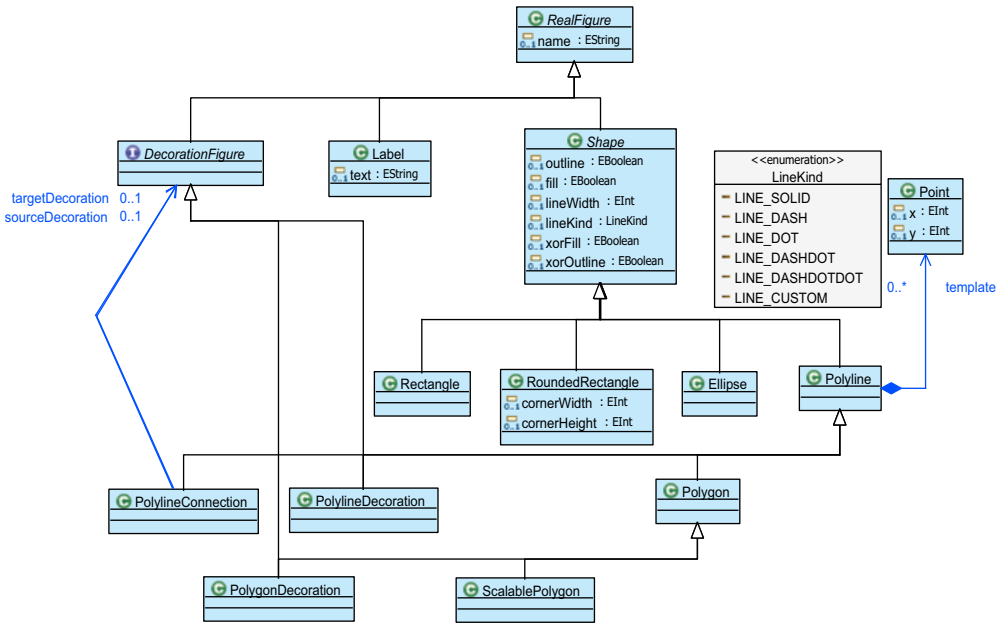

FIGURE 4. GMFGRAPH: Types de figures.

qu'un polygone (`Polygon`) est une suite fermée. Certaines lignes multiples sont spécialisées dans les connections (`PolylineConnection`) et peuvent exposer une figure à son début et à sa fin, dont l'orientation dépendra de la pente des extrémités. Ces décorations (`DecorationFigure`) peuvent être des lignes brisées (`PolylineDecoration`) ou des polygones (`PolygonDecoration`).

2. Palette d'outils (`gmftool`)

La palette d'outils définit quels sont les outils disponible dans l'éditeur graphique. L'outil le plus important est l'outil de création, qui permet d'instancier un élément du métamodèle, qu'il soit représenté par un noeud ou une connection. Les concepts reliés à la création sont représentés en Figure 5.

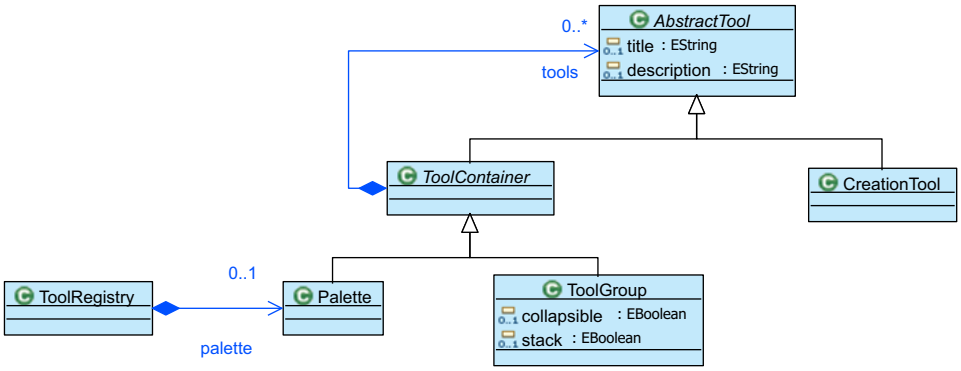

FIGURE 5. GMFTOOL: Outils de création.

Le modèle est représenté par le registre d'outil (`ToolRegistry`). Ce registre référence une palette d'outils (`Palette`) composés d'outils ou de groupes d'outils (`ToolGroup`). Un de ces outils est l'outil de création (`CreationTool`).

Ce modèle permet en outre de créer des menus et des menus contextuels. Des images (`Image` comme `DefaultImage` ou `BundleImage`) peuvent être définies pour les outils et les éléments des menus.

3. Description des correspondances

Le dernier modèle spécifique à la description de syntaxes graphiques est le modèle de correspondance. Il est en charge de la mise en relation de la syntaxe graphique proprement dite avec la syntaxe abstraite représentée par le métamodèle. Au passage, il spécifie également les actions prises par les éléments décrits dans la palette d'outils.

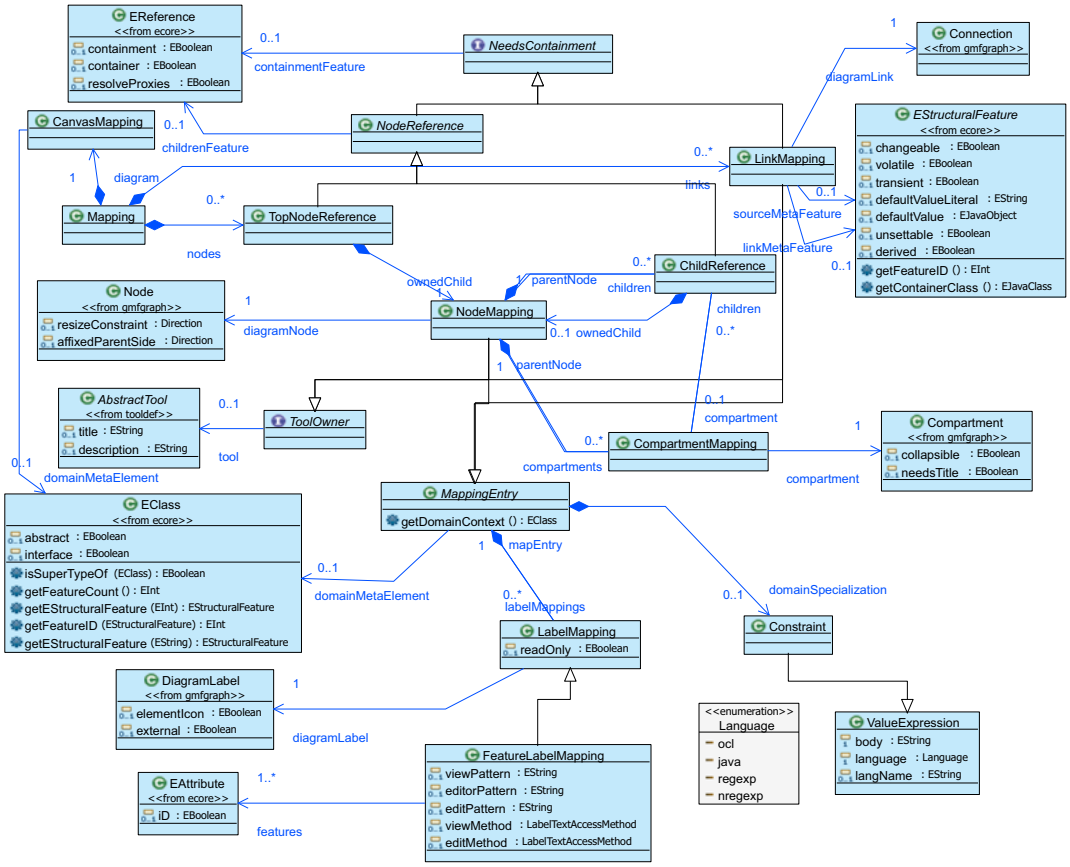

FIGURE 6. GMFMAP

La Figure 6 décrit les éléments les plus importants du métamodèle GMFMAP. L'objet principal est `Mapping`, qui contient notamment la description des éléments mis en relations en général (`CanvasMapping`), et le détail des mises en correspondance pour les noeuds (`NodeMapping`) et les relations (`LinkMapping`).

La mise en relation générale (`CanvasMapping`) indique la métaclasse maître du modèle, c'est à dire la métaclasse dont une instance sera l'élément racine du modèle (`domainMetaElement`, notée `Element` dans l'éditeur réflexif). Elle référence

également la palette (`Palette` dans l'éditeur réflexif) du modèle `GMFTOOL`, et le `Canvas` (`Diagram Canvas` dans l'éditeur réflexif) du modèle `GMFGRAPH`.

Les noeuds qui peuvent être représentés directement dans un diagramme (c'est à dire les noeud qui ne doivent pas forcément apparaître dans le compartiment d'un autre noeud) doivent être déclarés par un `TopNodeReference`. Le `TopNodeReference` indique comment l'élément de modèle correspondant au noeud peut être retrouvé à partir de l'élément de modèle racine en donnant la référence de composition du métamodèle qui doit être naviguée (`containmentFeature`). Un `TopNodeReference` doit nécessairement spécifier une description de correspondance de noeud (`ownerChild`).

Une description de correspondance de noeud (`NodeMapping`) met en relation une métaclasse du métamodèle (`domainMetaElement`, notée `Element` dans l'éditeur réflexif), un noeud du modèle `GMFGRAPH` (`diagramNode`) et un outil du modèle `GMFTOOL` (`tool`). Si un noeud peut en contenir d'autres via un compartiment de sa représentation, il doit contenir un `CompartmentMapping` d'une part (référençant le `Compartment` correspondant dans le modèle `GMFGRAPH`), et d'autre part d'autres correspondances de noeuds filles (`ChildReference`), contenant elles lesdites correspondances de noeuds (`ownedChild`), quelle référence naviguer pour les obtenir (`childrenFeature` et/ou `containmentFeature`) tout en indiquant la correspondance de compartiment précédemment définie (`compartment`). Une `ChildReference` peut également être définie pour tout élément affixé; dans ce cas, aucune référence à une correspondance de compartiment n'est nécessaire.

Deux cas doivent être distingués pour établir la correspondance de liens (`LinkMapping`). Dans le premier cas, la liaison est exprimée dans le métamodèle par une simple référence (`linkMetaFeature`, notée `TargetFeature` dans l'éditeur réflexif). Dans le second cas, la liaison est exprimée par une métaclasse. Dans ce dernier cas, à l'instar des correspondances de noeuds, il faut indiquer comment trouver ces correspondances depuis la classe "racine" (`containmentFeature`), ladite métaclasse (`domainMetaElement`, notée `Element` dans l'éditeur réflexif), en plus de fournir les références pour atteindre la source (`sourceMetaFeature`) et la cible (`linkMetaFeature`, notée `TargetFeature` dans l'éditeur réflexif) de la connection. Il faut bien sûr dans les deux cas indiquer la connection du modèle `GMFGRAPH` (`diagramLink`) et éventuellement l'outil du modèle `GMFTOOL` (`tool`).

Il peut être nécessaire dans les correspondances de noeuds et de liens d'exposer dans la représentation certaines de leurs propriétés sous forme de texte. Par exemple, dans le langage de diagrammes de classes UML, le nom d'une classe apparaît dans sa représentation. Pour se faire, une correspondance peut contenir une correspondance (`FeatureLabelMapping`) entre ladite propriété (`features`) et le label de diagramme du modèle `GMFGRAPH` (`diagramLabel`).

Au cas où on ne veut représenter un élément de modélisation avec un noeud ou une connection donnée que sous certaine condition, il est possible d'adjoindre à la correspondance concernée une contrainte type OCL (`Constraint`). Ceci permet par exemple d'avoir différentes représentations pour un même élément, dont le choix dépendra de certaine propriétés sur un modèle.