Un analyseur sémantique pour MOF

Olivier Festor, Nizar Ben Youssef

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

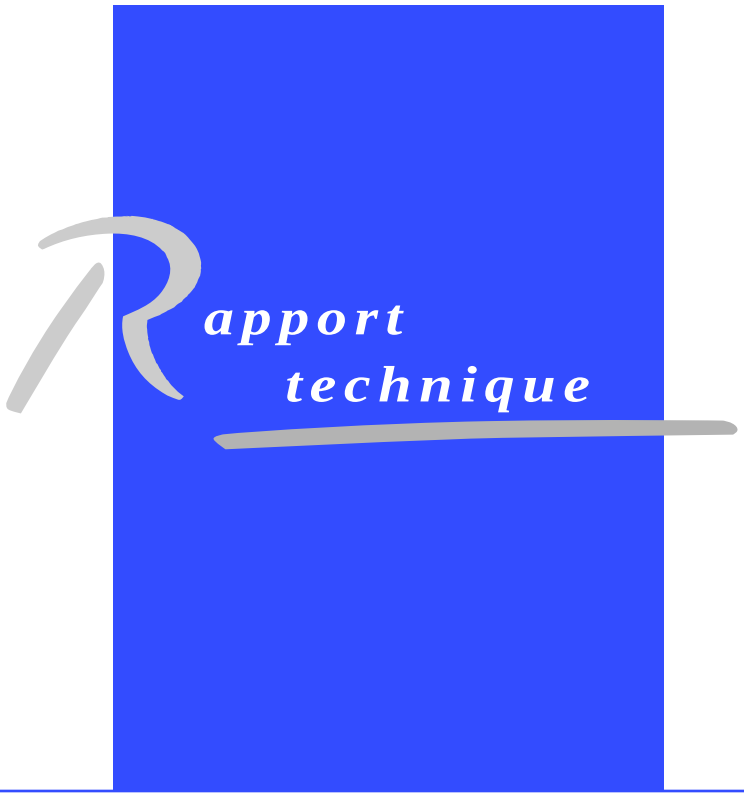
Un analyseur sémantique pour MOF

Olivier Festor, Nizar Ben Youssef

No 0233

26 juillet 1999

THÈME 1



ISSN 0249-0803

Un analyseur sémantique pour MOF

Olivier Festor, Nizar Ben Youssef

Thème 1 — Réseaux et systèmes
Projet RÉSÈDAS

Rapport technique n° 0233 — 26 juillet 1999 — 12 pages

Résumé : MODERES Java est un environnement de développement de modèles de l'information de gestion. Sa vocation est de fournir un ensemble d'outils et d'interfaces intégrées pour l'ensemble des langages de description des modèles de l'information utilisés en gestion de réseaux et de services.

Ce rapport présente l'un des composants de l'environnement à savoir l'analyseur sémantique des spécifications MOF (Managed Object Format). MOF est le langage de description d'interfaces de gestion utilisé dans l'approche CIM (Common Information Model) du DMTF (Distributed Management Task Force). Le rapport comporte une description détaillée des vérifications sémantiques à entreprendre sur ce type de spécifications. Il comporte également une présentation de l'implémentation d'un module de vérification basé sur ces contrôles dans l'environnement MODERES Java.

Mots-clé : MOF, Java, MODERES, CIM, WBEM

(Abstract: pto)

Ce travail a été partiellement réalisé dans le cadre de l'action ANTARES (Architectures et Nouvelles Technologies pour l'Administration des Réseaux et Service) du GIE DYADE

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : 03 83 59 30 30 - International : +33 3 3 83 59 30 30
Télécopie : 03 83 27 83 19 - International : +33 3 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ

A MOF Semantics Checker

Abstract: MODERES Java is a software environment for the development of management information models. It aims to provide a set of integrated tools and interfaces for all information model notations used in the domain of mangement information modeling.

This report presents one component of the environment, i.e. the semantics checker for MOF (Managed Object Format) specifications used in the CIM (Common Information Model) approach standardized by the DMTF (Distributed Management Task Force). The report contains a detailled description of performed verifications as well as a complete implementation and usage guide for the semantics checker as implemented within the environment.

Key-words: MOF, Java, MODERES, CIM, WBEM

1 Introduction

MODERES Java [3][2], développé dans le projet RESEDAS, permet la saisie, l'analyse et la manipulation des modèles de l'information de gestion quelque soit le formalisme standard utilisé pour les décrire (SNMP, GDMO/ASN.1, MOF, ...). Avec la montée en puissance de l'approche WBEM (Web-Based Enterprise Mangement), il a été récemment étendu pour permettre la manipulation du modèle de l'information CIM (Common Information Model) [1], i.e le traitement des spécifications MOF (Managed Object Format) sous-jacentes [4]. Tout l'environnement est développé en Java, et facilite la manipulation de l'information de gestion, en offrant, outre des outils complets, un ensemble d'interfaces de programmation permettant aux utilisateurs et développeurs d'y insérer leurs propres outils de gestion.

Ce rapport présente l'analyse sémantique réalisée par l'analyseur sémantique MOF intégré dans l'environnement. Il présente aussi l'architecture de cet analyseur ainsi qu'un guide d'utilisation complet.

Le rapport est organisé comme suit. La section 2 donne une caractérisation d'une erreur sémantique MOF. La section 3 décrit la phase de pré-analyse visant à distinguer les classes simples des associations. La section 4 enumère toutes les erreurs détectées par l'analyseur sémantique. La section 5 donne un guide d'utilisation de l'analyseur sémantique au sein de l'environnement MODERES.

2 Les erreurs sémantiques dans MOF

Dans cette section nous donnons une vue globale sur les erreurs qui sont détectées par l'analyseur sémantique MOF de l'environnement MODERES [4]. Ces erreurs sont déterminées pour chacun des élements du méta-modèle défini dans les spécifications de la version 2.0 du modèle CIM :

- le qualifieur ,
- la classe ,
- la propriété ,
- la méthode ,
- l'association ,
- la référence ,
- l'indication ,
- l'instance.

L'analyse sémantique de ces éléments effectuée dans MODERES est réalisée sur deux niveaux différents. D'une part une vérification de l'existance de toute définition référencée dans la spécification d'un élément, i.e que toute référence à une définition correspond à un élément chargé par l'analyseur syntaxique MOF de MODERES est entreprise. Comme la sémantique d'un élément dans CIM repose essentiellement sur les qualifieurs qui lui sont attribués l'analyseur sémantique MOF de MODERES edoit d'autre part effectuer des vérifications suppémentaires sur les qualifieurs utilisés dans le modèle.

En plus de la détection d'erreurs sur les réferences, l'analyseur sémantique permet de compléter l'analyse syntaxique MOF, en associant aux spécifications MOF, des références vers les objets Java qui les représentent.

3 Distinction entre associations, indications et classes

Le compilateur de spécifications MOF de l'environnement MODERES, ne permet pas de distinguer une classe simple d'une classe association, ni d'une indication. En effet, dans la version 2.2 des spécifications MOF [1], une classe est considérée comme association si elle vérifie l'une des conditions suivantes :

1. Elle instancie le méta-qualifieur `Association` avec la valeur **"TRUE"** ;
2. Elle contient au moins deux attributs références ;
3. Elle hérite d'une association (vérifiant les conditions 1 et 2).

La condition (3) donnée dans les spécifications MOF[1], implique que la distinction entre classe simple et classe association nécessite un parcours de l'arbre d'héritage des classes. En conséquence, nous avons opté pour l'inclusion de cette fonctionnalité dans l'analyseur sémantique MOF. Il en est de même pour la fonctionnalité

1. Explicitée pour la définition d'une association mais pas pour la définition d'une indication dans la version actuelle!

similaire de distinction entre classe simple et classe indication (spécifiée par une unique condition, similaire à la condition (1) avec le méta-qualifieur `Indication`).

L'analyseur sémantique fait le parcours des spécifications de classes pour en extraire les associations en appliquant les critères suivants :

- une classe est considérée comme association si elle instancie le qualifieur `Association`, avec la valeur `"TRUE"` en premier dans sa liste de qualifieurs;
- toute classe héritant d'une association est une association ;
- pour toute super-association (racine dans un arbre d'héritage d'associations), l'analyseur vérifie qu'il existe au moins deux attributs de types références.

  Si une erreur est détectée, un objet d'erreur est généré avec les champs suivants :
  - type d'erreur : `ASSOCIATION_LOW_ARITY` ;
  - message : `"Association defined with arity lower than 2."`.
  - une référence vers la super-association.

  L'erreur d'arité faible est propagée pour toute association de la même branche d'héritage, i.e que la même erreur sera générée pour toute classe héritant de cette super-association.

Parallèlement, l'analyseur permet d'extraire les indications de la liste des classes, en appliquant l'unique critère suivant :

- une classe est considérée comme indication si elle instancie le qualifieur `Indication` avec la valeur `"TRUE"` en premier dans sa liste de qualifieurs.

4 Les clauses sémantiques vérifiées

4.1 Analyse des qualifieurs

L'analyse des qualifieurs se fait exclusivement sur les instances de qualifieurs (i.e les qualifieurs associés aux définitions des différents élements de CIM), les définitions de qualifieurs étant supposées sémantiquement correctes si elles sont syntaxiquement correctes [2].

Sous ces conditions, nous définissons deux niveaux d'analyse sémantique :

- vérification de la "bonne formation" : Elle consiste à vérifier que l'instanciation du qualifieur est correcte vis à vis de sa définition (donnée dans le fichier MOF) ;
- vérification de la "validité" : Elle vérifie les instanciations des qualifieurs standards (donnés dans les spécifications de la version 2.2 de CIM) i.e. que toute instanciation de ces qualifieurs est conforme à la sémantique qui leur est donnée par le DMTF.

4.1.1 Critères de bonne formation des qualifieurs

1. *Tout qualifieur utilisé dans une spécification, doit avoir sa définition accessible dans cette même spécification.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `QUALIFIER_DEFINITION_ERROR` ;
   - message : `"Qualifier definition not found."` ;
   - une référence vers l'instance du qualifieur, ainsi qu'une référence vers l'élément instanciant le qualifieur.

   De plus, l'analyseur n'effectuera aucune autre vérification sur ce qualifieur.
2. *Toute valeur prise par une instance de qualifieur, doit être du même type de données que celui spécifié dans la définition du qualifieur.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `QUALIFIER_TYPE_ERROR` ;
   - message : `"Type error for qualifier initializer."` ;
   - une référence vers l'instance du qualifieur, ainsi qu'une référence vers l'élément instanciant le qualifieur.

2. En supposant que personne ne s'amusera à donner une nouvelle sémantique aux qualifieurs standards de CIM 2.2 !

De plus, en cas d'absence de valeur à l'instanciation des qualifieurs de type booléen[3], l'analyseur sémantique complète le résultat de l'analyseur syntaxique en leur associant la valeur booléenne "`TRUE`".

3. *Toute instance de qualifieur, doit respecter les contraintes de portée (Scope) spécifiées dans la définition du qualifieur, i.e. que l'instance du qualifieur doit être associée à un élément dont le type est donné dans la liste de portée du qualifieur.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `QUALIFIER_SCOPE_ERROR` ;
   - message : "`Qualifier instance out of scope.`" ;
   - une référence vers l'instance du qualifieur, ainsi qu'une référence vers l'élément instanciant le qualifieur.
4. *Toute instance de qualifieur, doit respecter les contraintes d'héritage (Flavor) spécifiées dans la définition du qualifieur.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `QUALIFIER_FLAVOR_ERROR` ;
   - message : "`Qualifier instance out of flavor.`" ;
   - une référence vers l'instance du qualifieur, ainsi qu'une référence vers l'élément instanciant le qualifieur.

4.1.2 Critères de validité des qualifieurs

Les critères de validité décrits ci-dessous sont définis à partir des spécifications des qualifieurs standards donnés dans la version 2.2 de CIM.

1. ABSTRACT : *Aucune instance ne doit être déclarée pour une classe définie avec le qualifieur* `Abstract`.

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `ABSTRACT_CLASS_ERROR` ;
   - message : "`Abstract class can not be instanciated.`" ;
   - une référence vers l'instance de la classe abstraite.
2. KEY : *Aucune valeur par défaut ne doit être attribuée aux propriétés clefs d'une classe.*

   Si une erreur est détectée, l'objet d'erreur est généré avec les champs suivants :
   - type de l'erreur : `KEY_DEFVAL_ERROR` ;
   - message : "`Key property may not have a default value.`" ;
   - une référence vers la propriété clef définie avec une valeur par défaut.
3. OVERRIDE : Une instance du qualifieur `Override` permet d'indiquer une surcharge d'un élément de spécification de classe (propriété, méthode ou référence). Il est défini par une chaîne de caractères de la forme [<class-name>.]<feature-name>, où <class-name> indique la super-classe définissant l'élément surchargé <feature-name>.

   La valeur prise par une instance du qualifieur `Override` *associée à un élément donné, doit correspondre à une définition accessible dans l'arbre d'héritage de la classe contenant l'élément.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `OVERRIDED_FEATURE_UNDEFINED` ;
   - message : "`Definition for overrided feature is missing.`" ;
   - une référence vers l'instance du qualifieur, ainsi qu'une référence vers l'élément instanciant le qualifieur.
4. PROPAGATED : Une instance du qualifieur `Propagated` permet d'indiquer qu'une propriété est propagée de la classe forte vers la classe faible dans une association "faible" (*weak association*). Il est défini par une chaîne de caractères de la forme [<class-name>.]<property-name>, où <class-name> indique la classe forte de l'association, et <property-name>la propriété propagée.

   La valeur prise par une instance du qualifieur `Propagated` *associé à une propriété donnée, doit correspondre à une définition accessible dans l'arbre d'héritage de la classe référencée.*

   Si une erreur est détectée (la classe ou la propriété référencées ne sont pas accessibles), l'objet d'erreur est généré avec les champs suivants :
   - type de l'erreur : `PROPAGATED_PROPERTY_UNDEFINED` ;

3. Dans les spécifications CIM 2.2, la valeur VRAI leur est implicitement associée

- message : "`Definition for propagated property is missing.`";
- une référence vers l'instance du qualifieur, ainsi qu'une référence vers l'élément instanciant le qualifieur.

5. TERMINAL : *Une classe définie avec le qualifieur ne doit pas être sous-classée.*
   Si une erreur est détectée, un objet d'erreur est généré avec les champs suivants :
   - type de l'erreur : `TERMINAL_CLASS_ERROR` ;
   - message : "`Terminal class can not have subclasses.`";
   - une référence vers la sous-classe de la classe terminale.
6. WEAK : Une instance du qualifieur `Weak` attachée à un attribut référence, permet de désigner la classe "faible" dans une association, i.e celle qui héritera les propriétés clefs des autres classes de l'association (voir qualifieur `Propagated`). L'analyseur vérifie alors chacun des trois points suivants :
   Aucune association ne peut avoir plus d'un attribut référence instanciant le qualifieur `Weak`.
   Si une erreur est détectée, un objet d'erreur est généré avec les champs suivants :
   - type de l'erreur : `WEAK_ASSOCIATION_ERROR` ;
   - message : "`Weak association is defined with more than one weak reference.`";
   - une référence vers l'association erronée.

   Toute classe "forte" dans une "association faible", doit avoir au moins un attribut clef.
   Si une erreur est détectée, un objet d'erreur est généré avec les champs suivants :
   - type de l'erreur : `WEAK_ASSOCIATION_ERROR` ;
   - message : "`Class in weak association may have at least one key property.`";
   - une référence vers l'association erronée et une référence vers l'attribut référence désignant la classe "forte" ne possédant pas de propriétés clefs.

   Tous les attributs clefs des classes "fortes" dans une association faible, doivent être propagés vers la classe "faible" de l'association. (i.e. doivent figurer dans la liste des attributs clef de la classe faible)
   Si une erreur est détectée, un objet d'erreur est généré avec les champs suivants :
   - type de l'erreur : `WEAK_ASSOCIATION_ERROR` ;
   - message : "`Key property <property-name> in class <class-name> may be propagated to weak class.`";
   - une référence vers l'association erronée et une référence vers l'attribut référence désignant la classe "faible".

4.2 Analyse d'une liste de qualifieurs

A toute définition CIM (classe, propriété, méthode, référence,...), on peut associer une liste de qualifieurs. L'analyseur sémantique vérifie alors pour chaque définition CIM, cette liste de qualifieurs selon les critères suivants :

1. *Aucun qualifieur ne doit être dupliqué dans la liste des qualifieurs.*
   Si une erreur de duplication est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `QUALIFIER_LIST_ERROR` ;
   - message : "`Erreur de duplication de qualifieur.`";
   - une référence vers l'élément avec une telle liste.
2. *Tout qualifieur de la liste doit vérifier les critères de "bonne formation".*
3. *Tout qualifieur de la liste doit vérifier les critères de "validité".*

4.3 Analyse des définitions de classes

La classe CIM est définie par un nom, un nom d'alias optionnel, un nom de super-classe optionnel, une liste optionnelle de qualifieurs, une liste optionnelle de propriétés et une liste optionnelle de méthodes.

1. *Pour toute définition de classe, le lien d'héritage doit être correct, i.e. la définition de la super-classe doit être accessible.*
   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `CLASS_INHERITANCE_ERROR` ;

- message : "`Unresolved super-class reference.`";
- une référence vers la définition de classe erronée.

2. *Pour toute définition de classe, l'analyseur vérifie que la liste des qualifieurs est correcte (voir 4.2).*
3. *Pour toute définition de classe, la liste des propriétés doit être correcte i.e. que chaque propriété de la liste doit être correctement définie. De plus aucune propriété ne doit être dupliquée dans la liste.*

   Si une erreur de duplication de propriétés est détectée, l'object d'erreur associé comporte les champs suivants :
   - type de l'erreur : `PROPERTY_LIST_ERROR` ;
   - message : "`Property duplicated in class definition.`";
   - une référence vers la définition de classe erronée ainsi qu'une référence vers la dernière définition de la propriété dupliquée.
4. *Pour toute définition de classe, la liste des méthodes doit être correcte i.e que chaque méthode de la liste doit être correctement définie. De plus aucune méthode ne doit être dupliquée dans la liste.*

   Si une erreur de duplication de méthodes est détectée, l'object d'erreur associé comporte les champs suivants :
   - type de l'erreur : `METHOD_LIST_ERROR` ;
   - message : "`Method duplicated in class definition.`";
   - une référence vers la définition de classe erronée ainsi qu'une référence vers la dernière définition de la méthode dupliquée.
5. *Deux classes dans un même namespace ne peuvent avoir le même nom.*

   Si une erreur de duplication de classes est détectée, l'object d'erreur associé comporte les champs suivants :
   - type de l'erreur : `CLASS_DUCPLICATION_ERROR` ;
   - message : "`class name conflict.`";
   - une référence vers la définition de classe erronée ainsi qu'une référence vers la dernière définition de la classe dupliquée.

4.4 Analyse des définitions de propriétés

La propriété CIM est définie par un nom, une liste optionnelle de qualifieurs, un type et une valeur par défaut optionnelle.

1. *Pour toute définition de propriété, la liste des qualifieurs qui peut lui être associée, doit être correcte.*
2. *Aucune définition de propriété, ne doit surcharger une propriété déjà définie dans l'arbre d'héritage de la classe déclarante, sauf si la surcharge est explicitement définie par le qualifieur Override.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `PROPERTY_OVERRIDE_ERROR` ;
   - message : "`Overriding property in super-class : <super-class-name>.`";
   - une référence vers la classe, et une référence vers la propriété erronée.

   Cette vérification n'est effectuée que si le qualifieur `Override` est absent de la liste de qualifieurs de la propriété.

4.5 Analyse des définitions de méthodes

La méthode CIM est définie par un nom, une liste optionnelle de qualifieurs, un type de retour et une liste de paramètres.

1. *Pour toute définition de méthode, la liste des qualifieurs qui peu lui être associée, doit être correcte.*
2. *Aucune définition de méthode, ne doit surcharger une méthode déjà définie dans l'arbre d'héritage de la classe déclarante, sauf si la surcharge est explicitement définie par le qualifieur Override.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `METHOD_OVERRIDE_ERROR` ;
   - message : "`Overriding method in super-class : <super-class-name>.`";
   - une référence vers la classe, et une référence vers la méthode erronée.

   Cette vérification n'est effectuée que si le qualifieur `Override` est absent de la liste de qualifieurs de la méthode.

4.6 Analyse des définitions d'associations

Une association CIM est une sous-classe d'une définition de classe CIM. Elle comprend en plus une liste de références vers d'autres classes ou instances CIM.

1. *Pour toute définition d'association, la sémantique d'une classe simple doit être vérifiée, i.e que les vérifications de la super-classe, des listes des qualifieurs, propriétés et méthodes, doivent être correctes.*
2. *Aucune définition d'association, ne doit surcharger l'arité de sa super-association, i.e qu'elle ne doit pas déclarer de nouveaux attributs de type référence (dans MOF la surcharge de références est autorisée, mais pas l'ajoût).*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type d'erreur : `ASSOCIATION_OVERRIDE_ARITY` ;
   - message : "`Association changes arity of super-association: <super-association-name>.`" ;
   - une référence vers l'association erronée.

4.7 Analyse des définitions de références

Une référence CIM est définie par un nom, le nom de la classe qu'elle référence et une liste optionnelle de qualifieurs.

1. *Pour toute définition de référence, la liste des qualifieurs qui peut lui être associée, doit être correcte.*
2. *Aucune définition de référence, ne doit surcharger une méthode déjà définie dans l'arbre d'héritage de la classe déclarante, sauf si la surcharge est explicitement définie par le qualifieur Override.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type de l'erreur : `REFERENCE_OVERRIDE_ERROR` ;
   - message : "`Overriding reference in super-association: <super-association-name>.`" ;
   - une référence vers l'association et une référence vers la référence erronée.
3. *Pour toute définition de référence, la définition de la classe référencée doit être accessible.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type d'erreur : `REFERENCE_CLASS_ERROR` ;
   - message : "`Referenced class definition not found.`" ;
   - une référence vers l'association, et une référence vers la référence erronée.

4.8 Analyse des définitions des indications

Une indication CIM est une sous-classe d'une définition de classe.

1. *Pour toute définition d'indication, la sémantique d'une classe simple doit être vérifiée, i.e. que les vérifications de la super-classe, des listes des qualifieurs propriétés et méthodes, doivent être correctes.*

4.9 Analyse des définitions d'instances

Une instance CIM est définie par un nom de classe, une liste optionnelle de qualifieurs et une liste non vide de valeurs de propriétés.

1. *Pour toute définition d'instance, la liste des qualifieurs qui peut lui être associée, doit être correcte.*
2. *Pour toute définition d'instance, la définition de la classe de l'instance doit être accessible.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type d'erreur : `INSTANCE_CLASS_ERROR` ;
   - message : "`Unresolved reference to class definition.`" ;
   - une référence vers la définition de l'instance erronée.

   De plus l'analyseur ne procédera à aucune autre vérification sur l'instance après détection d'une telle erreur.
3. *Pour toute définition d'instance, la définition de la classe de l'instance, doit avoir au moins une propriété clef (ou en hérite).*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :
   - type d'erreur : `INSTANCE_CLASS_KEYLESS` ;

- message : "`Class is Keyless for instance definition.`";
- une référence vers la définition de l'instance erronée.

4. *Pour toute définition d'instance, toutes les propriétés clefs définies dans la classe de l'instance (ou héritées), doivent être initialisées à des valeurs non nulles.*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :

   - type d'erreur : `INSTANCE_KEY_INITIALIZATION` ;
   - message : "`Key property <property-name>defined in class <class-name>not initialized in instance.`", où <property-name>désigne la propriété clef et <class-name>désigne la classe définissant cette clef ;
   - une référence vers la définition de l'instance erronée.

   Cette erreur est générée pour chaque propriété clef non instanciée.

5. *Pour toute définition d'instance, toutes les propriétés initialisées, doivent avoir leur définition accessible dans la définition de la classe de l'instance (ou l'une de ses super-classes).*

   Si une erreur est détectée, l'objet d'erreur associé comporte les champs suivants :

   - type d'erreur : `INSTANCE_PROPERTY_ERROR` ;
   - message : "`Property undefined for instance initialization.`";
   - une référence vers la définition de l'instance et une référence vers la propriété erronée.

6. *Pour toute définition d'instance, toute initialisation de propriété doit être du même type que celui donné dans la définition de la propriété de la classe de l'instance.*

   Si une erreur est détectée, l'objet d'erreur associée comporte les champs suivants :

   - type d'erreur : `INSTANCE_PROPERTY_INITIALIZATION` ;
   - message : "`Property type initialization undefined in class <class-name>.`", où <class-name> désigne la classe définissant cette propriété ;
   - une référence vers la définition d'instance et une référence vers l'initialisation de propriété erronée.

5 Implémentation et guide d'utilisation

5.1 Implémentation

L'analyseur sémantique MOF est implémenté dans la classe `MOF_Specification` du *package* `mofrepository` de MODERES. Pour une description plus détaillée de l'environnement MODERES en Java nous renvoyons le lecteur à [3].

L'analyseur sémantique est accessible à travers l'invocation de la méthode `checkSemantics()` qui retourne une liste d'objets de la classe `MOF_SemanticError`. Si la liste est vide, aucune erreur n'est détectée. Dans le cas contraire, chaque élément contient une explication de l'erreur détectée.

Les objets de la classe `MOF_SemanticError` possédent les champs suivants :

- un entier représentant le code de l'erreur ;
- une chaîne de caractères donnant le détail de l'erreur ;
- une référence vers la classe MOF à l'origine de l'erreur ;
- une référence vers la propriété, méthode ou référence MOF à l'origine de l'erreur. Ce champs peut être nul, si l'erreur est spécifique à un autre type d'éléments MOF (classe, qualifieur,...) ;
- une référence vers l'instance de qualifieur à l'origine de l'erreur. Ce champs peut être nul, si l'erreur est spécifique à un autre type d'éléments MOF (classe, propriété,...).

Chaque objet de la classe `MOF_SemanticError`, offre une méthode utile pour l'affichage : `getMessage()` qui renvoie une chaîne de caratères contenant une description détaillée de l'erreur et du(des) composant(s) à son origine.

Un exemple d'utilisation de l'analyseur sémantique est donné dans la figure 1. Il est tiré du programme donné dans le fichier `MODERES_BasicMOFParser.java` du répertoire d'exemples. Les lignes relatives à l'utilisation de l'analyseur sémantique sont écrites en gras.

```
package FR.loria.resedas.moderes.mofexamples;

import java.io.*;
import com.objectspace.jgl.*;
import FR.loria.resedas.moderes.mofrepository.*;
import FR.loria.resedas.moderes.moffrontend.*;

/** Basic single file parser
  * Loads a MOF file and generates an output if selected
  * @author Olivier Festor
  * @version 0.9
  */
 public class MODERES_BasicMOFParser {

   /** Main method of the parser
     * @param args list of arguments
     */
   public static void main(String args[]) {
     // Declares the parser to be used
     MODERESCoreMOFParser parser;
     // instanciates a specifications list to be filled
     MOF_Specification spec = new MOF_Specification();

     try { // Creates the parser and opens input file
       parser = new MODERESCoreMOFParser(new java.io.FileInputStream(args[0]));
     } catch (java.io.FileNotFoundException e) {
       System.out.println("MOF Parser Version 0.9:  File " + args[0] + " not found.");
       return;
     }

     try {
       // Parsing starts here
       parser.ParseMOFFile(spec);
       // if no error occured, the parser says it!
       System.out.println("MODERES MOF Parser Version 0.9:  Java program parsed successfully.");

        // semantics checks starts here
        DList errors = spec.checkSemantics();
        if ( errors.isEmpty() ) {
          System.out.println("MOF Semantics: Specifications checked successfully.");
        }
        else {
          DListIterator iterator = errors.begin();
          while ( iterator.hasMoreElements() ) {
          MOF_SemanticError error = iterator.nextElement();
          System.out.println("Semantic Error: "+error.getMessage());
        }
     } catch (ParseException e) {
       // if a parsing error occured, the system issues the following message
       System.out.println("MOF Parser Version 0.9:  exit due to parse error.");
       System.out.println(e.getMessage());
     }
   }
 }
```

FIG. 1 – *Exemple d'utilisation de l'analyseur sémantique*

5.2 Utilisation dans l'environnement graphique de MODERES

Dans cette section, nous présentons l'utilisation de l'analyseur sémantique MOF à partir de l'environnement graphique de MODERES.

Après le chargement d'un ou plusieurs fichiers MOF dans l'environnement (menu **Repositories**), l'accès à l'analyseur sémentique s'effectue en cherchant le sous-menu **Check WBEM MOF Repository** dans le menu **Applications** (voir figure 2).

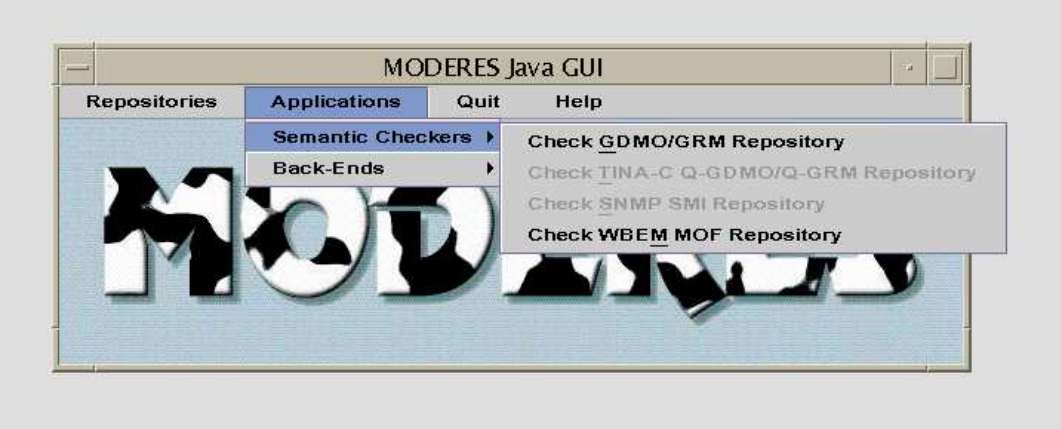


FIG. 2 – *Accès à l'analyseur sémentique depuis l'environnement graphique MODERES*

Une fois l'analyse terminée, une fenêtre sera ouvetre par l'environnement pour donner le résultat des vérifications :

– Aucune erreur n'a été détectée : figure 3.

FIG. 3 – *Analyse sans erreurs détectées*

– Une ou plusieurs erreurs ont été détectées : figure 4. Le descriptif de chaque erreur est alors affiché dans la fenêtre d'erreurs. De plus devant chaque erreur, un bouton permet d'ouvrir une fenêtre de l'editeur graphique MOF[4] affichant la définition de la classe ou de l'instance à l'origine de l'erreur.

6 Conclusion

Dans ce rapport, nous avons présenté l'analyseur sémantique implémenté dans l'environnement MODERES Java. Cet analyseur est implanté dans la classe MOF_Specification du module moderes. Il permet d'effectuer une vérification sémantique complète sur les différents schémas MOF chargés dans la spécification, et renvoie

4. En phase finale de développement.

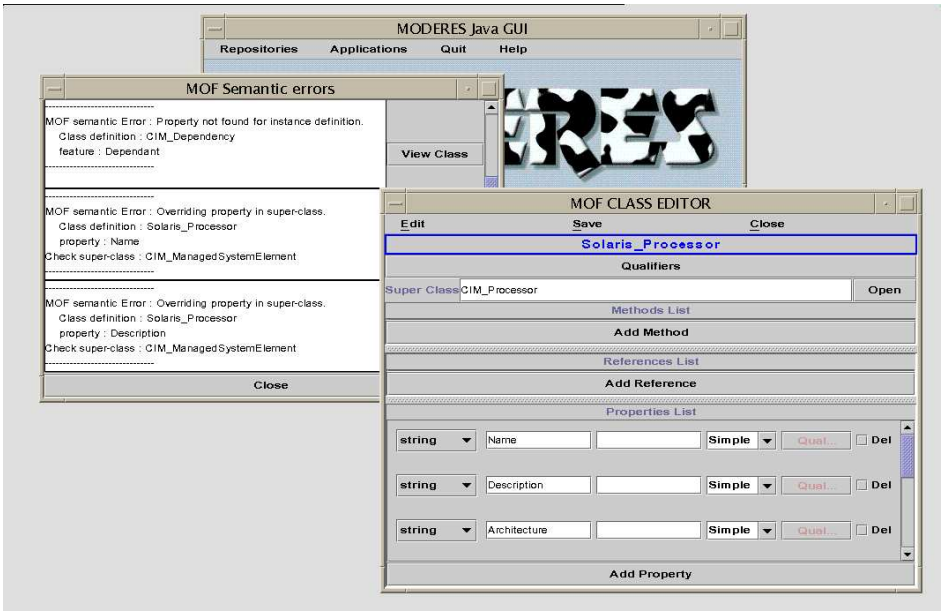


FIG. 4 – *Fenêtre d'erreurs détectées par l'analyse*

à l'utilisateur un rapport exploitable de l'erreur détectée. Dans sa dernière distribution, cet environnement est conforme à la version 2.2 de CIM, récemment actualisée par le DMTF.

L'analyse sémantique implantée dans cette version de l'environnement effectue les traitements sur la base des schémas. La derniè version de la spécification MOF encourage la réalisation de cette vérification au niveau des espaces de nommage. Cette facilité est en cours de réalisation dans notre environnement.

Références

[1] DMTF. Common Information Model (CIM) version 2.2. Technical report, Distributed Management Task Force, June 1999.

[2] O. Festor. The gdmo and grm modules semantic checker of the moderes java toolkit. Technical Report RT-0208, INRIA, July 1997.

[3] O. Festor. MODERES Java: Architecture and Core Packages. Technical Report RT-0205, INRIA, May 1997.

[4] O. Festor. The Managed Object Format specification parser of the MODERES Java Toolkit. Rapport technique RT-0218, INRIA, Février 1998.

Unit´e de recherche INRIA Lorraine, Technopˆole de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhˆone-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399