Motif pour la métamodélisation: Commentaire attribué

Alain Plantec, Mickael Kerboeuf, Vincent Ribaud

▶ **To cite this version:**

Alain Plantec, Mickael Kerboeuf, Vincent Ribaud. Motif pour la métamodélisation: Commentaire attribué. Deuxième atelier de IDM07 sur les motifs de métamodélisation, Jan 2007, France. pp.x-y. hal-00504322

HAL Id: hal-00504322

https://hal.univ-brest.fr/hal-00504322v1

Submitted on 20 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Motif pour la métamodélisation

Commentaire attribué

Alain Plantec — Mickaël Kerboeuf — Vincent Ribaud

LISyC - EA3883, Université de Bretagne Occidentale
équipe IDM (Ingénierie Des Modèles)
C.S. 93837, 29238 Brest cedex 3
{plantec,kerboeuf,ribaud}@univ-brest.fr

RÉSUMÉ. *Le motif “Commentaire attribué” montre comment intégrer le concept de commentaire dans un métamodèle*

ABSTRACT. *"Attributed comment" pattern shows how to integrate comments into a metamodel*

MOTS-CLÉS : *métamodèle, commentaire*

KEYWORDS: *metamodel, comment*

1. Motivation

Suivant une approche classique, un commentaire est une méta-information formelle documentaire (Uschold, 2003), placée dans un modèle pour ajouter du sens à un ou plusieurs éléments de modélisation. Il s'agit d'une source d'informations parfois indispensable à la compréhension des modèles. Alors que les commentaires sont porteurs de sens, la notion de commentaire est souvent absente des métamodèles. Ils sont considérés comme des décorations dont l'intérêt est pris en compte au niveau de la syntaxe concrète mais souvent ignorés au niveau abstrait. La structure interne d'un commentaire est par nature libre du point de vue du langage hôte.

Inversement à l'approche classique, *la programmation littérale* (Beebe, 2002) prône l'intégration du code dans la documentation. Cette technologie originalement développée dans *WEB* (Knuth, 1983) introduit la notion de lien explicite entre la documentation d'un système et son code : le source d'une application consiste en une documentation dans laquelle le code à compiler ou à interpréter est inséré. Les outils qui supportent cette technologie produisent automatiquement d'une part la documentation finale (en *Latex* dans *WEB*) et d'autre part le code du système.

Les évolutions plus récentes de la *programmation littérale* exploitent le concept de *commentaire structuré*. Un modèle ou un code source comprend alors des commentaires qui respectent une syntaxe propre. Ici encore, le lien entre la documentation et le modèle est explicite. Les outils comme *javadoc* ou encore *doxygen* exploitent cette technologie. Le concept de commentaire fait alors bien partie du métamodèle du langage.

2. Indication d'utilisation

Le motif peut être utilisé pour permettre aux commentaires d'être contenus dans les méta-données et donc d'être échangés entre différents outils sans perte d'information favorisant ainsi l'intéropérabilité des systèmes et outils.

Une utilisation typique est la mise en oeuvre de "pretty-printer" ou d'outils de restitution de code source à partir des méta-données. Les commentaires peuvent être mis en valeur, traduits ou encore être utilisés comme spécification source à d'autres outils comme des générateurs de code ou de documentation.

3. Structure

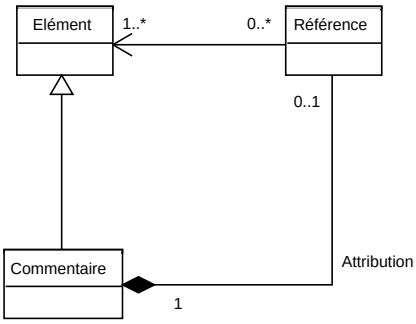

Figure 1. *Structure du motif "commentaire attribue"*

4. Constituants

4.1. *Elément*

Représente un élément du métamodèle.

4.2. *Commentaire*

Un *Commentaire* est une documentation dans une spécification. A ce niveau, un commentaire correspond à une documentation libre qui fait partie de la spécification et qui donc est elle même un élément du métamodèle. Un *commentaire structuré* serait typiquement mis en oeuvre comme une sous-classe de *Commentaire*

Un commentaire peut ne pas être attribué (commentaire libre) ou bien concerner un ou plusieurs éléments de métamodèles (commentaire attribué).

4.3. *Référence*

Comme dans le motif “Elément nommé” (Savaton *et al.*, 2006), un *Référence* représente la désignation d’un élément du méta-modèle. Le moyen par lequel un *Référence* désigne l’élément du métamodèle n’est par contre pas précisé. Il pourrait s’agir d’un nommage par un identifiant ou encore d’un positionnement par coordonnées cartésiennes. Typiquement, pour représenter ces deux possibilités de désignation, on introduirait deux sous-classes de *Référence*.

Un *Référence* est toujours associé à un ou plusieurs éléments de métamodèle qu'il désigne. Tout élément de métamodèle est référençable plusieurs fois (plusieurs commentaires peuvent lui être attribués) mais peut ne pas être référencé (une instruction n'est généralement par référençable par exemple).

5. Exemple

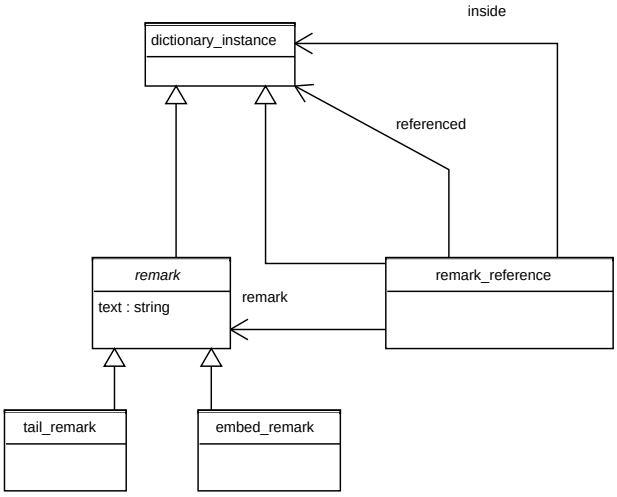

Figure 2. *Modélisation des commentaires du langage EXPRESS*

L'exemple présenté par la figure 2 montre la partie d'un métamodèle d'EXPRESS (ISO 1994a) concernant les commentaires. Comme le montre l'extrait de modèle EXPRESS de la figure 3, un commentaire peut directement contenir l'identifiant d'un élément de model, le commentaire est alors attribué à cet élément. Ainsi, un commentaire ne peut-être attribué qu'à un seul élément de métamodèle dont l'identifiant est indiqué en tête de contenu de commentaire.

L'entité *dictionary_instance* joue le rôle de *Elément*. En *EXPRESS*, on dispose de deux représentations des commentaires, soit sur une ligne, soit sur plusieurs lignes, ces deux types de commentaires sont respectivement représentés par *tail_remark* et par *embed_remark*. La documentation est contenue dans une chaîne de caractères (attribut *text*).

Une instance de *remark_reference* établit un lien entre le *dictionary_instance* commenté (*referenced*), le *dictionary_instance* dans le contexte duquel le commentaire est placé (*inside*) et le commentaire (*remark*). *remark_reference* joue ainsi le rôle de *Référence*.

```
SCHEMA Node;
(*"WaitNode"
 * I am a waiting node, that never comes back
 * once it starts. It must have at least
 * one input, to send the signal to start.
 *)
ENTITY WaitNode
  SUBTYPE OF ( SynchroNode );
  signal : Data; --"signal" signal to stop waiting.
END_ENTITY;
```

Figure 3. *Exemples de commentaires attribués en EXPRESS*

Pour l'instanciation des méta-données, le compilateur de modèle EXPRESS résoud la référence vers le *dictionary_instance* commenté par résolution de symbole dans le contexte identifié par l'association *inside*.

Les métadonnées correspondant à l'exemple 3 encodées au format *STEP* (ISO 1994b) sont données dans la figure 4. Chaque instance est représentée par un identifiant numérique unique précédé du caractère `#`, suivit du nom de l'entité de l'élément puis de la liste des valeurs d'attribut entre parenthèses.

Les deux commentaires sont représentés par les instances `#15` et `#21` et les *remark_reference* liés sont respectivement les instances `#14` et `#20`. Par exemple, le *remark_reference* `#14` est placé dans le contexte du schéma *Node* (*schema_definition* `#2`) et désigne l'entité *WaitNode* (*entity_definition* `#8`)

6. Bibliographie

Beebe N. H., « A Bibliography of Literate Programming », http ://www.literateprogramming.com/litprog-bib.pdf, 2002.

ISO, *Part 11 : EXPRESS Language Reference Manual*. 1994a.

ISO, *Part 21 : Clear Text Encoding of the Exchange Structure*. 1994b.

Knuth D. E., The WEB system of structured documentation, Technical report, Stanford, CA, USA, 1983.

Savaton G., Delatour J., « Motif pour la métamodélisation : Elément nommé », *MP2006, Premier atelier de travail sur les motifs de métamodélisation, Lilles*, 2006.

Uschold M., « Where are the semantics in the semantic web ? », *AI Mag.*, vol. 24, n° 3, p. 25-36, 2003.

```
#2=SCHEMA_DEFINITION('Node',(),(),(#3,#8),());
#3=ENTITY_DEFINITION('Data',(),#2,#5,(),(),()...);
#5=SUPERTYPE_CONSTRAINT(#6,$);
#6=ENTITY_DEFINITION_REFERENCE(*,'Data',*,#3);
#8=ENTITY_DEFINITION('WaitNode',(),#2,$,(),(#10),()...);
#10=EXPLICIT_ATTRIBUTE('signal',#8,#11,$,.F.);
#11=NAMED_TYPE_REFERENCE(*,'Data',#3);
#14=REMARK_REFERENCE('WaitNode',$,#2,#8,#15);
#15=EMBEDDED_REMARK('"WaitNode"
      * I am a waiting node, that never comes back
      * once it starts. It must have at least
      * one input, to send the signal to start.
      ');
#20=REMARK_REFERENCE('signal',$,#8,#10,#21);
#21=TAIL_REMARK('"signal" signal to stop waiting.');
```

Figure 4. *Metadonnées au format STEP produites à partir de la compilation de l'exemple de la figure 3*