The Managed Object Format Specification Parser of the MODERES Java Toolkit

Olivier Festor

INRIA

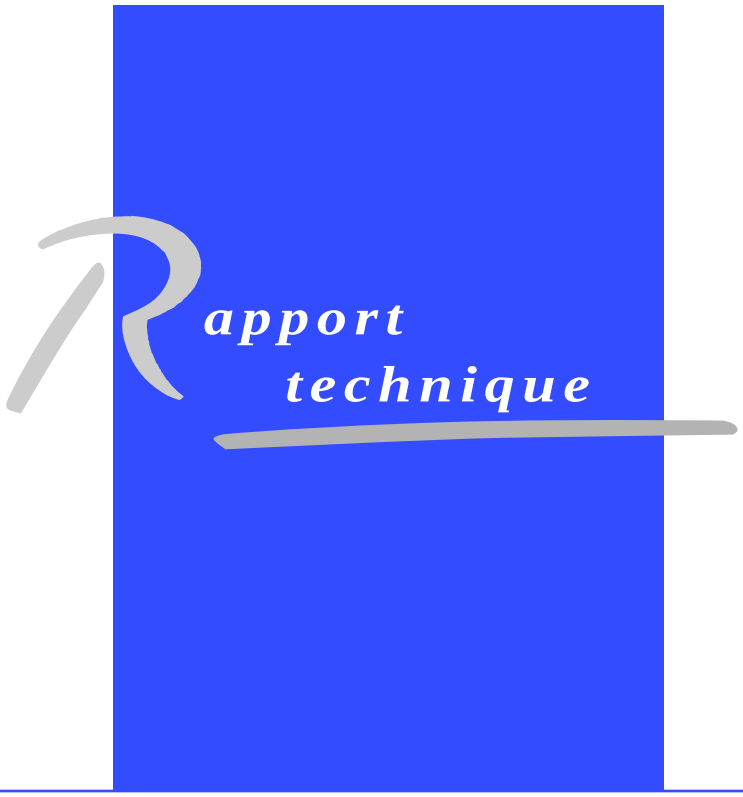
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

The Managed Object Format Specification Parser of the MODERES Java Toolkit

Olivier Festor

N ° 0218

THÈME 1



ISSN 0249-0803

The Managed Object Format Specification Parser of the MODERES Java Toolkit

Olivier Festor

Thème 1 — Réseaux et systèmes
Projet RESEDAS

Rapport technique n0218 — — 14 pages

Abstract: MODERES Java is an environment for the development of management information models. This environment aims at providing a set of tools for the manipulation of information models coming from different management frameworks. These tools include parsers, mapping and back-end facilities as well as navigation features within specifications.

This report presents one component of the environment, i.e. the MOF (Managed Object Format) syntax parser. MOF is the notation defined within the WBEM (Web-Based Enterprise Management) framework as part of the Common Information Model (CIM). This report contains a detailed description of the model as well as a complete implementation and usage guide for the syntax parser implemented within the MODERES environment.

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

(Résumé : tsvp)

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : (33) 83 59 30 30 – Télécopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : (33) 87 20 35 00 – Télécopie : (33) 87 76 39 77

Analyseur syntaxique de spécifications aux normes MOF dans l'environnement MODERES Java

Résumé : MODERES Java est un environnement de développement de modèles de l'information de gestion. Cet environnement a pour objectif de fournir un ensemble d'outils de manipulation de modèles issus de différentes approches de gestion. Ces outils comportent des analyseurs, des traducteurs de modèles ainsi que des facilités de navigation dans les spécifications.

Ce rapport présente l'un des composants de l'environnement à savoir l'analyseur syntaxique de schémas CIM (Modèle de l'Information Commun) spécifiés au format MOF (Managed Object Format). MOF est la notation retenue dans l'approche WBEM (Web-Based Enterprise Management) pour la description des modèles de l'information ainsi que pour la spécification des bases d'information de gestion. Ce rapport comporte une description détaillée du format MOF ainsi qu'une présentation de l'implémentation et de l'utilisation de l'analyseur développé dans l'environnement MODERES.

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

1 Introduction

MODERES Java is a toolkit developed in our group to manipulate several types of Management Information Models. Its main objective is to become an open environment used for Management Information Model design and usage and providing integration and approach independence facilities. This goal can be specialized in subgoals such as:

- provision of information model parsers for most deployed management approaches,
- provision of an approach independent information model description (e.g. GIOP),
- unified specification repository API,
- provision of several back-ends to allow:
  - automated mapping between management information models,
  - code generation for all possible targets, i.e. APIs of interest to management platforms such as XOM/XMP, CORBA IDL, JMAPI, TMN++ ...
  - navigation and access facilities within information model specifications.
  - ...

All these features are provided in an integrated environment which respects following constraints:

- *portability:* the whole environment is easy to port on most software and hardware platforms. Especially MODERES is available on at least major UNIX systems, Microsoft Windows'95 and NT as well as Mac OS 8 or later;
- *management platform independence:* the toolkit is independent from any management platform or agent toolkit development environment currently available on the market;
- *extensibility:* development of additional back-ends and applications over the environment are facilitated through the definition of APIs at different levels of the environment;
- *usability:* the distributed version of the toolkit allows easy use of built-in features through a simple and user-friendly GUI.

Initially developed and distributed in C++, The MODERES Toolkit has been redesigned and coded in Java.

In its first release ([Festor 96][Festor 97]) the toolkit was distributed with only GDMO (Guidelines for the Definition of Managed Objects) [ISO-10165.4 92, ISO-10165.1 92] (and latest ammendments) as well as GRM (General Relationship Model) [CCITT.X.725 95] specification parsing and semantics checking facilities.

The toolkit has recently been extended to also support specifications described using the MOF (Managed Object Format) notation used within the CIM (Common Information Model) approach [Jander 96, Todd 97g, Todd 97e, Todd 97b, Todd 97c, Todd 97d, Todd 97f, Todd 97a, DMTF 97a, DMTF 97b].
The associated parser and specification repository are the concern of this report.

The remainder of this report is organized as follows. Section 2 provides a short summary of components included in a MOF specification. Section 3 contains statements on which components of the MOF syntax are supported within version 1.0 of the MOF parser. In section 4, the Java classes used to build the MOF repository, i.e. the abstract syntax tree are presented. Section 5 describes how to install and use the MOF Parser in a Java application. Finally, a short conclusion summarizes the supported features and sketches future work on this part of the environment.

2 The Common Information Model and the MOF notation

The Common Information Model (hereafter called CIM) [DMTF 97b], is a model which has been chosen within the Web-Based Enterprise Management framework as the approach for the definition of management information.

CIM contains several parts. The first one is a meta-model which defines all components which can be used to define management information models. Other parts of CIM concern definition of specific models i.e. the core schema, the common schema and extension schemas.

The notation used to specify information models is the MOF (Managed Object Format) notation which is integrated in the CIM model [DMTF 97b]. The notation allows the description in a computer readable form, all components of an information model specification.

Basic components of a specification (also called schema) are: the class, qualifier, instance and the association. To each component, a specific syntax is associated. The notation provides one additional statement which is used to specify schema names, name spaces, and specification inclusion.

In this section, the syntax associated to each of these components is detailed.

2.1 Class specification

The class is the basic component of an information model. A class is defined by its name, the class from which it inherits its properties, a set of attributes and methods.

Production rules associated to the definition of a class are given in figure 1. Like any MOF component, a class definition can be refined through the use of one or more qualifiers (See qualifiers section). Those qualifiers are present in the definition prefix.

```
1  [ qualifiers] CLASS <class-label>
2         [ AS <alias-identifier>] [ : <super-class-label>]
3            {
4            [  property |  method ]*
5           }
6  ;
7
8  property -> [ qualifiers]
9                dataType <property-label> [  array] [default-value] ;
10 method -> dataType <method-label>  "("  parameter [ ,  parameter*] ")" ";"
11 parameter -> dataType [ array]
12
13 array ->  "[" positiveDecimalValue  "]"
16 default-value ->  =
17                      value
18                   | {  value [ ","  value]*  }
19 value -> simpleValue
20          | objectPath
21 qualifiers ->  "["  qualifier [, qualifier*] "]"
```

Figure 1: The class specification

To each definition, a label (`class-label` line `1`) is associated. This label forms the standard identifier of the class. In addition to this label, one can define an alias identifier (`AS alias-identifier` clause line `2`). This alias can be used in other specifications to reference the class.

Inheritance is specified using the `:  superclass-label` statement in which one indicates the name of the class from which the current one inherits its properties. This clause is optional.

Each class may contain attribute definitions (properties). The associated syntax clauses are given in line `8` and `9`. An attribute is defined through a set of qualifiers (optional), a name, a type (`dataType`) which can be simple or tabular and finally, a default value.

Methods in classes are defined using the syntax provided in lines `10` and `11`. A method is defined by its name, the data type it returns as well as the set of input and output parameters and their type.

2.2 Qualifier definition

A qualifier is a component used to define a property over an element. Qualifiers are usually used to associate a more precise semantics to a class, instance, method, property or any other MOF definition. New qualifiers can be defined using the syntax provided in figure 2.

A qualifier is defined by its label, an associated data type which can be simple or tabular (*arrayDefinition* constructor line `2`). One can associate a default value to a qualifier (*initializer* clause).
For each qualifier one also defines it application domain, i.e. the components of a CIM specification to which the defined qualifier applies. This is done through the use of the **FLAVOR** clause. Target components can

```
1   QUALIFIER <identifier> : DataType
2               [ arrayDefinition ]
3               [ = initializer]
4               [ "," SCOPE "(" metaElement [ "," metaElement]* ")"]
5               [ "," FLAVOR "(" flavor [ "," flavor]* ")" ]
6              ";"
7
8  arrayDefinition -> "[" positiveDecimalValue "]"
9
10 initializer -> simpleInitializer | arrayInitializer
11
12 metaElement -> SCHEMA | CLASS | ASSOCIATION
13           | INDICATION | METHOD | PROPERTY
14           | REFERENCE | ANY
15
16 flavor -> ENABLE_OVERRIDE | DISABLE_OVERRIDE
17          | RESTRICTED | TO_SUBCLASS
```

Figure 2: Qualifier definition syntax

be schemas, classes, associations, indications, methods, attributes or reference fields or any other entity except qualifiers.

The last component of a qualifier definition defines the propagation clauses towards inheritance and redefinition. The qualifier specifier can decide which propagation rules apply to the qualifier. It can be redefined (**ENABLE_OVERRIDE**) or not (**DISABLE_OVERRIDE**). One can force or restrain the qualifier to be propagated propagated to all subclasses and instances of the qualified element (**TO_SUBCLASS** and **RESTRICTED** values).

2.3 Association specification

The syntax associated to an association definition is an extension of the syntax for class definitions. The distinction between them is made through the use of a specific qualifier named **ASSOCIATION** which indicates that the class definition is in fact an association. The second difference concerns the support within an association of special properties called references.
Each association must at least have one reference property. The associated syntax is given in figure 3.

```
1  "[" ASSOCIATION, qualifiers "]" CLASS <class-label>
2        [ AS <alias-identifier>] [ : <super-class-label>]
3        [
4           {
5            property | method | [ reference]2++
6           }
7        ]
8  ;
9
10 reference -> [ qualifiers] <class-label>  REF <reference-name>
11                                  [ default-value] ;
```

Figure 3: Association syntax

A reference is defined by a label which identifies the role, a set of qualifiers as well as the name of the class whose instances can fit the role in any instance of the association. Towards inheritance one can redefine references considering strict inheritance rules. In any instance of an association each reference must point to at least one object instance.

Within the CIM, several reference specific qualifiers are defined (`Multiplicity`, `Aggregation`, `Min`, `Max` ...).

2.4 Instance definition

An instance is defined by a set of qualifiers and the name of the class to which the instance belongs. Syntax rules for instance definition are given in figure 4.

```
[ qualifiers] INSTANCE OF <class-label>
    [ AS <alias-identifier>]
 {
 [ [ qualifiers] (<property-label>|<reference-label>) =  initializer ;]*
 }
;

initializer -> Value | aliasIdentifier
               | ArrayInitializer | ObjectPath

aliasIdentifier -> $<label>

```

Figure 4: Instance definition syntax

To each attribute or reference, one can associate values. If an attribute is not assigned in the definition, it holds its default value.

2.5 The #pragma clause

The #pragma clause is used within CIM specifications to feed the compiler with useful information concerning name spaces, other specification modules and schemas. The syntax is given in figure 5.

```
  #pragma schema ( <schema-label> );
| #pragma namespace ( <name-space-name> );
| #pragma include ( <filename> );
```

Figure 5: The #pragma clause

The pragma schema clause indicates that all following specifications belong to the schema whose name is given as parameter of the pragma statement.

The name space pragma statement allows to reference a name space whose path is given in the statement. Each specification following this statement belongs to the indicated name space.

Finally the include pragma, enables the logical inclusion of other CIM specifications within the current one.

2.6 Basic data types

Within CIM several basic data types are defined. These data types are summarized in table 1.

Type	Description
(**u**/**s**)int**xx**	signed (*sint*) or unsigned integer (*uint*). **xx** values indicate the size which can be *8*, *16*, *32* or *64*.
real**xx**	real value which can be of *32* or *64* bits. The coding is an IEEE one.
boolean	boolean
char16	16 bits character UCS-2 coding.
string	UCS-2 string.
datetime	date time format: `yyyymmddhhmmss.mmmmmmsutc` where `yyyy` represents the year, `mm` the month, `dd` the day, `hh` current time, `mm` minutes, `ss` seconds, `mmmmmm` microseconds, `s` le time difference (`+` or `-`) followed by the offset `utc` stated in minutes.

Table 1: Basic CIM data types

2.7 The graphical notation

To the above presented notation, CIM also associates a graphical notation based on UML [Booch 96]. Since the MODERES toolkit only deals in its current version with the textual notation, we will not detail the graphical one in this report. The reader will find a detailed definition of the graphical components in the CIM document [DMTF 97b].

3 The MOF Parser

The MODERES MOF front-end provides in its first release, support for version 1.1 of the MOF notation. All language components are supported except the include statement which is parsed but which has no effect on the parser itself, i.e. the include specified file is not yet loaded nor parsed. Otherwise, all statements are supported.

4 Java repository classes

The parser is used to feed a repository, i.e. a set of objects allowing access to the parsed information specification. In this section, we present the architecture of the repository and all spcification containment classes.

The repository is made up of several MOF schemas. Each specification contains:

- an associated file name from which the specification was loaded,
- a set of schema specifications,
- an ordered list of specification statements including pragma statements.

Each schema specification contains a set of MOF definitions. Figure 6 depicts OMT diagram describing all Java classes which form the overall architecture of the repository.

A definition contains an identifier an an optional set of qualifiers. A definition is specialized in:

- a class definition,
- a method definition,
- a property definition,
- an instance definition,
- a property initialization,
- a reference definition.

The associated OMT diagram of Java classes is depicted in figure 7.

A class definition contains:

- a list of properties,
- a list of methods,
- an alias identifier (optional),
- a link to the name space in which it is defined.

The association definition extends the class definition with a list of references. The associated OMT diagram is given in figure 8.

An instance definition is built by a set of qualifiers, a reference to a name space path and a set of property initializers (for both attributes and references). The associated OMT diagrams are given in figure 9 and 10.

A qualifier is defined by its label, an associated data type which can be simple or table-based, a possible default value, one or more scope definitions and flavors. An instance of a qualifier is defined through its label, a set of initializing parameters and like the definition, a set of flavors. The OMT diagrams describing those classes are given in figures 11 and 12.

The last relevant class of the repository is the one associated with the definition of methods in classes. Its diagram is given in figure 13.

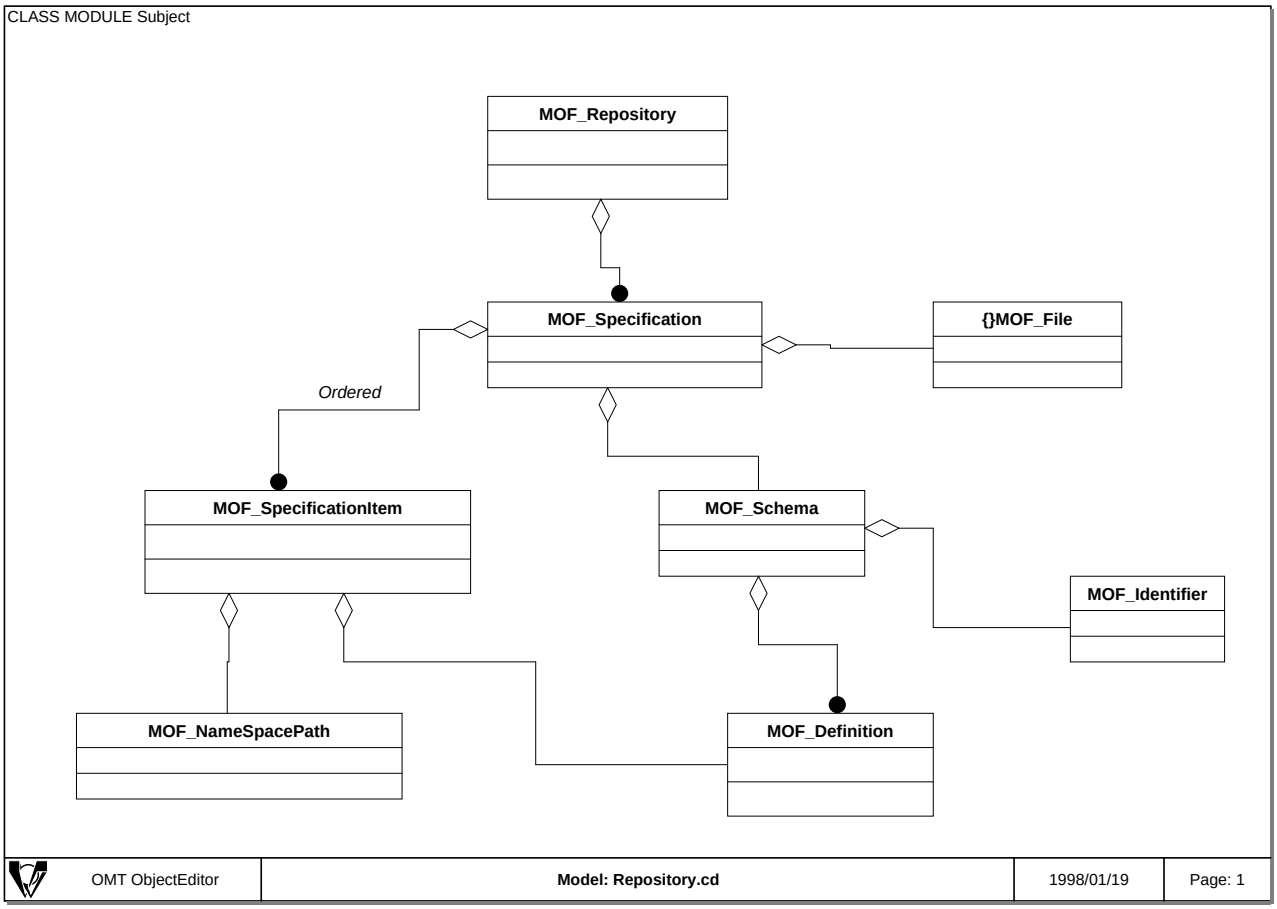


Figure 6: The Repository classes

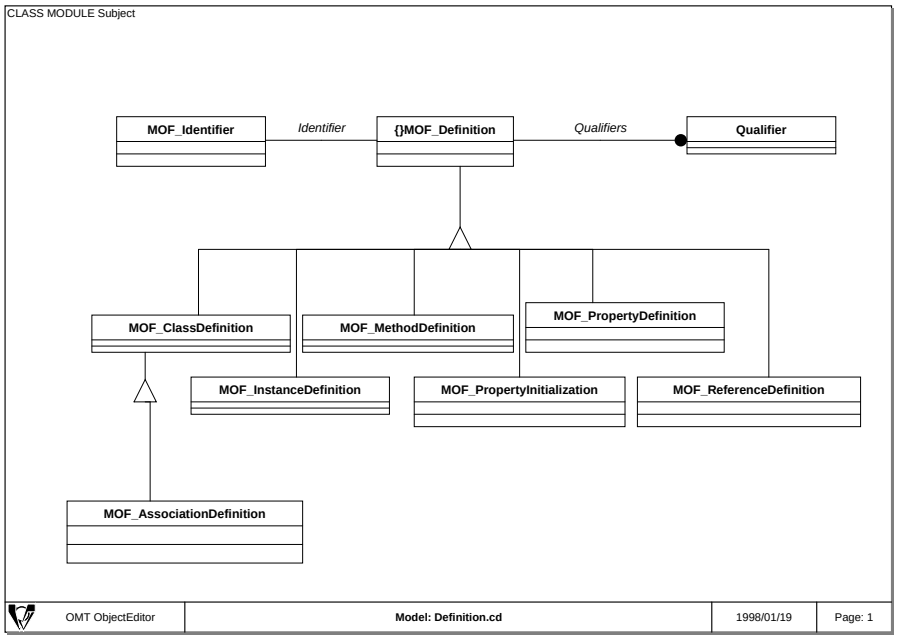


Figure 7: The Definition classes

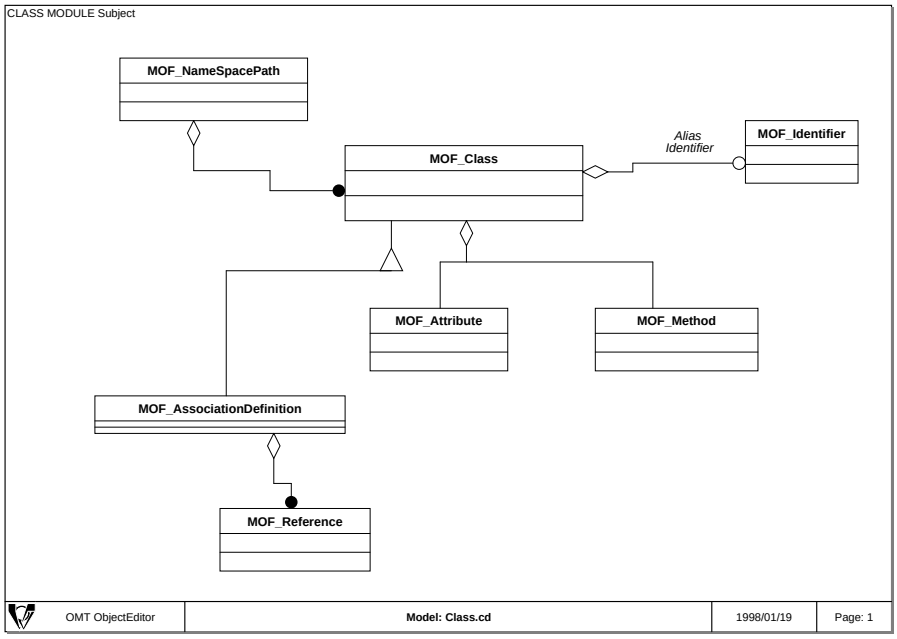


Figure 8: The class classes

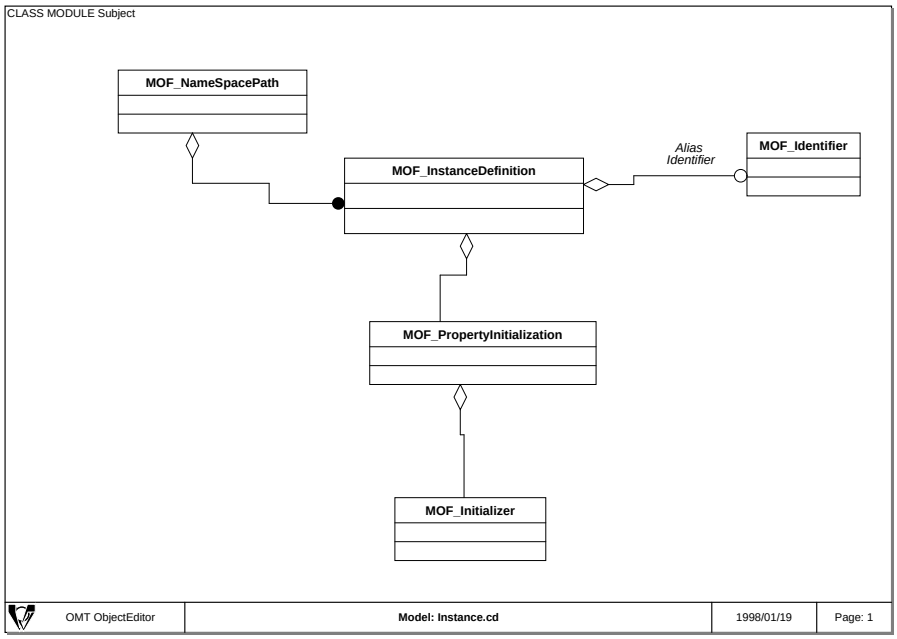


Figure 9: The Instance definition related classes

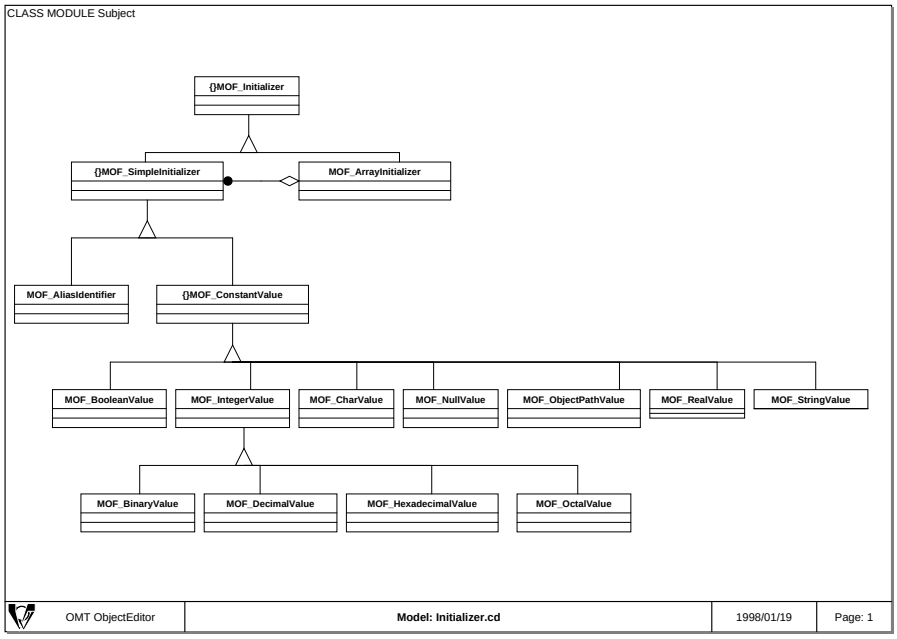


Figure 10: The Initializer classes

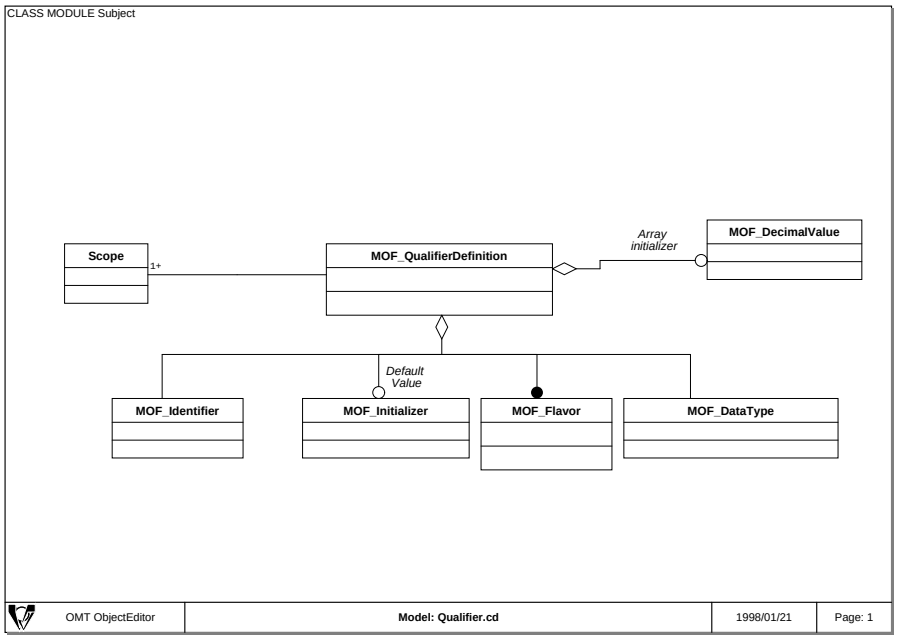


Figure 11: The Qualifier definition classes

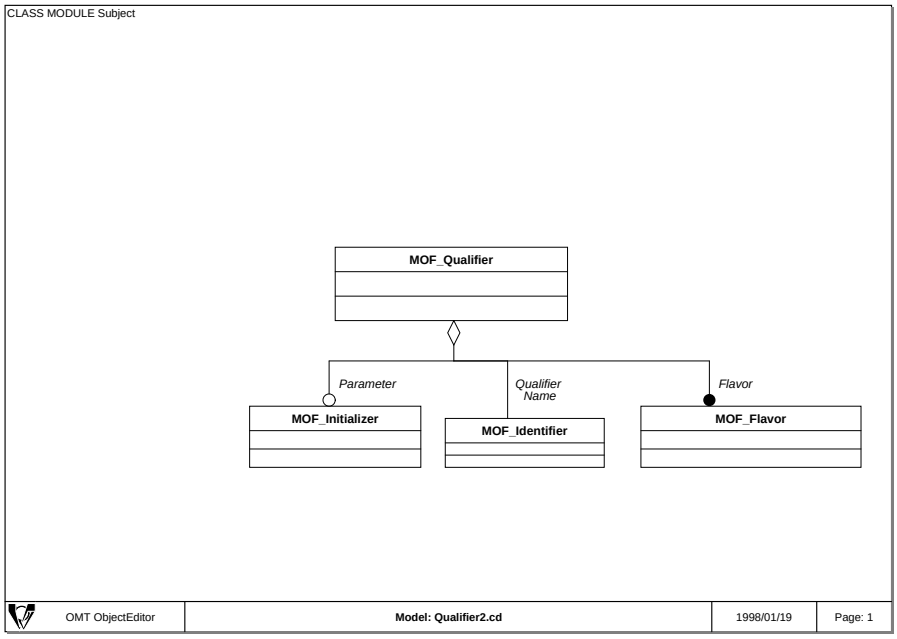


Figure 12: The Qualifier instance classes

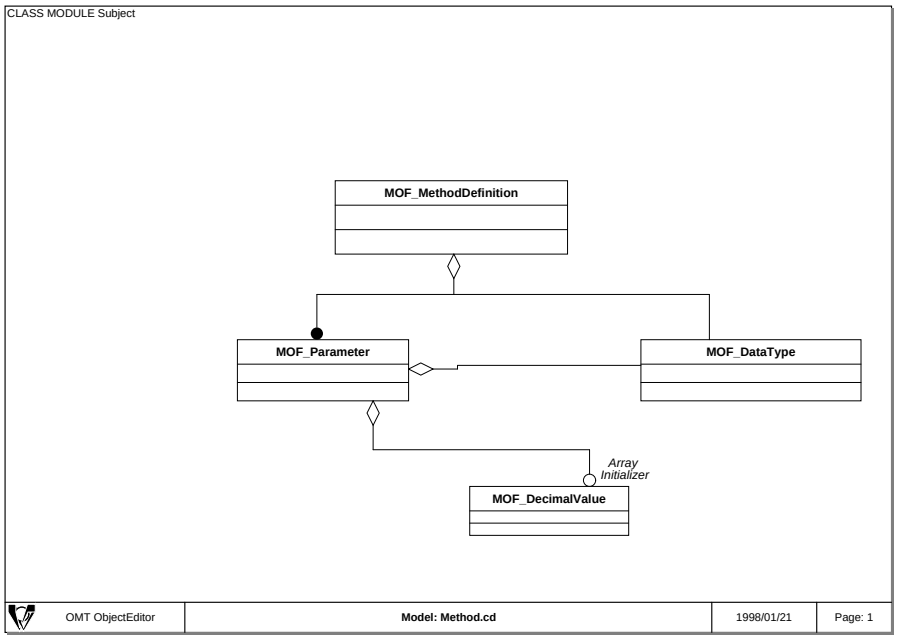


Figure 13: The Method definition classes

5 How to use the package

The MOF parser and the associated MOF repository is provided within two MODERES Java packages. The parser is defined in the `moffrontend` package and all API and repository classes are defined in the `mofrepository` package.

Both packages are provided within the MODERES toolkit and installed automatically. If a developer wants to use both packages to develop a new application, he should refer to the online documentation in the `docs/api/` directory of the MODERES distribution. There is a sample Java file in the distribution which shows how to instantiate the MOF Parser and the repository in any Java program. This file is provided in the `MODERESJava/FR/loria/resedas/moderes/mofexamples` and is called `MODERES_BasicMOFParser.java`

6 Conclusion and future work

In this report, we have presented a parser and its API supporting the MOF syntax used within the CIM model. This parser allows the loading of Version 1.1 MOF specifications and provides within the MODERES toolkit a rich set of methods to access these specifications and build back-ends and applications on top of the package.

The most important work which has to be undertaken is to make the parser compliant to version 2 of the CIM. This is currently under development. We are also working on the design and implementation of several back-ends to the MOF parser. Some of them will be freely avaliable. A semantics checker is also under development.

7 MODERES Contact & Copyright

7.1 Contact

MODERES toolkit is still under development. Several back-ends for MODERES are either in development phase or β-test and some improvements to access the repository will be available in a very short time.

We are maintaining the MODERES toolkit. So feel free to send us comments, bug reports. We will do our best to provide a stable and usage friendly tool.

Please send:

- bug reports,
- comments,
- any suggestions

to the author. We will respond in the shortest delay.

Contact person
Olivier Festor
RESEDAS Research Group
INRIA Lorraine
Technopole de Nancy-Brabois
- Campus scientifique -
615, rue de Jardin Botanique - B.P. 101
54600 Villers Les Nancy Cedex
France
E-mail: festor@@loria.fr
Tel: (33) 83.59.20.16
Fax: (33) 83.27.83.19
URL:http://www.loria.fr/festor

A WWW page is maintained for MODERES. The URL is:

http://www.loria.fr/exterieur/equipe/resedas/MODE.html

On this page you will find all new libraries developed for the the tool, new accessible applications, documentation, technical reports and white papers concerning planned extensions. You will also find many links to other network management pages from general information to companies which provide products for OSI and Internet Management.

7.2 Copyright

The MODERES environment is distributed freely under the conditions specified in the COPYRIGHT file provided with the package. The copyright is as follows:

The MODERES library is distributed freely under the conditions specified in the COPYRIGHT file provided with the library. The reader will find below the content of the Mode-PP copyright.

The Software MODERES (c) INRIA 1995 in its release 1.0 of the 16/04/1996, hereafter referred to as "The SOFTWARE".

The SOFTWARE has been designed and produced by O. Festor and E. Nataf, the researchers of the RESEDAS project, a research project of the National Computer And Automatics Institute (INRIA), Domaine de Voluceau, Rocquencourt, 78153 Le Chesnay Cedex.

INRIA holds all the patent rights concerning the SOFTWARE. The SOFTWARE has been registered at the Agency for the Protection of Programmes (APP) under the number:

IDDN.FR.001.190018.00.R.P.1996.000.10800

.

Foreword

The SOFTWARE is currently being developed and INRIA wishes for it to be used by the scientific world so as to test, evaluate and continually update it.

To this end, INRIA has decided to distribute the prototype of the SOFTWARE by FTP, in an Object-code form.

a) Extend of the rights granted by INRIA to the user of the SOFTWARE:

INRIA freely grants the right to use, modify and integrate the SOFTWARE in another program.

b) Reproduction of the SOFWARE

Clauses 9 and 10 of the Berne agreement for the protection of literary and artistic works (Union of Berne) respectively specify in their paragraphs 2 and 3 authorising only the reproduction and quoting of works on the condition that:

- *" This reproduction does not adversely affect the normal exploitation of the work or cause any unjustified prejudice to the legitimate interests of the author",*
- *"That the quotations given by way of illustration and/or tuition conform to the proper uses and that it mentions the source and name of the author if this name features in the source",*
- Any use or reproduction of the software items and/or documents exclusively owned by INRIA and carried out to obtain profit or for commercial ends being subject to obtaining the prior express authorisation of INRIA.
- Any commercial use made without obtaining the prior express agreement of the INRIA would therefore constitute a fraudulent imitation.

c) Information feed-back

Any user of the SOFTWARE shall send his comments on the use of the SOFTWARE to the INRIA at (email : festor@@loria.fr).

d) Guarantees:

Note that the SOFTWARE is a research product currently being developed.

INRIA disclaims any responsibility in any way in any instance of being obliged to put right any possible direct or indirect damage sustained by the user.

References

[Booch 96] Grady Booch et James Rumbaugh. Unified Method for Object-Oriented Development Document Set. Rapport, Rational Software Corporation, 1996. http://www/rational.com/ot/uml.html.

[CCITT.X.725 95] Comité Consultatif International Télégraphique et Téléphonique (CCITT), *Information Technology - Open Systems Interconnection - Structure of Management Information - Part 7: General Relationship Model*, International Standard, CCITT.X.725, November 1995, [ISO-10165.7 97].

[DMTF 97a] DMTF. Common Information Model (CIM) version 1.0. Rapport, Desktop Management Task Force, August 1997.

[DMTF 97b] DMTF. Common Information Model (CIM) version 1.1. Rapport, Desktop Management Task Force, September 1997.

[Festor 96] O. Festor, E. Nataf et L. Andrey. MODE-FE: A GRM/GDMO Parser and its API -Release 1.0-
Reference Manual. Technical Report no. 0190, INRIA Lorraine, 1996.

[Festor 97] O. Festor. MODERES Java: Architecture and Core Packages. Rapport no. RT-0205, INRIA, May 1997.

[ISO-10165.1 92] International Organization for Standardization (ISO), *Structure of Management Information - Part 1: Management Information Model*, International Standard, ISO-10165.1, January 1992.

[ISO-10165.4 92] International Organization for Standardization (ISO), *Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects*, International Standard, ISO-10165.4, January 1992.

[ISO-10165.7 97] International Organization for Standardization (ISO), *Structure of Management Information - Part 7: General relationship model* , International Standard, ISO-10165.7, 1997, [CCITT.X.725 95].

[Jander 96] M. Jander. Web-based Management: Weclome to the revolution. *Data Communications International*, 25(16):38–56, 1996.

[Todd 97a] S. Todd, *Draft HMMP Encoding*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

[Todd 97b] S. Todd, *Draft HMMP Events*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

[Todd 97c] S. Todd, *Draft HMMP Mandatory Schema*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

[Todd 97d] S. Todd, *Draft HMMP Query*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

[Todd 97e] S. Todd, *HMMP Overview*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

[Todd 97f] S. Todd, *HMMP Protocol Operations*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

[Todd 97g] S. Todd, *Introduction to HMMP*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

Unit´e de recherche INRIA Lorraine, Technop^ole 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 Rh^one-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)
ISSN 0249-6399