Un méta-modèle pour Systèmes d'Accès à des Bases de Composants Hétérogènes

Oualid Khayati, Agnès Front, Jean-Pierre Giraudin

Laboratoire LSR-IMAG
681, rue de la Passerelle, BP. 72
38402 Saint Martin d'Hères Cedex
FRANCE

Nom.Prenom@imag.fr

RÉSUMÉ. Les bibliothèques de composants sont des éléments clés dans les environnements de développement à base de composants et de réutilisation de composants. L'efficacité de tels environnements est optimale lorsque les développeurs et les concepteurs disposent de bibliothèques riches en composants testés et validés, ce qui augmentera la fiabilité et diminuera le temps de développement des systèmes d'information résultants. Malheureusement, mettre simplement à disposition des développeurs des bibliothèques de composants de taille importante n'augmentera pas forcément la productivité. La solution consiste à utiliser des techniques avancées de recherche et de classification de composants. Nous proposons dans cet article un méta-modèle pour la conception de modèles de bases de composants permettant d'améliorer l'organisation, la recherche et la sélection des composants.

ABSTRACT. Component based programming and components reuse, when practised on a large scale by communities of developers sharing software and conceptual components, will lead to the creation of large components collections. In such a context, facilities for querying a components collection become important. This paper presents a metamodel for a tool which can be used to organize and describe components in a form suitable for querying.

MOTS-CLÉS : réutilisation, recherche de composants, bibliothèque de composants, formalisme de composants, méta-modèle.

KEYWORDS: reuse, components retrieval, components repository, components formalism, metamodel

1. Introduction.

Les bibliothèques de composants sont désormais intégrées à tout processus de développement à base de composants et de réutilisation de composants. Leur rôle ne se limite pas à la gestion des composants logiciels pendant la phase de codage, mais il peut s'étendre sur tout le cycle de développement d'un système d'information. Plusieurs approches que nous pouvons classer en quatre catégories ont été proposées dans la littérature. La classification externe, qui exploite des informations externes aux composants pour les classer (par exemple la classification par facettes [6]), la classification structurelle qui extrait des informations sur la structure du composant (par exemple la classification par signature [5]), la classification comportementale qui s'intéresse aux aspects dynamiques et au comportement des composants pendant leur exécution [2] et la recherche par navigation qui exploite les relations sémantiques qui peuvent exister entre les composants [4].

Notre travail, a pour but de gérer des bases de composants hétérogènes tant du point de vue des modèles de composants que du point de vue des niveaux d'abstraction des composants gérés. Ainsi, nous proposons un système d'accès à des bases des composants (SABC) permettant d'aider les utilisateurs à sélectionner les composants répondant le mieux à leurs besoins, d'assister l'utilisateur pour la réutilisation des composants et de piloter le processus de développement de systèmes d'information.

Pour atteindre ce but, notre SABC s'appuie sur une base qui unifie les descriptions des différents modèles de composants des bases sources. Dans la section 2 de cet article nous présentons le méta-modèle du SABC que nous proposons. Dans la section 3 nous présentons un exemple de modèle partiel d'une base de composants instance du méta-modèle.

2. Le méta-modèle M-Sigma.

Le méta-modèle que nous proposons est une adaptation du méta modèle d'UML [1]. Trois « core packages » sont enrichis pour répondre à nos besoins spécifiques : Le « backbone package », le « relationships package » et le « classifiers package ». Les nouvelles méta-classes sont les suivantes : *ComponentsRepository, Description, Item, Advanced, Property, DescriptionFeature, AssociationDescription, UmlClassDiagram.*

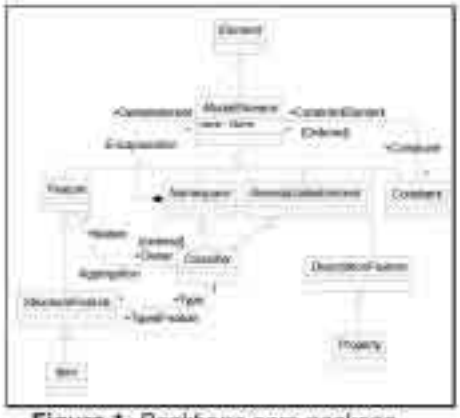

Figure 1. *Backbone core package.*

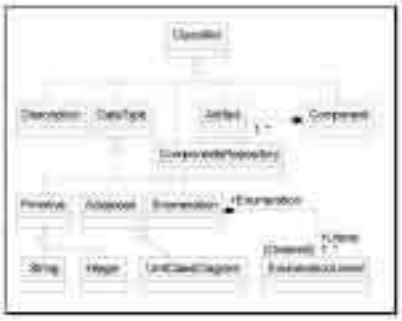

Figure 2. *Classifiers core package.*

– Element : c'est la méta-classe abstraite qui se trouve au sommet de la hiérarchie des méta-classes du méta-modèle.

– ModelElement : c'est une abstraction de tous les éléments de modélisation servant à la modélisation d'une base de composants. L'attribut *Name* est défini dans la classe abstraite ModelElement, ainsi tout élément de modélisation porte un nom.

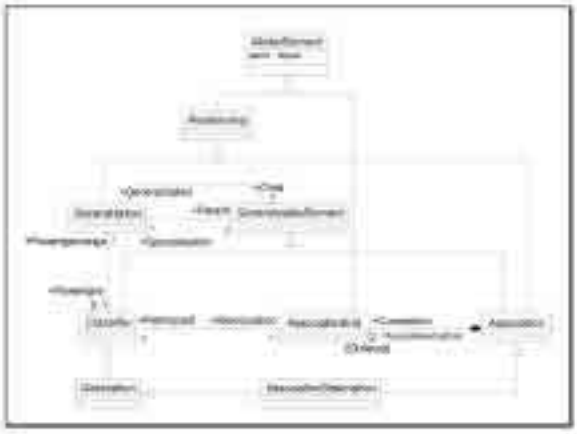

Figure 3. *Relationships core package.*

– Namespace : c'est une partie du modèle contenant un ensemble de *ModelElement*. Le nom d'un *ModelElement* est unique dans un *Namespace*. *Namespace* est une classe abstraite. Une spécialisation concrète de *Namespace* doit contenir des contraintes supplémentaires sur le type d'éléments qu'elle peut

contenir. Par exemple une instance de *Description* ne peut pas contenir une instance de *ComponentRepository*.

– Relationship : c'est une classe abstraite qui désigne un lien entre des instances de *ModelElement*. Une *Relationship* est spécifique à un *Namespace*, ce qui servira par la suite à définir des relations qui ont une portée limitée à une *Description* et ses Spécialisations. Par exemple, une *Relationship* définie sur les rubriques (*Item*) d'une *Description* est locale à toutes les instances de la description et elle est héritée par ses sous-descriptions (spécialisations).

– Constraint : c'est une contrainte ou une condition sémantique qui doit être vérifiée par les *ModelElement* (du même *Namespace*) pour que le modèle soit valide. La contrainte est exprimée sous la forme d'une expression booléenne.

– GeneralizableElement : représente une entité du modèle qui peut participer à une relation de généralisation (*Generalization*). D'après le méta-modèle UML un *GeneralizableElement* peut être la généralisation d'un autre *GeneralizableElement*. *GeneralizableElement* est une méta-classe abstraite.

– Classifier : dans le méta-modèle UML, une méta-classe *Classifier* déclare une collection de *Feature* comme les rubriques (*Item*). Elle a un nom qui est unique dans le *Namespace* qui l'encapsule. *Classifier* est une méta-classe abstraite qui est une spécialisation de *GeneralizableElement* et *Namespace*. Un classifier membre d'une relation de généralisation hérite des *Feature*(s) du classifier qu'il spécialise.

– Feature : représente un attribut encapsulé dans un *Classifier*. Dans le méta-modèle, une méta-classe *Feature* est spécialisée en *StructuralFeature*. Feature est une méta-classe abstraite. Dans l'état actuel du méta-modèle *M-Sigma* la méta-classe *Feature* ne joue pas un rôle important car elle se spécialise à son tour en une autre meta-classe abstraite *StructuralFeature*, mais elle permet d'avoir un méta-modèle ouvert et évolutif. Si on décide d'ajouter un autre type d'attribut, il suffira de spécialiser la méta-classe *Feature* en une métaclasse *XXXFeature*.

– StructuralFeature : c'est une spécialisation de la méta-classe Feature. C'est une méta-classe abstraite qui déclare une propriété structurelle (exemple *Item*).

– DescriptionFeature : c'est une méta-classe abstraite qui déclare une propriété descriptive (exemple *Property*).

– Property : c'est une spécialisation concrète de la méta-classe *DescriptionFeature*. Elle déclare une propriété descriptive associée à une des spécialisations concrètes de la méta-classe *Classifier*.

– Item : c'est une spécialisation concrète de la méta-classe *StructuralFeature*. Elle déclare une propriété structurelle associée à une des spécialisations concrètes de la méta-classe *Classifier*.

– Description : une description est composée d'un ensemble de rubriques (*Item*) et d'autres sous descriptions (*Description*). Une description peut être une spécialisation d'une autre description plus générale. Au sein d'une description on peut définir des contraintes (*Constraint*) sur les rubriques (*Item*) et les propriétés (*Property*). Dans une *Description*, on peut définir des relations (*Relationship*) entre les rubriques et entre les sous descriptions.

– ComponentsRepository : un SABC doit pouvoir intégrer plusieurs bases de composants. Chaque instance de cette méta-classe représente une base de composants indépendante qui a ses propres Descriptions de composants, ses propres relations entre descriptions et ses propres relations entre relations.

– Component : les instances de cette méta-classe sont des références sur les SABC. Une référence contient des informations sur l'emplacement physique d'où on peut extraire le composant.

– Artifact : un *Artifact* peut être un fichier ou une URL qui contient une portion d'un composant. Un *Component* est composé d'un ou plusieurs *Artifact*.

– DataType : représente un type de donnée pouvant être pris par une propriété (*Property*) ou une rubrique (*Item*). Un type de rubrique peut être du type *Primitive* (exp. entier, chaîne de caractères,…), ou *Enumeration* (exp. une énumération booléenne {« vrai », « faux »}), ou *Advanced* (exp. Diagramme UML)

– Generalization : cette méta-classe représente une relation dirigée de généralisation entre deux *GeneralizableElement* et le type du fils (le *Classifier* résultant de la spécialisation). C'est une spécialisation de la méta-classe *Relationship*.

– Association : une association définit une relation sémantique entre des *Classifier(s)* comme des *Description(s)*. Une instance de *Association* possède au moins deux instances de *AssociationEnd*. Un seul *Classifier* peut être connecté à plus d'un *AssociationEnd* dans une *Association*.

– AssociationEnd : c'est une extrémité d'une relation d'Association. *AssociationEnd* spécifie la relation entre une *Association* et un *Classifier*. Elle indique les *Classifier(s)* compatibles avec le bout de *l'Association*. Les *AssociationEnd*(s) d'une *Association* sont ordonnées.

– DescriptionAssociation : c'est une spécialisation des deux méta-classes *Association* et *Description*. *DescriptionAssociation* permet de décrire la sémantique d'une association à l'aide de rubriques. Il est aussi possible de définir des spécialisations et des compositions d'*AssociationDescription*(s) comme pour les descriptions.

Le méta-modèle que nous proposons a pour objectif l'aspect description de modèles de composants et de connexion entre composants. Nous avons enrichi la notion d'association en introduisant la méta-classe *AssociationDescription* qui étend le méta-

modèle UML en permettant la composition et la spécialisation d'associations et la définition d'association entre association. Nous avons aussi introduit des types de données avancées de rubriques comme les diagrammes UML qui seront utilisées pour une meilleure description des composants.

Nous présentons dans la section suivante une instance du méta-modèle proposé.

3. Le modèle C-Sigma : une instance du méta-modèle M-Sigma.

Nous présentons dans cette section un exemple d'instanciation partielle du méta-modèle M-Sigma limité aux notions de Formalisme et modèles de composants. Notre exemple illustre comment un SABC est capable de gérer une collection de composants hétérogènes (modèle de composants multiples et niveaux d'abstraction multiples). La description d'un composant (*DescriptionDeComposant*) est composée de deux sous descriptions : *FormalismeDeComposant*, *ModèleDeComposants* (cf. Figure 4).

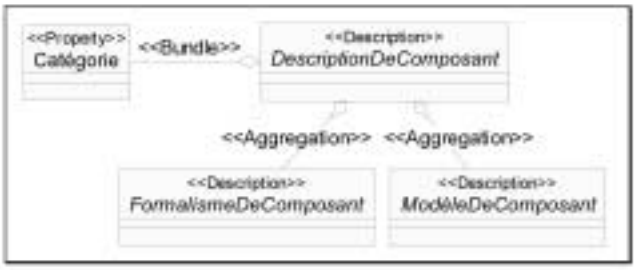

Figure 4 : *Une description de composant dans C-Sigma.*

Les informations contenues dans une description de composants sont divisées en deux grandes parties : les informations d'un composant dépendantes du modèle de composants incluse dans la sous description *ModèleDeComposant* et les informations générales d'un composant (indépendante du modèle de composants) incluse dans la sous description *FormalismeDeComposants*. Ainsi lors de la recherche d'un composant on peut classer les composants selon leur modèle de composants ou leur formalisme de descriptions de composant. Nous pouvons alors exprimer des requêtes spécifiques aux modèles ou au formalismes de composants.

La spécialisation *Formalisme C-Sigma* de la description *Formalisme C-Sigma* définit quatre rubriques principales (cf. Figure 5):

– Nom : permet l'identification du composant dans une base de composants,

– Description : décrit l'utilité du composant et le problème qu'il résout,

– Forces : décrit les avantages apportés par l'utilisation du composant,

– Faiblesses : liste les inconvénients ou les problèmes constatés lors de l'utilisation du composant.

De plus, Le *Formalisme C-Sigma* définit entre autres la relation *Processus de réutilisation*. Cette relation permet d'associer un patron processus qui offre une démarche de réutilisation du composant.

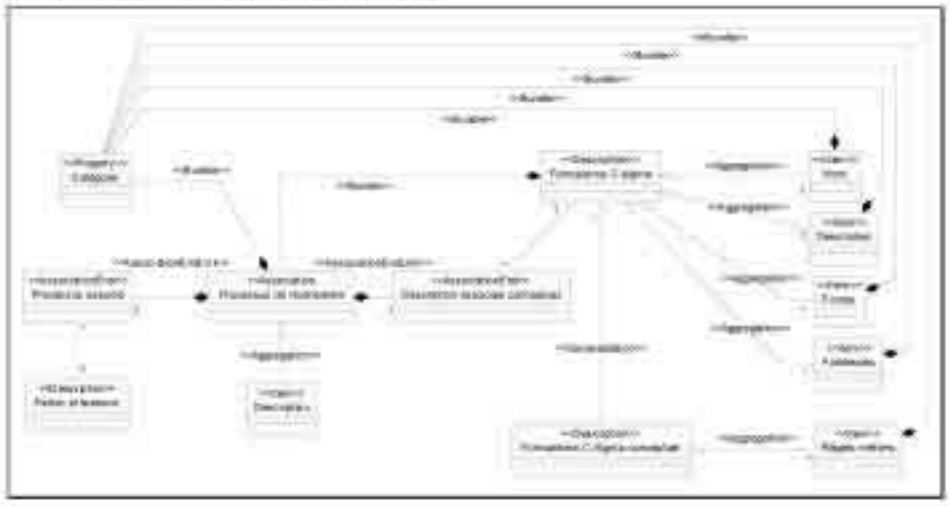

Figure 5 : *Le formalisme de composant C-Sigma*

La description *Formalisme C-Sigma conceptuel* introduit la rubrique *Règles métiers* et hérite de la description *Formalisme C-Sigma* la relation *Processus de réutilisation* et des quatre rubriques *Nom, Description, Forces, Faiblesses*.

Nous introduisons la notion de *Catégorie* qui désigne une propriété attachée aux descriptions, aux relations et aux rubriques du modèle *C-Sigma*. Cette notion de *Catégorie* permet une classification des éléments de modélisation. Dans notre modèle d'organisation nous définissons trois catégories : *réutilisation* (ex. *Processus de réutilisation, Forces, Faiblesses*), *solution* (ex. *Modèle de composants*) et *documentation* (ex. *FormalismeDeComposant, Description, Nom*).

Dans cette troisième section nous nous sommes limités à une sous-partie d'un modèle d'une base descriptive de composants (*C-Sigma*) et nous l'avons instanciée dans le méta-modèle présenté dans la section 2. Cette illustration met en évidence la souplesse et le pouvoir d'expression de notre méta-modèle.

4. Conclusion et perspective.

Dans la deuxième partie de cet article nous avons présenté une partie du modèle *C-Sigma*. Le modèle *C-Sigma* organise la base descriptive de composants selon des niveaux d'abstraction multiples et en séparant les informations spécifiques aux modèles de composants des informations de réutilisation et de documentation. Le modèle *C-Sigma* est une instance du méta-modèle *M-Sigma* présenté dans la première partie de cet article. Le méta-modèle *M-Sigma* permet d'instancier des modèles de bases descriptives de composants. Grâce à la notion de *Item* et de *Description M-Sigma* exploite et étend la technique de classification par facettes [6]. Ainsi, *M-Sigma* permet de structurer et de classer les composants selon leurs descriptions. D'un autre cotés les relations définies dans *M-Sigma* permettent la navigation, entre les différents concepts de la base descriptive de composants. *M-Sigma* définit les relations de Generalization, Aggregation, et AssociationDescription. A la différence de UML *M-Sigma* permet de définir des associations inter association. L'approche par navigation a certaines limites car le résultat final de la recherche dépend du point de départ de la navigation. La prochaine étape de notre travail sera d'enrichir le méta-modèle pour intégrer le concept de technique de recherche de composants. En d'autres termes nous allons étendre notre approche aux différentes techniques de recherche (structurelles et comportementales)

5. Bibliographie.

[1]. Object Management Group (OMG): Unified Modeling Language Specification 1.5, (2003)

[2] S. Atkinson and R. Duke. Behavioural retrieval from class libraries. *Australian Computer Science Communications*, 17(1), pages 13-20, January 1995.

[3] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns – Elements of Reusable Oriented Software*. Addison-Wesley, 1995.

[4] Adele Goldberg, *SMALLTALK-80: the interactive programming environment*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1984

[5] A. M. Zaremski and J. M. Wing. Signature matching: A key to reuse. Technical Report CMU-CS-93-151, Carnegie Mellon University, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, May 1993.

[6] Z. Zhang, L. Svensson, U. Snis, C. Srensen, H. Fgerlind, T. Lindroth, M. Magnusson, C. Stlund. Enhancing Component Reuse Using Search Techniques, *Proceedings of IRIS 23. Laboratorium for Interaction Technology*, University of Trollhttan Uddevalla, 2000.