MODERES Java: Architecture and Core Packages

Olivier Festor

INRIA

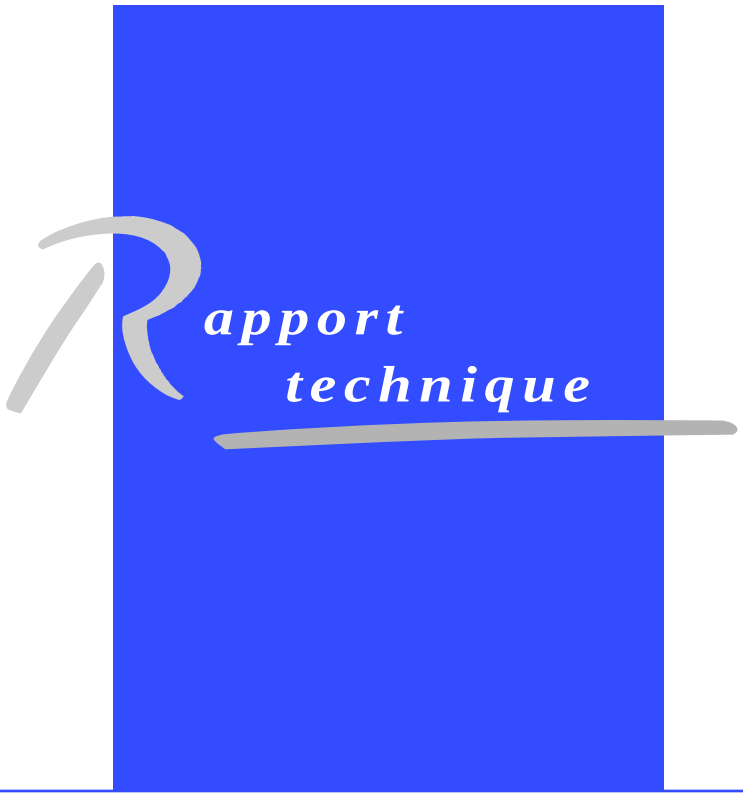
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

MODERES Java: Architecture and Core Packages

Olivier Festor

N ° 0205

THÈME 1



ISSN 0249-0803

MODERES Java: Architecture and Core Packages

Olivier Festor

Thème 1 — Réseaux et systèmes
Projet RESEDAS

Rapport technique n0205 — Juin 1997 — 18 pages

Abstract: MODERES is an environment for the development of management information models. The initial release of the environment was developped in C++ and provided tools for the manipulation of information models specified using GDMO and GRM.

This report presents the architecture and core packages of the MODERES Java environement. This environement is completely developped in Java and contains the initial MODERES tools as well as a new architecture for the use of the MODERES suite in an enterprise network.

Key-words: GDMO, GRM, Java, RGT, MODERES, SNMP, TINA, TMN

(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

MODERES Java: Architecture et composants de base

Résumé : MODERES est un environnement de développement de modèles de l'information de gestion. La version initiale de l'environnement était développée en C++ et fournissait un ensemble d'outils pour la manipulation de modèles de l'information spécifiés en GDMO/GRM.

Ce rapport présente l'architecture ainsi que les composants de base de l'environnement MODERES Java. Celui-ci comporte d'une part, les outils de MODERES sous forme de classes Java et définit l'architecture de la suite MODERES dans un environnement Intranet.

Mots-clé : GDMO, GRM, Java, RGT, MODERES, SNMP, TINA, TMN

1 Introduction

MODERES [Festor 96c] was the initial toolset developped in our group to manipulate GDMO (Guidelines for the Definition of Managed Objects) [ISO-10165.4 92] (and lattest ammendments) as well as GRM (General Relationship Model) [CCITT.X.725 95] specifications. The toolkit was developped in C++ and contained following tools and components:

- a GDMO parser (syntax parser),
- a GRM parser (syntax parser),
- a well defined specification repository (a syntax tree of C++ classes with all loaded specification informations),
- a full documented API allowing developers to manipulate loaded specifications,
- several back-ends:
  - a TeX back-end [FESTOR 96b],
  - an HTML back-end [Festor 96a],
  - a SDL'92 back-end [Samir , Tata 97],
  - an ORACLE database storing facility.

Due to the initial success of MODERES and due to the increasing demand on availability over multiple architecture, we have decided to port the toolkit on Java and extend it with several new features. Thus MODERES Java is an extension of the initial MODERES toolkit. In addition to beeing ported to Java and thus ensuring portability accross most platforms, the toolkit aims to provide additional back-ends to facilitate the use of OSI-based Information Models and their notations namely GDMO and GRM.

MODERES is distributed **FREELY** under the **COPYRIGHT** conditions as described in the `COPYRIGHT` file provided with the package. MODERES is developped as part of a research prototype on service and network management interoperability within the RESEDAS research group. It is not a closed environment, nor it is a commercial one. It aims at allowing people to experiment new features in Management Information Modelling. Even if stable and complete, it may contain minor bugs which, if known, will be fixed in new releases.

This report presents the basic architecture of the toolkit and the core packages which are the GDMO/GRM parser and the specification repository. As soon as new back-ends are available, they will be published in additional technical reports.

2 Standard Compliance

As the toolkit is used within our group to link standard Management Information Models with Formal Methods, we use the behaviour description part of the templates to put the formally described behaviour parts without affecting the standard notations.

This does not affect the compliance to the standard notations.

However, we have slightly extended the GDMO and GRM notations with three minor features. These features are:

- the ability to associate a module identifier to a set of GRM/GDMO specifications, i.e. a file,
- a behaviour clause associated with managed objects classes allowing the association of behaviour templates at the Managed Object level,
- a class keyword in the operation mapping part onto a `CREATE` operation of GRM specifications allowing the differentiation of a class identifier from the associated parameters.

These differences are described below.

2.1 Module identifiers

Each specification file may be associated with a module identifier. This module identifier is used at the semantics checking level to identify the origin of a label or a definition. Usually the module identifier is the identifier associated with a standard e.g. *"Recommendation X.721:1992:"* which identifies labels defined in the X.721 recommendation.

If required, the module identifier must be specified at the beginning of a file according to the following syntax:

MODULE " *<identifier>*";

<list of gdmo and grm specifications>

<end of file>

As an example, for the X.721 recommendation, the associated GDMO file should start with the following text:

MODULE *"Recommendation X.721:1992:"*";

...

The specification of the module identifier is optional in the MODERES framework.

2.2 GDMO extensions

We have added the support for behaviour specification at the managed object class specification level. There, the specifier can, in addition to the standard notation, define a behaviour clause like in many other templates of the GDMO and GRM framework.

Thus the new syntax associated to a GDMO managed object class template is as described below.

```
<class-label>  MANAGED OBJECT CLASS
[  DERIVED FROM <class-label> [ ,<class-label>]* ;
]
[  CHARACTERIZED BY <package-label> [ ,<package-label>]* ;
]
[  CONDITIONAL PACKAGES <package-label> [ ,<package-label>]*
                                    PRESENT IF  condition
                     [ ,<package-label> [ ,<package-label>]*
                                    PRESENT IF  condition]* ;
]
% added to the standard GDMO notation
[  BEHAVIOUR <behaviour-label>;
]
 REGISTERED AS <object-identifier> ;
```

This does not affect standard GDMO support within the parser.

2.3 GRM extensions

We have added one keyword to the relationship mapping template. There we have extended the system operation part of the operation mapping which concerns the CREATE operation with the CLASS keyword as illustrated below:

```
% old syntax
<system-management-operation> ->
        ...
        | CREATE [<class-label>] [<parameter-label>]*
        ...
```

```
%new syntax
<system-management-operation> ->
        ...
        | CREATE [ CLASS <class-label>] [<parameter-label>]*
        ...
```

The introduction of this keyword allows the front-end to distinguish a class label from a parameter label at the syntax parsing level of a specification and is thus not reported at the semantics check which acts now in a non tricky way to treat this part of a specification.

2.4 Known limits

The MODERES parser parses all GRM and GDMO files and provides a built in semantics checker even if just partially implemented in the 0.9 release. It is not yet linked to an ASN.1 compiler and thus is not able to perform all possible semantics checks.

MODERES does not make any tests on ASN.1 types. Since several ASN.1 parsers are already provided, the authors do not want to build yet another ASN.1 parser. However, since ASN.1 type identification is required for several semantical checks within GDMO and GRM specifications, a link to an existing ASN.1 compiler is planned.

3 Required software and packages

The MODERES Core package requires the availability on your machine of:

- the Java development toolkit release 1.1,
- the Java Generic Library (JGL) distributed freely by Objectspace (see URL: `http://www.objectspace.com/JGL`) Since the JGL may be distributed in executable form, we are looking how to integrated the distribution within the MODERES package.

The parser was generated using the JavaCC compiler compiler. This tool is not required to use MODERES Java.

4 Core features

In its first release, 0.9, MODERES Java, provides following features:

- an integrated GDMO/GRM parser,
- all compiled repository classes,
- a full documented API allowing programmers to build new applications on top of the parser through the repository API,
- a basic semantics checker (not completely implemented yet!),
- several examples on the use of the API, and an executable file allowing to parser GDMO/GRM specifications.

All back-ends developped for MODERESC++ are under development in the group and will be soon available within MODERES Java.

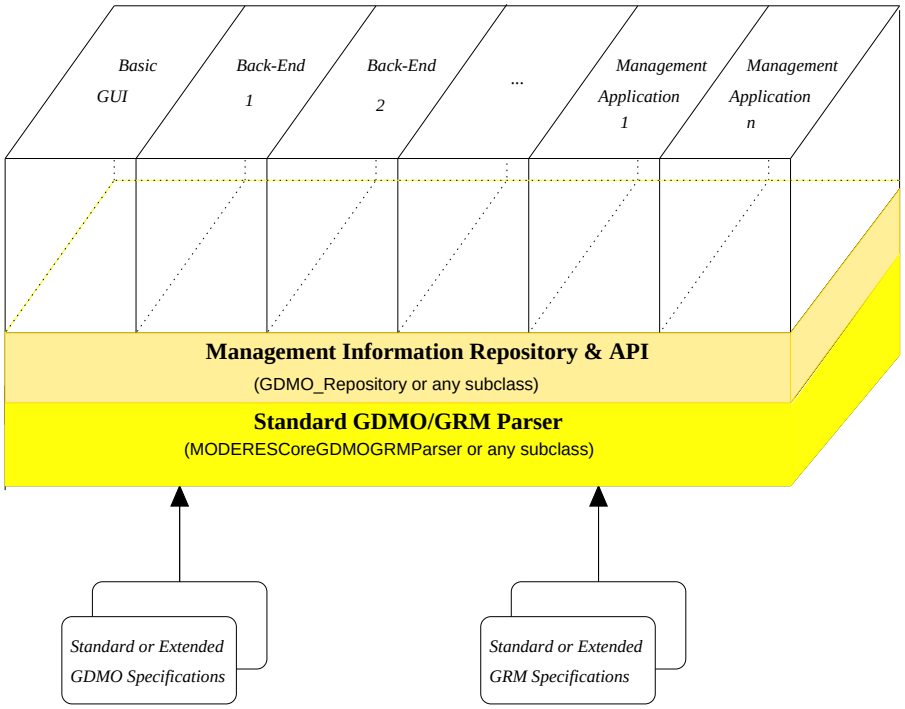


Figure 1: General architecture of the MODERES Java toolkit

5 Architecture

5.1 General Architecture

Figure 1 provides an overview of the MODERES toolkit architecture. The core of the toolkit is divided in three parts which are the parsing, the repository and its API and the back-ends and/or application ones.

Concerning the lower layers of the architecture, they are grouped into Java packages as described in the next sections.

5.2 Java packages

All code is organized in Java packages. The current release contains following packages:

- `FR.loria.resedas.moderes.repository`:

  all classes which define the repository filled during parsing of GDMO/GRM specifications and whose API allows to manipulate GDMO/GRM specifications.

- `FR.loria.resedas.moderes.frontend`:

  all classes related to the parser (front/end part). These files are the generic parser, some extended parsers and generated files from javacc (tokenManager, ErrorHandler, ...).

- `FR.loria.resedas.moderes.examples`

  all applications provided to illustrate the use of the MODERES parser, repository and its API.

In the next pages we will present more in depth all these packages.

5.3 The frontend package

The front end package contains several classes. These classes are:

- `MODERESCoreGDMOGRMParser` Basic parser class. sends all error messages as they are defined by javacc to the output stream. The associated java file is generated by javacc according to the `.jj` grammar file.

- `MODERESTextGDMOGRMParser` Sub-class of the `MODERESCoreGDMOGRMParser` class. In this subclass, error messages are redefined (simplified) and still sent to the output stream (See javacc documentation on how to extend a parser).
- `MODERESCoreGDMOGRMParserConstants`, `ASCII_CharStream`, `MODERESCoreGDMOGRMParserTokenManager`, `ParseError`, `Token` parsing files generated by javacc.

5.4 The repository package

The repository package contains all classes used to contain GDMO and GRM specifications, i.e. the syntax tree. These files are:

- `ASNType`

  simple class used to contain a reference to an ASN.1 type. In the first release of MODERES Java, this reference is just a module identifier and a type name.

- `GDMO_ActionTemplate`

  class which is used to store all fields of an action template specification.

- `GDMO_AttributeAndProperties`

  class which is used to store an attribute label and associated properties as defined in a package template specification.

- `GDMO_AttributeGroupTemplate`

  class which is used to store all fields of an attribute group template specification.

- `GDMO_AttributeTemplate`

  class which is used to store all fields of an attribute template specification.

- `GDMO_BehaviourTemplate`

  class which is used to store all fields of a behaviour template specification.

- `GDMO_ConditionalPackage`

  class which is used to store all fields of a conditional package, i.e. package label + condition as defined in a Managed Object Class template specification.

- `GDMO_Context`

  class which contains the context associated to a parameter as defined in the parameter template definition. This context includes the type and keyword.

- `GDMO_FieldAttribute`

  class which contains information associated to the field attributes (AND ATTRIBUTE IDS clause) within a notification template definition.

- `GDMO_Label`

  class which contains a label, i.e. module identifier, label as used in all GDMO/GRM specifications.

- `GDMO_LabelAssociationElement`

  class which contains one label and a list of labels associated to this label. This is used in the package template specification in the:

  - ATTRIBUTE GROUPS statement where one can extend an attribute group with attribute labels
  - ACTIONS where an action can be extended with parameters
  - NOTIFICATIONS where a notification can be extended with parameters

- `GDMO_ManagedObjectClassTemplate`

  class which is used to store all fields of a Managed Object Class template specification.

- `GDMO_Module`

  class used to contain a set of template definitions loaded from exactly one specification file. All templates are defined within the same module.

- `GDMO_NameBindingTemplate`

  class which is used to store all fields of a Name-Binding template specification.

- `GDMO_NotificationTemplate`

  class which is used to store all fields of a notification template specification.

- `GDMO_OneProperty`

  class which contains one propery associated to an attribute in a package template.

- `GDMO_OperationMapping`

  class which is used to store all fields of an operation mapping specification.

- `GDMO_PackageTemplate`

  class which is used to store all fields of a package specification.

- `GDMO_ParameterTemplate`

  class which is used to store all fields of a parameter specification.

- `GDMO_RelationshipClassTemplate`

  class which is used to store all fields of a relationship class specification.

- `GDMO_RelationshipMappingTemplate`

  class which is used to store all fields of a relationship mapping specification.

- `GDMO_RelationshipOperation`

  class which is used to store information concerning a relationship operation as used in an operation mapping template. Information contains the relationship operation type, the associated name and the role to which it applies.

- `GDMO_Repository`

  class which contains a repository of loaded modules. This class is the entry point to the MODERES repository. It offers several methods to walk through the specifications and perform actions onto the loaded specifications.

- `GDMO_Representation`

  contains all information on how a relationship is represented in the MIT. This information is used in both role mapping and role specification templates.

- `GDMO_RoleMapping`

  class which contains all information related to a role mapping specification.

- `GDMO_RoleSpecification`

  class which contains all information related to a role specification.

- `GDMO_SemanticsError`

  Basic semantics error class. The first release of the semantics error class contains a single string which enumerates the occured error. An extended class is currently under construction. This class will contain:

  - an error message string,
  - a label containing the label which causes the error,
  - a label of the template in which the error was detected,
  - additional not yet defined stuff.

- `GDMO_SmoRole`

  contains all information associated to a system management operation as used in the `maps-to` part of a role mapping template. It describes the system operation and either a role or the fact that a relationship object has been choosen.

- `GDMO_Supported`

  class which contains a supported relationship operation as defined in a relationship class specification.

- `GDMO_SystemOperation`

  class which contains information on a system operation as used in a relationship mapping template.

- `GDMO_Template`

  generic class from which all specification templates inherit. This class contains a label and an optional registration identifier.

To see which methods are offered by those classes, start a WEB browser in the `docs` directory of the distribution (See section 5.5) and go to the API documentation part. There all packages, classes and methods are fully documented.

Below we provide information on the organization of the repository API over which one can build any back-ends one wants to develop. The documentation is divided in two parts. First the inheritance hierachy of classes of the repository is presented, then each repository class is associated to the corresponding GDMO/GRM grammar part.

5.4.1 Inheritance hierarchy

The inheritance hierarchy is very simple. All classes that refer to a GDMO or a GRM template inherit from `GDMO_Template` as depicted in figure 2.

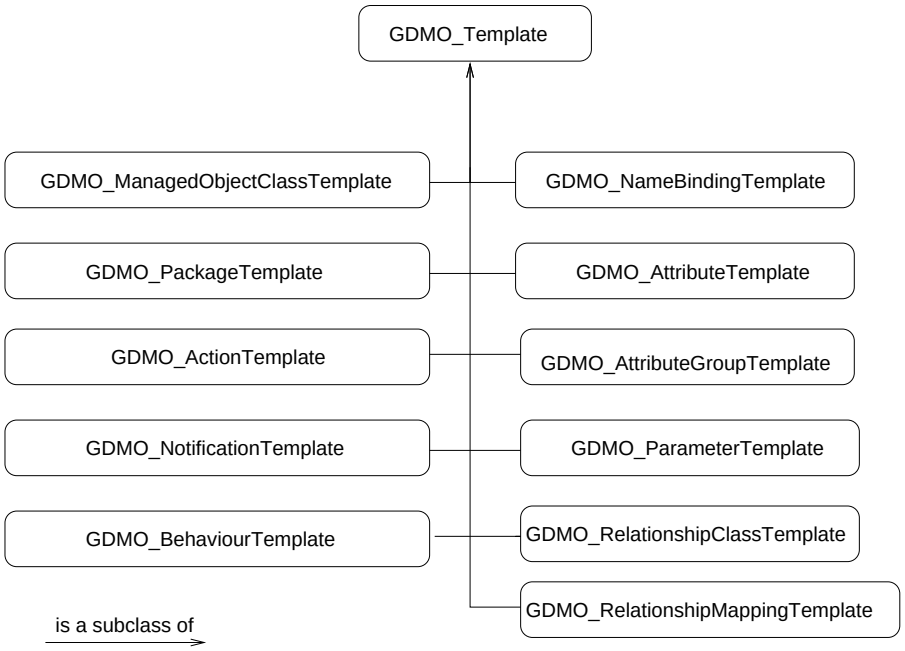


Figure 2: The inheritance hierarchy

All other classes are basic classes and do not inherit from specific classes, except some parser classes which all inherit from the `MODERECoreGDMOGRMParser` class.

5.4.2 Java repository classes and associated GDMO/GRM specifications

The BNF is not completely detailled in this document. Please refer to the related standard for a complete description of GDMO and GRM.

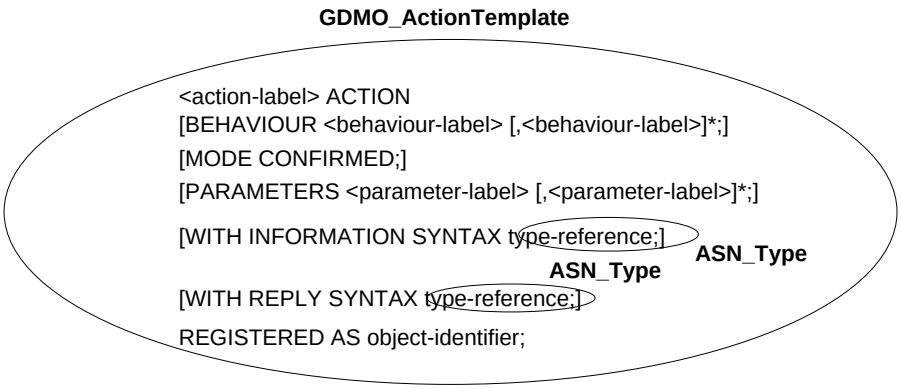


Figure 3: The action specification

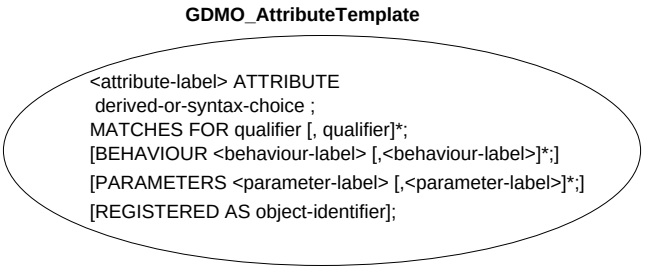


Figure 4: The attribute specification

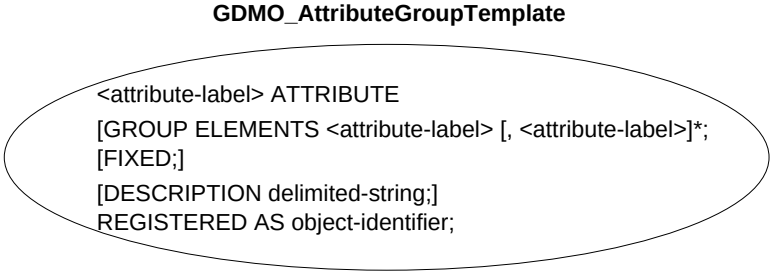


Figure 5: The attribute group specification

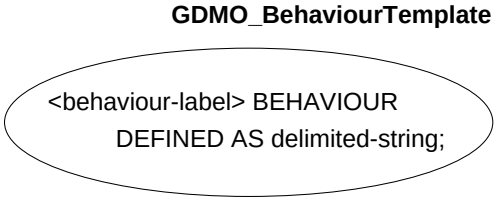


Figure 6: The behaviour specification

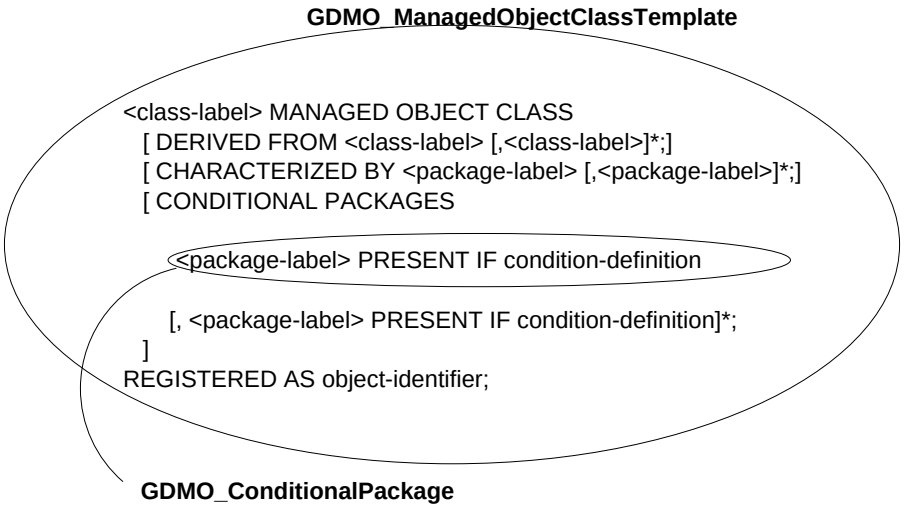


Figure 7: The managed object class specification

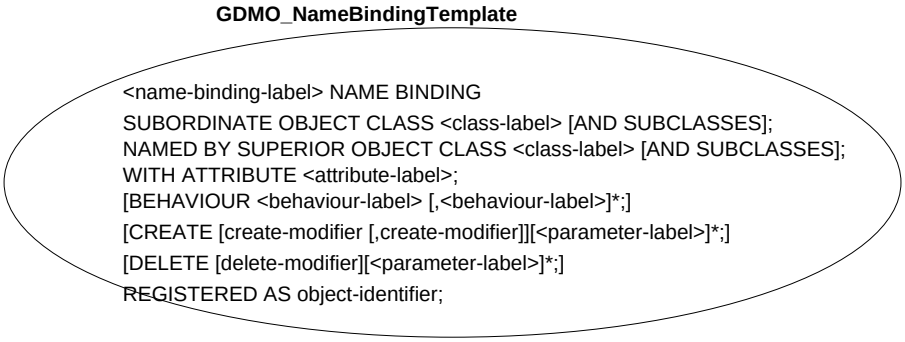


Figure 8: The Name-Binding specification

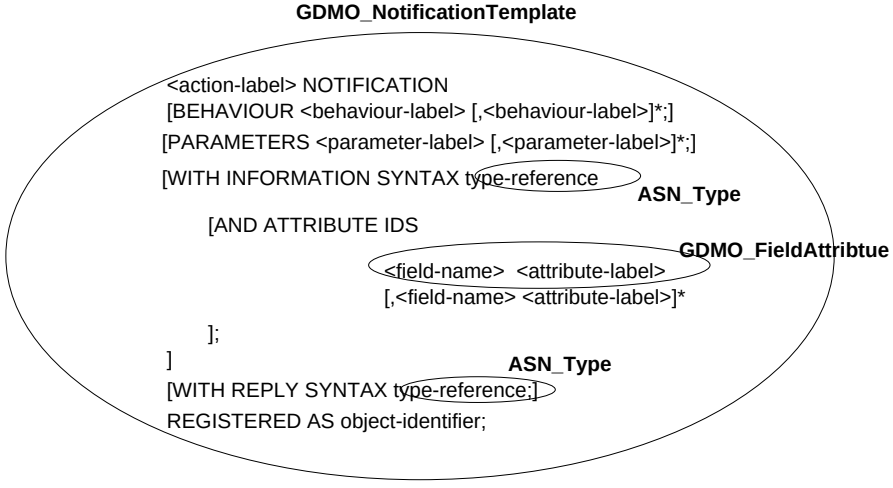


Figure 9: The notification specification

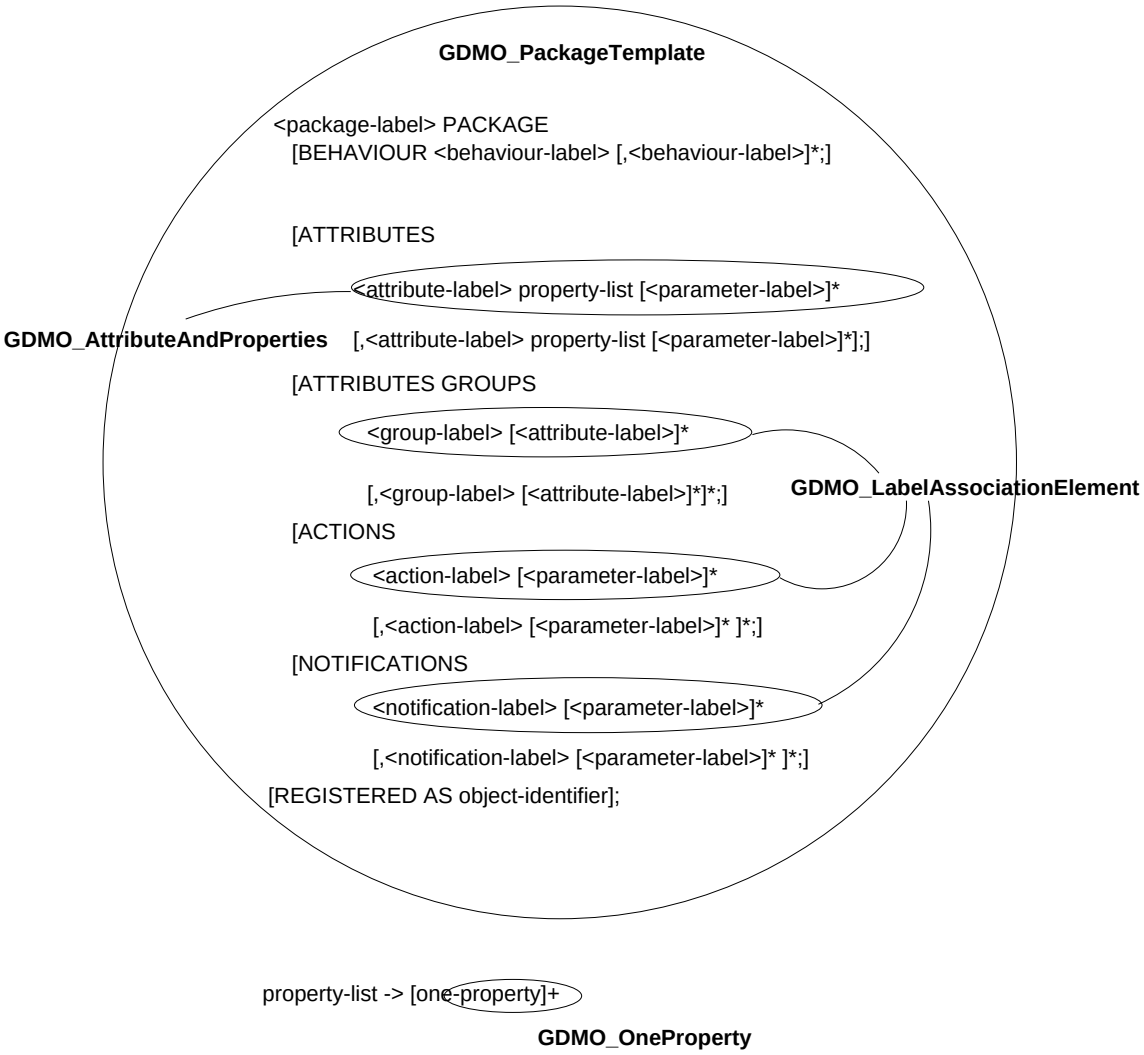


Figure 10: The package specification

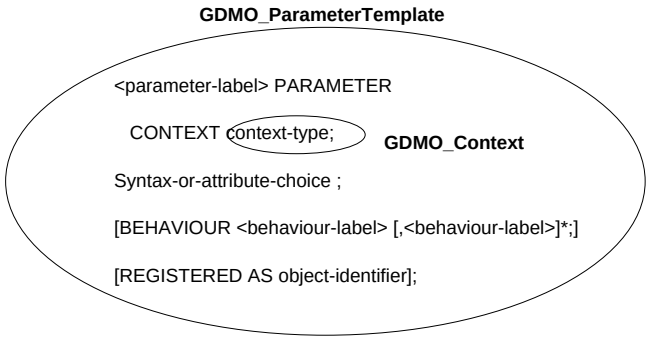


Figure 11: The parameter specification

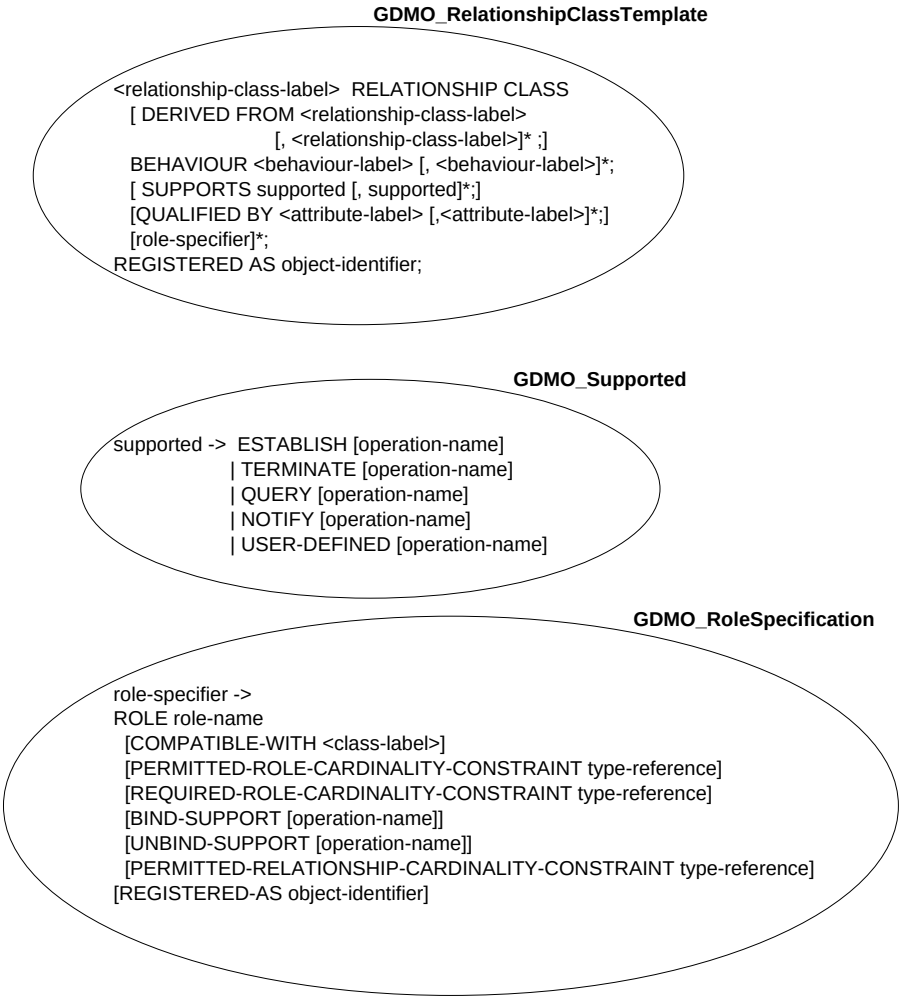


Figure 12: The Relationship class specification

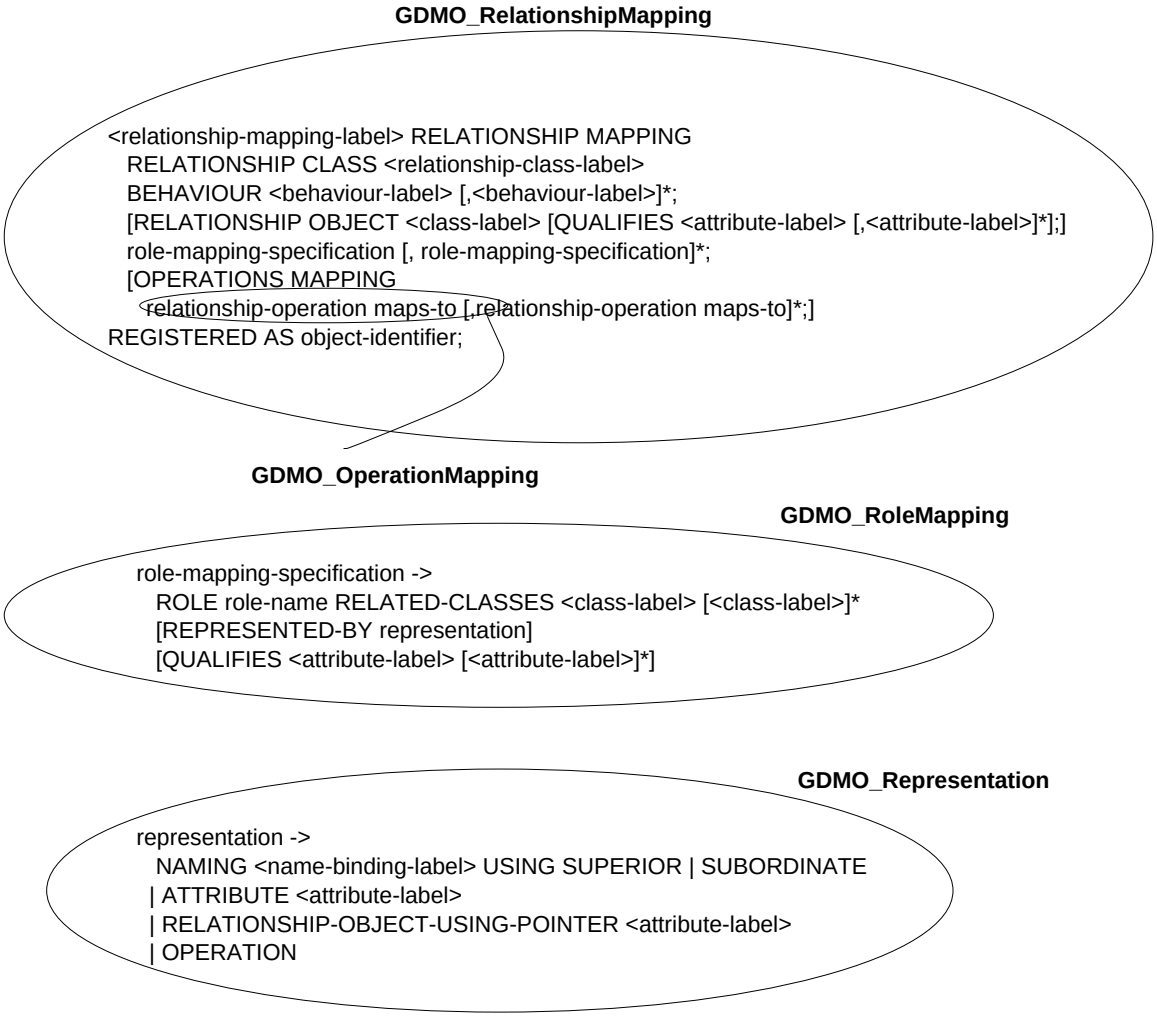


Figure 13: The Relationship mapping (1)

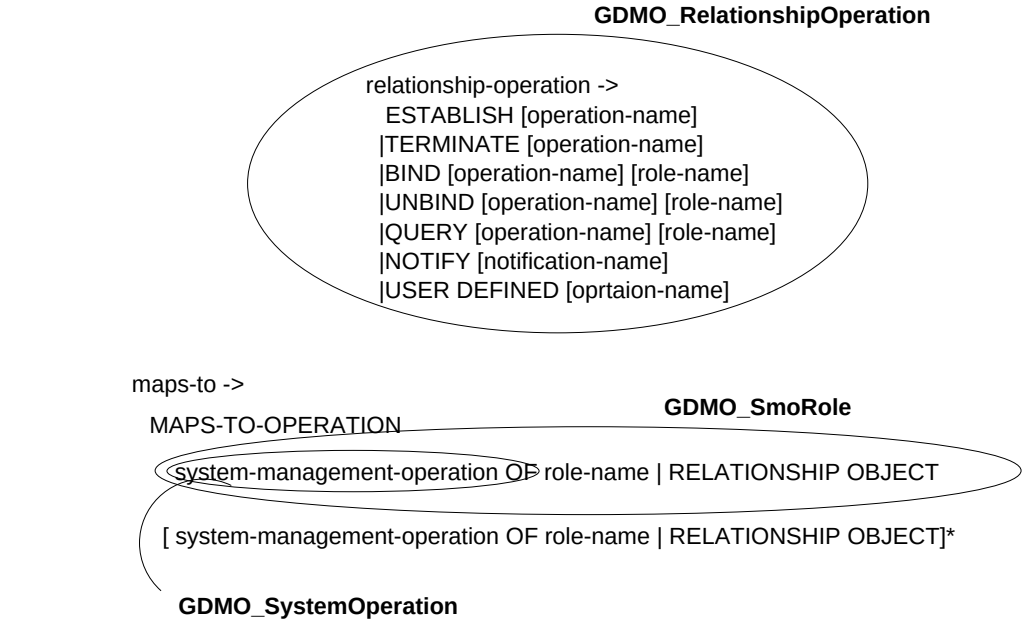


Figure 14: The Relationship mapping (2)

5.5 Directories

The whole MODERES suite is provided in a directory `MODERESJava` which when installed, contains following sub-directories:

```
src/
    FR/loria/resedas/moderes/
                              repository/
                              frontend/
                              examples/
class/
      FR/loria/resedas/moderes/
                               repository/
                               frontend/
                               examples/
bin/FR/loria/resedas/

docs/
     api/
     RR/
```

The `src` directory contains all source files for the tool. Only example source files are provided in the distribution. See `FR/loria/resedas/moderes/examples/`.

The `class` directory contains all `.class` files required to use the MODERES toolkit, except the JGL ones.

The `bin` directory contains all executables class files. These are the example applications provided with MODERES.

The `doc` repository contains all documentations required to use and extend MODERES. All you need to do is start a WEB browser and load the `index.html` file in this `docs` directory to load the whole documentation.

5.6 Basic applications

The MODERES Core package is provided with several simple examples which show how the parser and associated API can be used. Applications are provided together with their source files. The source files are located in the `src/FR/loria/resedas/moderes/examples/` directory. A compiled version of the examples are provided in the `class/FR/loria/resedas/moderes/examples/` directory.

In the first release of the toolkit, following examples are available:

- `BasicGDMOGRMParser`: allows one GDMO/GRM file to be parser and the loaded specification to be printed to another file if parsing was successfull and the `-o <output-file>` option was selected.

  to start this application, change to the `class/FR/loria/resedas/moderes/examples/` directory and type

  ```
  java BasicGDMOGRMParser <source-gdmo-grm-file [-o outputfile]
  ```

- `BasicMIFGDMOGRMParser`: allows multiple GDMO/GRM files to be parsed in one call and the repository to be filled with the loaded specifications. After this, the application starts semantics checking[1] and prints to the standard output stream all detected errors, if any.

  to start this application, change to the `class/FR/loria/resedas/moderes/examples/` directory and type:

  ```
  java BasicMIFGDMOGRMParser [<source-gdmo-grm-file]+
  ```

5.7 Documentation

The whole MODERES documentation is provided in the `docs/` directory of MODERESJava in HTML form . To access this information, start any browser and load the `index.html` file in the `docs` directory. From there, you will access any information required to use and extend MODERES.

[1] Not all semantics checks are currently implemented.

6 How to install the core package

To install the package and make things run, follow following steps.

1. get the MODERESJava.tar.gz distribution
2. expand the file

   type `gunzip MODERESJava.tar.gz`

   then type `tar -xvf MODERESJava.tar`
3. this creates the MODERESJava directory, all subdirectories and contained files.
4. set the `CLASSPATH` environment variable so that it contains a path to `MODERESJava/bin/` and `MODERESJava/src/`.
5. check that the JGL is installed and that your `CLASSPATH` points to it.
6. go to the Basic applications part of this document to test the parser.

Send us a mail if you have any problems with the toolkit.

7 Copyright

8 Contact

The 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 one of the authors. 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.

There is also a mailing list for MODERES related discussions. The moderes mailing list aims at providing a support for exchanging informations about the MODERES (Managed Object Development Environment by RESEDAS). This list will also serve to announce news and new releases of the MODERES environment.

The official language of this list is english.

The list is not moderated. Messages of this list are not archived.

Commands for the management of the list have to be sent by E-mail to the list server:

`listserv@loria.fr`

The "Subject" line must be empty. Possible commands in the body of your message are:

- to subscribe :

  `SUBSCRIBE moderes Surname Name Affiliation`

- to unsubscribe :

  `SIGNOFF moderes Surname Name Affiliation`

- to obtain help about the commands available on the mailing list:

  `HELP`

- To send a message to the list (the message wil be forwarded to all subscribers of the list) have to be sent directly to the address of the list, i.e.:

  `moderes@loria.fr`

9 Conclusion and future work

In this report, we have presented the core package and the architecture of the MODERES Java toolkit. Running over all systems on which Java is available, the MODERES toolkit provides several facilities for manipulating GDMO/GRM specifications. Since the core package comes with limited features, several extensions are under development in our group and should be available in a very short time period.

In addition to a full semantics checker, all back-ends already existing within the C++ version of MODERES will be extended and distributed soon. Several new tools are also under development, Java code generators, ... These tools will be announced in the MODERES mailing list, so stay tuned.

References

[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.

[Festor 96a] O. Festor. Mode-pp html: A gdmo/grm to html translator -release 1.0- reference manual. Technical Report no. 0199, INRIA Lorraine, 1996.

[FESTOR 96b] O. FESTOR. Mode-pp tex: A grm/gdmo pretty printing library based on mode-fe for the generation of tex documents. -release 1.0- reference manual. Rapport no. RT-0192, INRIA, May 1996.

[Festor 96c] 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.

[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.

[Samir] TATA Samir. Rapport de maîtrise.

[Tata 97] S. Tata, L. Andrey et O. Festor. *A practical experience on validating GDMO-based Information Models with SDL'88 and '92. SDLForum'97*, 1997. to appear.

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