Federating Meta-model and Model Data with EIA/CDIF's CORBA Compliant MIDDLEWARE.1

Univ.Ass. Dr. Rony G. Flatscher  
Wirtschaftsuniversität Wien  
Abteilung für Wirtschaftsinformatik  
Augasse 2-6  
A-1090 Wien  
Rony.Flatscher@wu-wien.ac.at

Abstract
This paper introduces the reader to EIA's CDIF "MIDDLEWARE.1" draft, which will allow for distributing model data, meta-model data and the CDIF Meta-meta-model definitions by employing the appropriate OMG's CORBA 2.0 standards.

With this infrastructure it becomes possible to gather information on distributed model data objects, including those objects representing the appropriate meta-model and meta-meta-model definitions. Meta-meta-model and meta-model definitions are read only, but attributes of model data can be changed at runtime. The reader should be able to assess the core technology being developed by EIA's CDIF technical committee in the form of OMG IDL definitions and is directed to additional sources of information related to this interesting and important issue.

1 Introduction
In 1996 the Object Management Group (OMG) issued a request for information (RFI) and a request for proposal (RFP) for preparing an OMG standard on "Object Analysis and Design" (OAD). In January of 1997 the following six submissions were proposed, using OMG's (cf. [W3OMG]) ordering:

1) Joint submission from Taskon/Reich, Technologies/Humans and Technology,
2) Joint submission from IBM and ObjecTime,
3) Joint submission from Rational Software, Microsoft, Hewlett-Packard, Oracle, Texas Instruments, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp,
4) Softeam,
5) Platinum Technology and
6) Ptech.

As OMG has been always interested in merging and joining proposals, if technically feasible, it usually encourages submitters to co-operate and define common standards. In the case of the OAD-RFP the first results became visible in March 1997 where almost all submitters announced that their proposed technology would be combined as complementary with that of other companies.

Some of the submission companies like Platinum or Rational are at the same time members of EIA's (abbreviation for: Electronic Industries Association) CDIF (abbreviation for: CASE Data Interchange Format) committee, which was founded in 1987. The CDIF technical committee has been developing standards for interchanging model data between tools of different vendors, by defining extendable meta-models as the basis of the transfers, which themselves are founded on the "EIA/CDIF Integrated Meta-meta-model" (cf. [CDIF94a], [Flat96]).

EIA/CDIF itself has been pro-actively working with OMG technology and has developed a draft for a standard which would allow for an additional form of distributing meta-model and model data besides CDIF's [CDIF94c] and [CDIF94d]: "MIDDLEWARE.1" (cf. [CDIF97]). MIDDLEWARE.1 defines the CDIF Meta-meta-model and meta-models in terms of OMG's interface description language (IDL) which defines interfaces for CORBA 2.0. By doing so, the OMG developed know-how and infrastructure for distributing objects among disparate systems from different vendors can be directly used for federating meta-model and model data.

The acceptance of the EIA/CDIF set of interim standards in the industry has been impressively shown by some of the submissions to OAD's RFP by OMG, e.g.:

1) Rational's joint submission to OMG of its specifications ("Unified Method Language", abbreviated: UML) uses explicitly CDIF's Meta-meta-model and the EIA/CDIF "Foundation Subject Area" meta-model (cf. [CDIF94e]) for transferring model data. The actual transfer occurs according to the CDIF interim standards (cf. [CDIF94b], [CDIF94c] and [CDIF94d]), employing a clear text encoding and a batch transfer.

2) Pyramid Technologies' proposal uses explicitly the EIA/CDIF interim standards for exchanging model data among different tools. In addition to Rational's adoption of CDIF, Pyramid Technologies also proposed the usage of the CDIF MIDDLEWARE.1 draft as of October 1996 (cf. [CDIF96c]).
All the submissions which tackled the problem of exchanging model data among tools of different vendors referred to the EIA/CDIF set of standards for achieving this important ability.

In the course of the RFP for OAD the OMG work may yield an OMG standard on OAD as early as September 1997 and it most likely will base transfer of model data on the CDIF architecture and its transfer interim standards. Because of the fact that an OMG standard is devised, it seems to be very likely that the CDIF MIDDLEWARE.1 proposal, which builds exclusively on the OMG developed CORBA-standard, will become part of the OAD standard for exchanging model data. Due to these ongoing efforts OMG and CDIF officially established a liaison with each other in March 1997.

2 CDIF Architecture

The CDIF architecture is documented in [CDIF94a], a brief overview is given e.g. in [Flat96].

2.1 Meta-meta-model

Figure 1 depicts the CDIF Meta-meta-model in the form of an extended entity-relationship diagram which defines the concepts and the extended entity-relationship model which gets used for defining meta-models. Rectangles in figure 1 represent entity types (named "meta-meta-entity"), lines with arrows relationship types (named "meta-meta-relationship"), lines without arrows depict the superclass/subclass hierarchy. All subclasses inherit all meta-meta-attributes and meta-meta-relationships of their superclass(es).

![Figure 1: CDIF Meta-meta-model, cf. [CDIF94a], p. 33.](image)
2.2 Meta-model

Instances of the Meta-meta-model entity types build the so-called meta-model. Instances of the CDIF Meta-meta-model relationship types are used to relate the instances of the CDIF Meta-meta-model entity types to each other. In meta-models the generic term "meta-object" is used to refer to the so-called "meta-entities", "meta-relationships" and "meta-attributes".

The CDIF interim standard meta-model [CDIF94e], named "Foundation Subject Area", is built according to the rules imposed by the CDIF Meta-meta-model and defines the root meta-model for all CDIF based transfers of model data. It contains the definitions of the fundamental meta-objects, which must be subclassed in/directly by all CDIF compliant meta-models. The two fundamental meta-objects consist of RootEntity for representing entity type concepts and IsRelatedTo (full name RootEntity.IsRelatedTo.RootEntity) for representing relationship type concepts in meta-models. Both in turn subclass the founding meta-object RootObject.

3 CDIF Transfer

A CDIF transfer allows for transferring, i.e. exchanging model data in a standardized, non-proprietary way. A CDIF transfer contains the meta-model according to which the model data was composed and is to be found in the exchange. If it is necessary for a transfer to refer to predefined i.e. standardized meta-model definitions, it is possible to merely denominate them. An important feature of CDIF transfers is the ability to extend any meta-model for the purpose of a CDIF exchange, should such a need arise.

A CDIF transfer demands from the exporter to supply the maximum amount of information and to not expect that just a certain tool will import the exported data. On the other hand an importer is expected to read all transfer data and build a so-called "working model" for it, even containing model data it would not be able to understand.

3.1 Batch Transfer ("SYNTAX.1", "ENCODING.1")

In 1994 the interim standards [CDIF94b], [CDIF94c] and [CDIF94d] defined a clear-text encoded, batch oriented CDIF transfer. From the beginning the CDIF technical committee envisioned additional ways of transferring CDIF compliant model data, hence hinting of possible additional syntaxes and encodings by starting the numbering of the interim standard definitions with "1".

---

1 Meta-meta-entities, meta-meta-relationships and meta-meta-attributes are sometimes dubbed with the more generic term "meta-meta-object".

2 Instances of meta-objects represent the model data to be transferred respectively exchanged.

3 In such a case the importer is expected to possess all necessary definitions of the referred to subject areas.

4 After a successful import the importer is allowed to discard all model data it does not understand conceptually.
3.2 CORBA Exchange ("MIDDLEWARE.1")

In 1996 the CDIF technical committee developed a draft (cf. [CDIF96c]) for adding a radical new way of exchanging CDIF compliant model data based on the OMG middleware standard CORBA (cf. [OrHaEd96], [ÖsRiVo96]). This paper refers to the second draft of the proposed CDIF interim standard "MIDDLEWARE.1" (cf. [CDIF97]), which should be finalized in summer 1997.

3.2.1 Framework

"MIDDLEWARE.1" allows for instantiating the CDIF Meta-meta-model and CDIF meta-models as CORBA objects and methods are defined to be able to exploit the relationships between Meta-meta-entities resp. meta-entities.

The overall framework in terms of the OMG/IDL can be pictured like ([CDIF97, p. 7]):

```
module CDIF {   /* define the CDIF module     */
   /* module/interface definitions in MIDDLEWARE.1 */
   <definition of CDIF-datatypes>
   <definition of exceptions>
   <module definition for CDIF "MetaMetaModel">
   <module definition for CDIF "Foundation Subject Area">
   <module definition for a CDIF standardized Subject Area>...
}; /* module CDIF end */
```

All parameters for methods are of type "IN". If a method returns more than one object reference then a CORBA sequence will be used. In such cases the appropriate typedef-statements will use a name with the trailing term Set (e.g. "typedef sequence<MetaObject> MetaObject_Set") in "MIDDLEWARE.1". Extensions to standardized meta-models or the definition of non-standardized subject areas have to be defined as separate modules and are not part of the standardized CDIF namespace.

3.2.2 CDIF Standardized Datatypes

The definitions for meta-meta-objects occur with meta-meta-attributes and for meta-objects with meta-attributes. MIDDLEWARE.1 first defines the thirteen CDIF datatypes available for assigning them to (meta-)meta-attributes with typedef statements (e.g. "typedef string TextDT"). Then all datatype definitions are used in the union defined for AnyDT which gets used for defining the type of the parameters for accessing (meta-)meta-attributes (cf. [CDIF97, p. 11ff]):

```
union AnyDT switch( short) { case 1 : BitMapDT theBitMapValue;
   case 2 : BooleanDT theBooleanValue;
   case 3 : DateDT theDateValue;
   case 4 : EnumeratedDT theEnumeratedValue;
   case 5 : FloatDT theFloatValue;
   case 6 : IdentifierDT theIdentifierValue;
   case 7 : IntegerDT theIntegerValue;
   case 8 : IntegerListDT theIntegerListValue;
   case 9 : PointDT thePointValue;
   case 10 : PointListDT thePointListValue;
   case 11 : StringDT theStringValue;
```
In the case that a (meta-)meta-attribute does not contain a value (dubbed: "no value") then the discriminator needs to be set to a value which corresponds to the default section.

3.2.3 Defined Exceptions

There are four exceptions defined: IllegalDataType (value is not drawn from the datatype's domain), ValueMustBePresent (value for a mandatory meta-/meta-attribute is missing), IllegalTraversal (meta-/meta-object does not participate in given meta-/meta-relationship) and IllegalAccess (meta-object does not contain given meta-attribute).

3.2.4 CDIF Meta-meta-model Mapping

The IDL definitions for the CDIF Meta-meta-model build the module MetaMetaModel_02_00 which defines interfaces for all meta-meta-entities according to the CDIF Meta-meta-model. The meta-meta-attributes of meta-meta-objects are read only, i.e. their values can be retrieved but not set. Therefore there are get-methods defined only. It is also notable, that for this particular mapping there are no exceptions defined.

What follows are the IDL definitions as proposed in [CDIF97, 12ff], demonstrating that the mapping of the CDIF Meta-meta-model into OMG's IDL is clear and straightforward.

```idl
module MetaMetaModel_02_00 {
    interface MetaObject;
    interface MetaEntity;
    interface MetaRelationship;
    typedef sequence<MetaObject> MetaObject_Set;
    interface MetaObject {
        AnyDT getAliases();
        AnyDT getCDIFMetaIdentifier();
        AnyDT getConstraints();
        AnyDT getDescription();
        AnyDT getName();
        AnyDT getUsage();
        typedef MetaEntity MetaMetaEntity;
        typedef MetaRelationship MetaMetaRelationship;
        MetaMetaEntity get_metaMetaEntity();
        MetaObject_Set traverseSrcToDest( in MetaMetaRelationship which );
        MetaObject_Set traverseDestToSrc( in MetaMetaRelationship which );
        boolean is_identical( in MetaObject other_object );
    }
    interface SubjectArea : MetaObject { AnyDT getVersionNumber(); }
    interface CollectableMetaObject : MetaObject { }
    interface MetaAttribute : CollectableMetaObject {
        AnyDT getDataType();
        AnyDT getDomain();
        AnyDT getIsOptional();
        AnyDT getLength();
    }
};
```
The CDIF Meta-meta-model is regarded to be at the same time the meta-model for itself therefore the statements "typedef MetaEntity MetaMetaEntity" and "typedef MetaRelationship MetaMetaRelationship" allow for interchanging both modeling levels within the methods defined with MetaObject and inherited via subclassing.

The methods defined for MetaObject allow e.g. for finding all source or destination (meta-)meta-entities a particular (meta-)meta-entity is related to via a given (meta-)meta-relationship. Method get_metaMetaEntity() returns the object reference of the class of which the (meta-)meta-entity is an instance of. There is no method defined to determine the instances of any given (meta-)meta-object, hence there is no navigation possible from the Meta-meta-model to a meta-model and from a meta-model to its appropriate model data.

### 3.2.5 Meta-model Mapping

The CDIF draft predefines one OMG/IDL module for the standardized meta-model "Foundation Subject Area" (cf. [CDIF94e]), whose RootEntity resp. IsNotRelatedTo serve as the superclasses for all other subject area meta-entity and meta-relationship definitions. The root meta-object RootObject defines the only meta-attributes for this subject area (cf. [CDIF97, 15ff]):

```plaintext
module Foundation_01_00 {
    interface RootEntity;
    interface RootEntity_IsRelatedTo_RootEntity;
    typedef sequence<RootEntity> RootEntity_Set;
    typedef sequence<RootEntity_IsRelatedTo_RootEntity> RootEntity_IsRelatedTo_RootEntity_Set;
}

interface RootObject {
    AnyDT getCDIFIdentifier();
    void setCDIFIdentifier( in AnyDT value ) raises( IllegalDataType, ValueMustBePresent );
    AnyDT getDateCreated();
    void setDateCreated( in AnyDT value ) raises( IllegalDataType );
    AnyDT getDateUpdated();
    void setDateUpdated( in AnyDT value ) raises( IllegalDataType );
    AnyDT getTimeCreated();
    void setTimeCreated( in AnyDT value ) raises( IllegalDataType );
    AnyDT getTimeUpdated();
    void setTimeUpdated( in AnyDT value ) raises( IllegalDataType );
    MetaMetaModel_02_00::AttributableMetaObject get_attributableMetaObject();
}
```
As with the CDIF Meta-meta-model mapping there are methods defined for traversing meta-relationships in subject areas e.g. for finding all source or destination meta-entities a particular meta-entity is related to, all meta-relationships a meta-entity participates in, as well as returning the object reference for the meta-entity resp. meta-relationship the instance belongs to. Each additional subject area needs to in/directly subclass this foundation interface definition.

4 Conclusions

The presented draft for an OMG/CORBA compliant standard for interacting and thereby retrieving definitions of CDIF meta-models and their appropriate model data has some interesting implications:

- federating the meta-model and thereby models being instances of meta-models is possible by the general means of OMG's CORBA; no CDIF specific definitions for federating resp. distributing CDIF meta-model and model data are necessary for "MIDDLEWARE.1" to achieve this important and powerful feature,

- according to the proposal "MIDDLEWARE.1" a method for testing whether an object reference is identical to another one is only firmly defined with regard to the meta-entities, meta-relationships and meta-attributes building the meta-model by using the meta-meta-attribute "CDIFMetaIdentifier". The CDIF standards define a meta-attribute
CDIFIdentifier with arbitrary values for uniquely identifying meta-entities and meta-relationships for the purpose of transferring model data within a regular (batch-oriented) CDIF transfer. Hence, values for the meta-attribute CDIFIdentifier are transient and therefore no predefined implementation for is_identical() is suggested for meta-model instances;

- the full names for meta-relationships in "MIDDLEWARE.1" are generated by prepending the source meta-entity name and appending the target meta-entity with an underscore (_) rather than with a point as in the CDIF (Meta-)meta-model definitions (cf. [CDIF1994a]), as an example one can lookup the above definition of the meta-relationship RootEntity_IsRelatedTo_RootEntity,

- "MIDDLEWARE.1" does not define a repository which would help in locating specific meta-model definitions or model data. Instead it only determines that the standardized "CORBA naming services" are to be used for distributed object lookup by name and defines the namespace rules to be applied to distributed CDIF meta-objects (cf. [CDIF1997, p.24ff]).

For object-oriented system designers the OMG/IDL interface definitions may serve as a starting point to discuss the implementation of CDIF management and exchange software with object-oriented development environments. For final implementations it is advisable to wait until the "MIDDLEWARE.1" draft is finalized and agreed upon with OMG in order to rely e.g. on firm definitions on the intended inheritance model on the one hand, and the rules to be applied for resolving multiple inheritance on the other hand.

5 References


