Rony G. Flatscher

Metamodeling in EIA/CDIF - Meta-Metamodel and Metamodels

Article (Accepted for Publication)
(Refereed)

Original Citation:

Flatscher, Rony G.

(2002)

Metamodeling in EIA/CDIF - Meta-Metamodel and Metamodels.

ACM Transactions on Modelling and Computer Simulation, 12 (4).

pp. 322-342. ISSN 1558-1195

Metamodeling in EIA/CDIF – Meta-Metamodel and Metamodels

Rony G. Flatscher
Wirtschaftsuniversität Wien
Augasse 2-6
A-1090 Vienna, Europe
Rony.Flatscher@wu-wien.ac.at

ABSTRACT
This paper introduces the EIA/CDIF set of standards for the modeling of information systems and its exchange among computer aided software tools of different vendors. It lays out the meta-metamodel and the standardized metamodels which get fully depicted in a hierarchical layout and annotated with the unique identifiers of all the standardized modeling concepts. The paper also stresses the fact that EIA/CDIF has been used as the baseline in the creation of an international standard, the ISO/CDIF set of models, an ongoing project.

Categories and Subject Descriptors
B.8.m [**Miscellaneous**]: Design management; D.2.2 [**Design Tools and Techniques**]: Computer-aided software engineering (CASE), Object-oriented design methods; D.2.4 [**Software/Program Verification**]: Model checking, Validation; D.2.9 [**Management**]: Software configuration management; D.2.10 [**Design**]: Methodologies; D.2.13 [**Reusable Software**]: Domain engineering, Reuse models; H.1.1 [**Systems and Information Theory**]: Information theory; H.2.1 [**Logical Design**]: Schema and subschema; H.3.1 [**Content Analysis and Indexing**]: Dictionaries; I.6.4 **[Model Validation and Analysis**]; I.6.5 [**Model Development**]: Modeling methodologies; I.7.2 [**Document Preparation**]: Format and notation, Languages and systems; K.6.3 [**Software Management**]: Software development, Software maintenance.

General Terms
Management, Documentation, Design, Standardization, Languages.

Keywords
CDIF, Case Data Interchange Format, EIA, meta-metamodels, metamodels, languages, system design, clear text encoding.

1. INTRODUCTION
The American industries association „Electronic Industries Association“ (abbreviated: EIA; since January 1st 1999 it is called „Electronic Industries *Alliance*“) started the effort for standardizing the exchange of model data created with different computer aided software engineering (abbreviated: CASE) tools in 1987. The EIA/CDIF (CDIF is an acronym for: „CASE Data Interchange Format“) standardization group encompassed vendor companies among others IBM, Platinum, ORACLE, as well as companies like Aerospatiale and Boeing. This way both the producers of CASE tools and the users deploying these CASE tools came together to create the means for exchanging model data between CASE tools of different vendors. In the average there were between four and six week-long meetings organized per year for carrying out the standardization work. In between of the meetings the draft documents were edited and reviewed at the subsequent meetings.

In 1991 the first set of standards were issued by EIA/CDIF, defining an overview of and the framework for the EIA/CDIF architecture, which basically consisted of the meta-metamodel, which defines the concepts available for creating EIA/CDIF metamodels. Such metamodels in turn specify the available concepts for creating models and as long as the metamodel definition is shared among CASE tool makers, models being instances of metamodels can be exchanged. EIA/CDIF metamodels are created for so-called „subject areas“ like conceptual data modeling or dataflow modeling with the intention to capture the core semantics of all modeling variants available for such an area. It is interesting to note that from the very beginning the ability to extend EIA/CDIF-compliant metamodels was of paramount importance to the standardization group.

In 1994 a revised version of the EIA/CDIF meta-metamodel „Framework for Modeling and Extensibility“ ([7], [20]) together with a recensioned overview „CASE Data Interchange Format - Overview“ ([6], [25], [26]) was standardized and drew from the experiences gained in creating EIA/CDIF metamodels. In addition a clear-text exchange format was finalized, where the general rules were layed out in the document „CDIF Transfer Format – General Rules for Syntaxes“ ([8], [27]), the syntax for such exchanges was layed out in „SYNTAX.1“ ([9], [28]) and the encoding in „ENCODING.1“ ([10], [29]).[1] In this series of standards the very first, the founding EIA/CDIF metamodel „Foundation“ ([11], [30]) was standardized which has to be referred to in all EIA/CDIF compliant metamodels.

In the subsequent years additional standards were finalized by EIA/CDIF. In 1995 the „Common Subject Area“ ([12], [31]) was created, which contains all concepts the EIA/CDIF group thought that may be applicable to more than one metamodel. The EIA/CDIF metamodel standard „Data Modeling“ ([13], [32]) covers the subject area of conceptual data modeling, including complex relationship types. The EIA/CDIF metamodel standard „Data Flow“ ([14], [33]) defines the concepts available for creating data flow models and the standardized „Presentation and

[1] The number „1“ in the titles for both, the syntax and the encoding standards is intended to point out that there may be numerous syntaxes and encodings to exchange the *same* model data!

Location Subject Area“ [15] which is concerned with exchanging graphical representations of models in a three-dimensional space. In addition to these official standards there exist drafts for the subject areas „Data Definition“, „Object-oriented Analysis and Design Core“, „Business Process Modeling“, „State Event Modeling“, „Project Management Planning and Scheduling“, and „Computer Aided Control Systems Design“.

In 1997 an additional means for exchanging model data was created as an EIA/CDIF standard, called „OMG IDL Bindings“ ([16], [21]). This particular standard defines CORBA IDL compliant interfaces for the EIA/CDIF meta-metamodel and the founding EIA/CDIF metamodel „Foundation“ as well as the rules for creating the appropriate definitions for additional EIA/CDIF compliant metamodels. With the help of this standard metamodels and model data can be distributed. As a result it is possible to make them available via networks and because of the defined interfaces to allow for discovering and inspecting metamodel definitions as well as model data in real time. Because of the definition of appropriate interfaces it is possible to query and set model data as well.

Aside from creating the EIA/CDIF standards the different members of the standardization group established relationships to different other standardization bodies and groups. One result of this work yielded an European ECMA standard in 1997: „Portable Common Tool Environment (PCTE) – Mapping from CASE Data Interchange Format (CDIF) to PCTE“ [3]. Another very important official liasion was established with the „Software Engineering Data Description and Interchange“ (abbreviated: SEDDI) working group of ISO: ISO/IEC JTC1/SC7/WG11. This particular group has been working on an international version of EIA/CDIF, dubbed „ISO/CDIF“. Informal contacts were established with ANSI (X3L8, X3H4), ECMA (TC33:PCTE), IEEE (P1175), and OMG (MOF/UML/SMIF which created the XMI set of standards).

In 1998 EIA/CDIF turned the baseline of its standards and drafts over to ISO/IEC JTC1/SC7/WG11[2] and finished its active work on additional EIA/CDIF standards. All EIA/CDIF standards are available from EIA.

2. THE EIA/CDIF ARCHITECTURE

The EIA/CDIF architecture follows a four layer scheme as depicted in table 1 and resembles the architecture which originally was devised for ANSI‘s „Information Resource Dictionary System“ [1] and which is being used by OMG’s „Meta Object Facility“ [34] too.

Table 1: Four-layer architecture

M3	**M**eta-**M**eta**m**odel	the EIA/CDIF meta-metamodel
M2	**M**eta**m**odel	the EIA/CDIF metamodels „Foundation“, „Common“, ...
M1	**M**odel	the model, e.g. the process ‚check reconciliation‘, ...
M0	User data	e.g. check # „M0123“, customer „Waldorf Astoria“, ...

In table 1, starting from the bottom, the ‚M0‘ layer is concerned with user data, where the valid structure of such data is defined at the ‚M1‘ level, which therefore can be regarded as the intension of ‚M0‘ data and is called the ‚model‘. In table 1 the user data „M0123“ (a value for the attribute ‚check #‘ of a ‚check‘) and the value „Waldorf Astoria“ (a value for the attribute ‚name‘ of a ‚customer‘) are extensions (instances) of concepts defined in the model at the ‚M1‘ level. Here, the model at the ‚M1‘ layer represents the process of ‚check reconciliation‘ and will contain the exact definitions of the structure of ‚check‘ and ‚customers‘.

This abstraction process is applied to the ‚M1‘, ‚M2‘ and ‚M3‘ layer. The ‚M2‘ level defines the intension for models at the ‚M1‘ level, typically for a specific modeling domain, called ‚subject area‘ in EIA/CDIF. Because the models of the ‚M2‘ layer describe the intension of models at the ‚M1‘ layer, the models at ‚M2‘ are named metamodels (‚models of models‘). In table 1 the metamodel for the model example could be from the model domain of ‚business process‘ or ‚workflow modeling‘. The metamodels ‚Foundation‘, ‚Common‘ are examples of EIA/CDIF metamodels as are ‚Data Modeling‘ and ‚Dataflow Modeling‘.

The top layer ‚M3‘ is concerned with determining the constructs available for creating metamodels and represents itself a model of metamodels. Consequently, it is called the ‚meta-metamodel‘ layer. In EIA/CDIF a version of an extended entity-relationship-attribute model is used as the meta-metamodel. Therefore the main building blocks are „entity types“, „relationship types“ and „attributes“. For the purpose of describing the concepts of the ‚M3‘ layer the meta-metamodel itself is being used, hence the intension for the meta-metamodel is the ‚M3‘ model itself.

The EIA/CDIF four layer architecture allows for four intension/extension pairs: M3/M3, M3/M2, M2/M1 and M1/M0. The standardization efforts of EIA/CDIF are focused on the meta-metamodel, the metamodels, the extensibility of metamodels and the exchange of metamodel and model data. Allowing for extending standardized metamodels is of paramount importance to EIA/CDIF as it becomes possible to use EIA/CDIF for CASE tools which do have additional or more semantically refined concepts than the standards themselves. Yet, tools adhering to the published standards only are still able to import models based on extended metamodels as the extensions must be given within the exchanged data, discarding the unknown metadata and its instantiation after that process, but keeping the semantically standardized parts of it.

Conceptually, metamodel data is created by instantiating the meta-metamodel, model data by instantiating the appropriate metamodel. EIA/CDIF is not concerned about the ‚M0‘ layer which would be the result of instantiating the model itself.

EIA/CDIF compliant exchange of model data must adhere to

- the EIA/CDIF meta-metamodel, and
- the fundamental EIA/CDIF metamodel ‚Foundation‘, which may get refined by means of specialization; the mandatory usage of this metamodel causes it to become the root of every

[2] In 2000 the working groups 11, 14, 15 and 16 were merged into a new working group named “ODP and Modelling Languages”, which has been established as “ISO/IEC JTC1/SC7/WG19” and has been in operation since November 1st, 2000. The outstanding standardization work on ISO/CDIF encompasses the completion of the metamodels “Data Definition”, “State Transition”, “Data Modeling”, “DataFlow Modeling”.

EIA/CDIF metamodel and ensures that any such metamodel is a rooted tree.

2.1 The EIA/CDIF Meta-Metamodel

The EIA/CDIF meta-metamodel realizes the M3 layer of the EIA/CDIF architecture. It defines the concepts which are available for creating metamodels and can therefore be regarded to be the intension of metamodels. If one implements the meta-metamodel in a (relational, object-oriented) database management system, it can be used to store the metamodel definitions and hence may serve as a repository for EIA/CDIF metamodels.

Figure 1 depicts this groundlaying EIA/CDIF meta-metamodel.

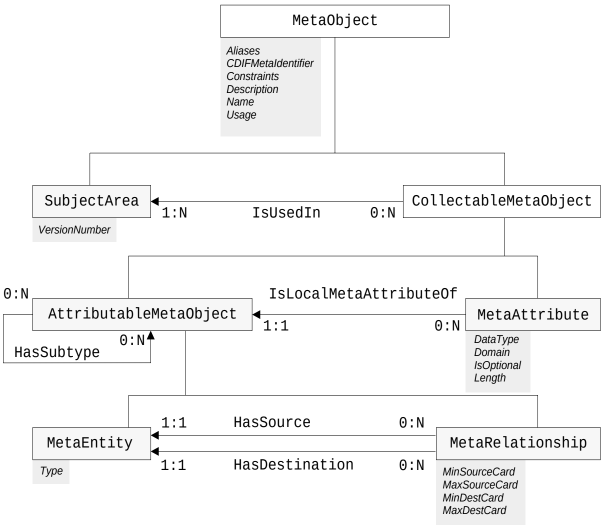

Figure 1: The EIA/CDIF Meta-Metamodel (M3 layer).

The EIA/CDIF meta-metamodel represents an extended entity-relationship-attribute-model (cf. [2]). It consists of a generalization hierarchy with the entity type „MetaObject" at its root, specialized attributable entity and relationship types. Entity types may be abstract or concrete and are represented as rectangles. In figure 1 concrete entity types which may get instantiated are depicted with a light gray background. In the EIA/CDIF standards an entity type at the M3 layer is synonymously termed „meta-meta-entity".

Relationship types have a binary multiplicity and allow for defining participation constraints in the form of cardinalities with a minimum/maximum notation. They are depicted as arrows determining the direction in which a fully qualified relationship type name has to be read. The fully qualified relationship type name is built by first denoting the name of the source entity type, followed by a dot, the name of the relationship type followed by a dot and the name of the destination entity type to which the arrowhead points to. E.g., the fully qualified name of the relationship type „IsUsedIn" therefore is called „0:N CollectableMetaObject.IsUsedIn.SubjectArea **1**:N". The minimum/maximum cardinalities of this relationship type has to be interpreted as: „an instance of type CollectableMetaObject *must* participate in at least one relationship of type" and „an instance of type SubjectArea *may* participate in such relationships". In the EIA/CDIF standards a relationship type at the M3 layer is synonymously termed „meta-meta-relationship".

Entity and relationship types may carry attributes, which may draw their values according to one of the following pre-defined data types: *BitMap, Boolean, Date, Enumerated, Float, Identifier, Integer, IntegerList, Point, PointList, String, Text* and *Time*. In the EIA/CDIF standards an attribute at the M3 layer is synonymously termed „meta-meta-attribute". At the M3 layer only meta-meta-entities may carry attributes.

The following entity types are defined in the EIA/CDIF meta-metamodel:

- „MetaObject" (abbreviated: MO): this abstract entity type is the root entity type and defines those attributes which all specialized entity types share: *Aliases* (data type: *String*), *CDIFMetaIdentifier* (mandatory, data type: *Identifier*), *Constraints* (data type: *Text*), *Description* (mandatory, data type: *Text*), *Name* (mandatory, data type: *Identifier*) and *Usage* (data type: *Text*).
- „SubjectArea" (abbreviated: SA): this entity type represents a modeling domain and defines the mandatory attribute *VersionNumber* (data type: *String*). In EIA/CDIF the union of all *SubjectAreas* yields the conceptual „Integrated Meta-model" (abbreviated: IMM). Hence, the relationship type „IsUsedIn" serves as a viewing mechanism to extract those definitions from the IMM which pertain to a specific SubjectArea.
- „CollectableMetaObject" (abbreviated: CMO): this abstract entity type allows for relating the collectable entity types to SubjectAreas via the relationship type „IsUsedIn". Because of the participation constraint defined for this relationship type, Instances of type *CollectableMetaObject* participate fully in instances of type *IsUsedIn*.
- „AttributableMetaObject" (abbreviated: AMO): this entity type serves as the supertype of „MetaEntity" and „Meta-Relationship". With the help of the relationship type „IsLocal-MetaAttributeOf" it is possible to assign attributes (entities of type *MetaAttribute*) to all AMOs. The relationship type „Has-Subtype" allows for subtyping and because of the defined cardinalities multiple inheritance is available.
- „MetaAttribute" (abbreviated: MA): this entity type allows for defining attributes which according to the cardinalities given for the relationship type „HasLocalMetaAttributeOf" must be related to exactly one instance of type for *AttributableMeta-Object*. The following meta-meta-attributes are available: *DataType* (mandatory, data type: *Enumerated*), *Domain* (mandatory for *Enumerated* data types, data type: *Text*), *IsOptional* (data type: *Boolean*), *Length* (mandatory for *String* data types, data type: *Integer*).
- „MetaEntity" (abbreviated: ME): this entity type allows for defining entity types and defines the attribute *Type* (data type: *Enumerated*).
- „MetaRelationship" (abbreviated: MR): this entity type allows for defining relationship types and defines the following mandatory attributes: *MinSourceCard* (data type: *String*), *MaxSourceCard* (data type: *String*), *MaxDestCard* (data type: *String*). Clearly, these attributes allow for determining the minimum and maximum cardinalities with respect to the source and destination *MetaEntity* which each gets defined by the relationship types „HasSource" and „HasDestination". It follows that the relationship types are binary only and that the

meta-metamodel and metamodels can only be constructed with binary relationship types which simplifies the understanding and implementation of M3 and M2 models.[3]

The following relationship types are defined in the EIA/CDIF meta-metamodel which do not possess any attributes:

- *„IsRelatedTo"* („0:N CollectableMetaObject.IsRelatedTo.-SubjectArea 1:N"): This relationship type allows for assigning CMOs to SAs. Conceptually, this is the EIA/CDIF viewing mechanism. Instances of type *CollectableMetaObject* participate fully in this relationship type (participation constraint: „**1**:N"). There can be any number of CMOs related to SAs and a particular instance of type CMO can be related to more than one SA.
- *„IsLocalMetaAttributeOf"* („0:N MetaAttribute.IsLocalMeta-AttributeOf.AttributableMetaObject 1:1"): This relationship type allows for assigning MAs to AMOs. Instances of type *MetaAttribute* participate fully in this relationship type (participation constraint: „**1**:1"). There is no limit imposed on the number of MAs an AMO may carry.
- *„HasSubtype"* („0:N AttributableMetaObject.HasSubtype.-AttributableMetaObject 0:N"): This relationship type allows for subtyping of AMOs. According to the cardinalities multiple inheritance is available, i.e. a specialized AMO may have more than one AMO as its direct supertype.[4] In the case of multiple inheritance no specific order is implied.
- *„HasSource"* („0:N MetaRelationship.HasSource.Meta-Entity 1:1"): This relationship type allows for determining which ME serves as the source for a given MR. Each instance of type *MetaRelationship* must participate exactly once in relationships of this type.
- *„HasDestination"* („0:N MetaRelationship.HasDestination.-MetaEntity 1:1"): This relationship type allows for determining which ME serves as the destination for a given MR. Each instance of type *MetaRelationship* must participate exactly once in relationships of this type.

2.2 Standardized EIA/CDIF Metamodels

All EIA/CDIF compliant metamodels are conceptually constructed by instantiating the appropriate entity types of the M3 layer. In addition every such metamodel must use the standardized and fundamental EIA/CDIF metamodel „Foundation" as a starting point and refines by the means of specialization directly or indirectly the entity type „RootEntity" and the relationship type „IsRelatedTo". This ensures that every metamodel represents a rooted tree.

In this section all standardized EIA/CDIF metamodels are sketchily introduced by depicting the appropriate generalization hierarchies in a much more informative and concise form than with the original standards.[5] Of course, the detailed definitions of the EIA/CDIF metamodels need to be taken from the original standards, which can be ordered from EIA.[6]

2.2.1 The Fundamental EIA/CDIF Metamodel „Foundation"

The fundamental EIA/CDIF metamodel „Foundation" [11] (abbreviated: FND) consists of an instance of *AttributableMeta-Object*, an instance of *MetaEntity* and an instance of *Meta-Relationship*. Figure 2 depicts this metamodel.

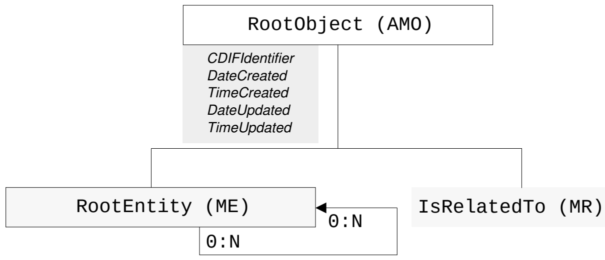

Figure 2: The fundamental EIA/CDIF metamodel „Foundation".

„RootObject" is the sole instance of *AttributableMetaObject* and forms the root of this metamodel and defines those attributes which all specialized *AttributableMetaObjects* share: *CDIFIdentifier* (mandatory, data type: *Identifier*), *DateCreated* (data type: *Date*), *TimeCreated* (data type: *Time)*, *DateUpdated* (data type: *Date*) *and TimeUpdated* (data type: *Time*). It is represented as a rectangle. The mandatory attribute *CDIF-Identifier* serves as a surrogate for uniquely identifying all *AttributableMetaObjects.*

„RootEntity" is an instance of *MetaEntity* and represents an entity type at the M2 layer. It and all of its subtypes are represented as rectangles. In the EIA/CDIF standards an entity type at the M2 layer is synonymously termed „meta-entity".

„IsRelatedTo" is an instance of *MetaRelationship* and represents a relationship type at the M2 layer, although it is an instance of an entity type at a higher level (M3). It and all of its subtypes are represented as arrows. In the EIA/CDIF standards a relationship type at the M2 layer is synonymously termed „meta-relationship".

[3] Although the definitions of metamodels can only use binary relationship types, this does *not* necessarily imply, that models themselves could not take advantage of n-ary relationship types. As a matter of fact the EIA/CDIF metamodel standard for conceptual „Data Modeling" allows for the definition of n-ary, even complex relationship types.

[4] This merely states that it is possible to simultaneously and directly re-use the definitions of more than one AMO by the means of specialization.

[5] The structural analysis of the standardized EIA/CDIF metamodels was originally carried out by the author for [23], all of the generalization hierarchies which are given below are excerpted from it.

[6] Although [23] comprehensively analyses and documents EIA/CDIF, the book is written in German. Yet, in the context of the work for this book the original standardization text was processed and an HTML-rendering created which can be found in [37]. Yet, as there is some graphical documentation missing from the original standards, one needs to get them from EIA (this is especially true for EIA/CDIF's conceptual „Data Modeling" metamodel which employs exclusive relationship types that are documented graphically only, [23], pp356-383) in order to be able to state „full compliance with the EIA/CDIF standard".

Every EIA/CDIF compliant metamodel will have to at least specialize this fundamental metamodel by refining the fundamental M2-entity type *RootEntity* and M2-relationship type *IsRelatedTo*.

This particular metamodel consists of 8 *CollectableMetaObjects*; specifically: 1 *AttributableMetaObject*, 1 *MetaEntity*, 1 *Meta-Relationship* and 5 *MetaAttributes*. A list representation of the generalization hierarchy is given in figure 3.

1. RootObject *1*
    1. RootEntity *2*
    2. 0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3*

Figure 3: Generalization Hierarchy of the Metamodel „Foundation“.[7]

2.2.2 Standardized „Semantic“ EIA/CDIF Metamodels

All EIA/CDIF metamodels of this section are regarded to be of „semantic“ nature, defining concepts for a specific subject area (modeling domain). In this respect these metamodels can be seen as tool and vendor independent ontologies. Tool exporters and tool importers may analyze and comprehend the EIA/CDIF metamodels independently (of each other) and create the mappings from their particular metamodel to EIA/CDIF's. This way each exporter and importer is only concerned with the mapping to and from EIA/CDIF's metamodel, both are insulated from each other.[8]

In the case that some concepts of the exporter's metamodel are not available in EIA/CDIF's, then the extensibility mechanism can be used to extend the standardized metamodel definitions to cover the proprietary ones by means of subclassing existing EIA/CDIF *AttributableMetaObjects*. An importing tool is allowed to discard non-standardized data from the transfer, yet is able to grasp as much of the meaning of the exported data as possible.

From the model at the M3 level and the fundamental M2 model „Foundation“ it follows that every metamodel is an entity-relationship-attribute model and can be regarded to be a (graphical) language for creating models. All model definitions are created by instantiating the appropriate *AttributableMeta-Objects* of the respective metamodel.

2.2.2.1 The EIA/CDIF Metamodel „Common“

The EIA/CDIF metamodel „Common“ [12] (abbreviated: CMN) defines those *AttributableMetaObjects* which the EIA/CDIF committee believed that are usable/applicable in more than one subject area. E.g. the meta-entity *AlternateName* together with the meta-relationship „1:1 RootEntity.*Has*.**AlternateName** 0:N“ allows for annotating any metaentity (due to relating to *Root-Entity*) with synonyms. This metamodel makes a distinction between meta-entities conveying semantic (*SemanticInformation-Object*) and those conveying presentational information (*PresentationInformationObject*). For *SemanticInformationObject* one is able to explicitly categorize the respective *Abstraction-Level*. Also, this metamodel allows for capturing information about the user (*ToolUser*) who created a meta-entity.

This particular metamodel consists of 45 *CollectableMetaObjects*; specifically: 1 *AttributableMetaObject*, 10 *MetaEntity*, 9 *Meta-Relationship* and 25 *MetaAttributes*. A list representation of the generalization hierarchy is given in figure 4.

1. RootObject *1*
    1. RootEntity *2*
        1. AbstractionLevel *12*
        2. **AlternateName** *14*
        3. PresentationInformationObject *30*
        4. SemanticInformationObject *4*
            1. DataObject *22*
            2. **Derivation** *26*
            3. ProcessObject *31*
        5. **TextualConstraint** *51*
        6. ToolUser *56*
    2. 0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3*
        1. 0:N RootEntity.*CreatedBy*.ToolUser 0:1 *68*
        2. **1**:1 RootEntity.*Has*.**AlternateName** 0:N *69*
        3. 0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1 *70*
        4. 0:N RootEntity.*Uses*.AlternateName 0:1 *71*
        5. 0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N *72*
        6. **1**:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N *73*
        7. **1**:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N *74*
        8. 0:N **TextualConstraint**.*IsConstraintOn*.Semantic-InformationObject **1**:N *80*

Figure 4: Generalization Hierarchy of the Metamodel „Common“.

2.2.2.2 The EIA/CDIF Metamodel „Data Model“

The EIA/CDIF metamodel „Data Model“ [13] (abbreviated: DMOD) defines those *AttributableMetaObjects* which the EIA/CDIF committee believed that are used in most entity-relationship-attribute based conceptual data modeling tools. For the purpose of building components the EIA/CDIF „general structuring mechanism“ (abbreviated: GSM) is used which distinguishes between *DefinitionObjects* and *ComponentObjects*. Some meta-entities in this metamodel employ multiple inheritance; if a meta-entity belongs to more than one supertype it is depicted in italic in figure 5.

An overview of the most important concepts defined in this metamodel:

- A *DataModel* consists of the *DataModelObjects Entity*, *Relationship* or *Cluster*, which at the same time are *DefinitionObjects*.
- *DataModelObjects* play *Roles* which are assigned to *Relationships* which associate the *DataModelObjects* with

---

[7] After the name of each *AttributableMetaObject* the standardized pre-defined identifier value – enclosed in asterisks (*) - for the attribute *CDIFIdentifier* is given. Meta-entities are printed in bold, if they fully participate in some meta-relationship. Meta-relationships are fully qualified and their cardinalities is given as well; the unqualified meta-relationship names are shown in italic

[8] If all exporters and importers established („bilateral“) mappings with each other, a geometric series of mappings would be necessary.

each other. It is possible that multiple *DataModelObjects* play the same *Role*.

- *Cluster*, *Entities*, *Relationships*, *Roles* and *RolePlayers* may possess *Attributes* respectively, *ProjectedAttributes*.
- *Clusters* may be built of *Clusters*, *Entities* and *Relationshpis*.
- *Relationships* may associate two or more *Entities* and can be complex by relating *DataModelObjects* with each other.
- *Entities* are identified via a (primary) *Key* and may possess multiple *CandidateKeys* as well as *ForeignKeys*.
- *AccessPaths* can be built of *Attributes* or *Keys*.
- It is possible to define outer and inner cardinalities for the *Relationships* between *DataModelObjects*.
- For *InheritableDataModelObjects* (*Entities* and *Relationships*) it is possible to indicate that they are abstract or concrete. *SubtypeSets* can be defined to be either disjoint or overlapping.

This particular metamodel consists of 146 *CollectableMetaObjects*; specifically: 1 *AttributableMetaObject*, 23 *MetaEntity*, 36 *MetaRelationship* and 86 *MetaAttributes*. A list representation of the generalization hierarchy is given in figure 5.

1. RootObject *1*
    1. RootEntity *2*
        1. SemanticInformationObject *4*
            1. **AccessPath** *1001*
            2. ComponentObject *8000*
                1. Attribute *17*
                    1. ProjectedAttribute *37*
            3. DataModel *1008*
            4. DataModelObject *1012*
                1. *Cluster* *1005*
                2. InheritableDataModelObject *1033*
                    1. *Entity* *1016*
                    2. ***Relationship*** *1040*
            5. **DataModelSubset** *1013*
            6. DefinitionObject *8002*
                1. *Cluster* *1005*
                2. *Entity* *1016*
                3. ***Relationship*** *1040*
                4. **Role** *1042*
                5. **RolePlayer** *1048*
            7. **Key** *1035*
                1. CandidateKey *1003*
                2. **ForeignKey** *1032*
            8. **ProjectionComponent** *1036*
            9. RoleConstraint *1045*
            10. **SubtypeSet** *1070*
            11. **SubtypeSetMembershipCriterion** *1074*
    2. 0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3*
        1. 0:N AccessPath.*Incorporates*.Attribute 0:N *1716*
        2. 0:1 AccessPath.*Instantiates*.Key 0:1 *1700*
        3. 0:N Attribute.*IsDiscriminatorFor*.SubtypeSet-MembershipCriterion 0:N *1701*
        4. 0:N Attribute.*IsInheritedFrom*.Attribute 0:1 *1702*
        5. 0:N Cluster.*Collects*.DataModelObject 0:N *1703*
        6. 0:N ComponentObject.*References*.DefinitionObject 0:1 *8022*
        7. 0:N DataModel.*Collects*.DataModelObject 0:N *1704*
        8. **1**:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N *1705*
        9. 0:N DataModelObject.*IsMemberOf*.DataModelSubset 0:N *1706*
        10. 0:N DataModelSubset.*Excludes*.Attribute 0:N *1707*
        11. 0:N **DataModelSubset**.*IsSubsetOf*.DataModel **1**:1 *1708*
        12. 0:1 DefinitionObject.*Contains*.ComponentObject 0:N *1131*
        13. **1**:1 DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N *1709*
        14. **1**:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N *1712*
        15. **1**:1 Entity.*IsIdentifiedBy*.**Key** 0:N *1711*
        16. 0:N **ForeignKey**.*References*.CandidateKey **1**:1 *1714*
        17. **1**:N InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N *1715*
        18. **1**:1 InheritableDataModelObject.*IsSupertypeFor*.**SubtypeSet** 0:N *1719*
        19. 0:N Key.Incorporates.*SemanticInformationObject* 0:N *1733_Duplicate_2*
            1. 0:N CandidateKey.*Incorporates*.ForeignKey 0:N *1713*
            2. 0:1 ForeignKey.*Incorporates*.RolePlayer 0:1 *1728*
            3. 0:N Key.Incorporates.*Attribute* 0:N *1733*
        20. 0:N ProjectedAttribute.*IsProjectionOf*.Attribute 0:N *63*
        21. 0:N **ProjectionComponent**.*IsFullProjectionOf*.DefinitionObject **1**:1 *1721*
        22. 0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N *1722*
        23. **2**:N **Role**.*BelongsTo*.**Relationship** **1**:1 *1724*
        24. 0:N RoleConstraint.*Incorporates*.SemanticInformationObject 0:N *1727*
            1. 0:N RoleConstraint.*Incorporates*.RoleConstraint 0:N *1725*
            2. 0:N RoleConstraint.*Incorporates*.RolePlayer 0:N *1726*
        25. 0:N RolePlayer.*IsSupportedBy*.Key 0:1 *1729*
        26. **1**:N RolePlayer.*Plays*.**Role** 0:1 *1730*
        27. 0:N RolePlayer.*Refines*.RolePlayer 0:1 *1731*
        28. 0:1 RolePlayer.*RefinesForSubtype*.DataModelObject 0:N *1732*
        29. **1**:1 SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion** 0:N *1736_Duplicate_3*
        30. 0:N **SubtypeSetMembershipCriterion**.*Selects*.-InheritableDataModelObject **1**:1 *1737*

Figure 5: Generalization Hierarchy of the Metamodel „Data Model“.

2.2.2.3 *The EIA/CDIF Metamodel „Data Flow“*

The EIA/CDIF metamodel „Data Flow“ [14] (abbreviated: DFM) defines those *AttributableMetaObjects* which the EIA/CDIF committee believed that are used in most data flow modeling tools. For the purpose of building components the EIA/CDIF „general structuring mechanism“ (abbreviated: GSM) is used which distinguishes between *DefinitionObjects*, *ComponentObjects*, *ReferencedElements* and *EquivalenceSets*. *ReferencedElements* allow for defining paths determining in which sequence *ComponentObjects* are to be built/traversed. An *EquivalenceSet* allows for defining two or more *ComponentObjects* to be (semantically) equivalent. Because the GSM is used extensively in this metamodel, there are relatively few meta-relationships defined for it. Some meta-entities in this metamodel employ multiple inheritance; if a meta-entity belongs to more than one supertype it is depicted in italic in figure 6.

An overview of the most important concepts defined in this metamodel:

- *ExternalAgents* are the source and the sink of *Flows*.
- *ExternalAgentDefinitions* determine the structure of *ExternalAgents* by defining *FlowPorts* through which *Flows* pour.
- *DFMProcesses* work up what comes in through *Flows* and may employ *Stores* for retrieving, changing or storing working means (goods, information). The results of the *DFMProcesses* are transported via *Flows* to other *DFMProcesses* or *ExternalAgents*.
- *DFMProcessDefinitions* determine the structure of *DFMProcesses* which contain *FlowPorts* and which may be used e.g. to pinpoint which component *DFMProcesses* are threadable.

This particular metamodel consists of 84 *CollectableMetaObjects*; specifically: 1 *AttributableMetaObject*, 24 *MetaEntity*, 9 *MetaRelationship* and 50 *MetaAttributes*. A list representation of the generalization hierarchy is given in figure 6.

1. RootObject *1*
    1. RootEntity *2*
        1. SemanticInformationObject *4*
            1. ComponentObject *8000*
                1. Attribute *17*
                2. DFMProcess *6005*
                3. **EquivalenceSet** *1113*
                4. ExternalAgent *6004*
                5. FlowProducerConsumer *232*
                    1. Flow *227*
                    2. *FlowPort* *6015*
                        1. *FlowInputPort* *6001*
                            1. *ConstraintPort* *6012*
                            2. *ControlPort* *6014*
                            3. *SupportPort* *6010*
                        2. *FlowOutputPort* *6000*
                6. Port *1129*
                    1. *FlowPort* *6015*
                        1. *FlowInputPort* *6001*
                            1. *ConstraintPort* *6012*
                            2. *ControlPort* *6014*
                            3. *SupportPort* *6010*
                        2. *FlowOutputPort* *6000*
                7. **ReferencedElement** *8020*
                8. Store *6028*
            2. DataFlowModel *210*
            3. DefinitionObject *8002*
                1. DFMProcessDefinition *213*
                2. ExternalAgentDefinition *6002*
                3. FlowDefinition *221*
                4. StoreDefinition *239*
    2. 0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3*
        1. 0:N ComponentObject.*References*.DefinitionObject 0:1 *8022*
        2. 0:1 DataFlowModel.*HasRoot*.DFMProcessDefinition 0:1 *6026*
        3. 0:1 DefinitionObject.*Contains*.ComponentObject 0:N *1131*
        4. 0:N **EquivalenceSet**.*HasMember*.ComponentObject 2:N *1114*
        5. 0:N FlowProducerConsumer.*ProducesOrConsumes*.Flow 0:N *6023*
            1. 0:N FlowProducerConsumer.*Consumes*.Flow 0:N *6024*
            2. 0:N FlowProducerConsumer.*Produces*.Flow 0:N *6025*
        6. 0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N *8025*

Figure 6: Generalization Hierarchy of the Metamodel „Data Flow“.

2.2.2.4 *The EIA/CDIF Metamodel „Presentation, Location and Connectivity“*

The EIA/CDIF metamodel „Presentation, Location and Connectivity“ [15] (abbreviated: PLAC) defines those *AttributableMetaObjects* which the EIA/CDIF committee believed suitable for exchanging annotated node-edge diagrams depicted in a threedimensional space. It is expected that PLAC models are not instantiated themselves, i.e. no M0 layer will get created for them.[9] This is the first metamodel that employs the concept of a reference to meta-entities, meta-relationships and values of meta-attributes of a given meta-entity or meta-relationship.

An overview of the most important concepts defined in this metamodel:

- A *Diagram* consists of *GraphicalElements*, namely *Nodes* and *Edges* which themselves are placed as *AbsolutePoints* or *RelativePoints* relative to threedimensional *Points*.
- *Edges* connect *Nodes* with each other, whereby they may consist of multiple connected *EdgeElements*.
- *Annotations* may be composed of multiple *AnnotationArguments.*

---

[9] „PLAC models” are instances of the PLAC metamodel and directly represent the diagrams to be exchanged. Therefore there is no practical need to instantiate the „PLAC models” themselves.

This particular metamodel consists of 61 *CollectableMetaObjects*; specifically: 1 *AttributableMetaObject*, 14 *MetaEntity*, 11 *Meta-Relationship* and 35 *MetaAttributes*. A list representation of the generalization hierarchy is given in figure 7.

1. RootObject *1*
    1. RootEntity *2*
        1. PresentationInformationObject *30*
            1. **AnnotationArgument** *10013*
            2. Diagram *10002*
            3. **GraphicalElement** *10004*
                1. Edge *10006*
                2. **EdgeElement** *10007*
                3. **PositionedElement** *10005*
                    1. Annotation *10010*
                    2. Node *10011*
            4. **Point** *10003*
                1. AbsolutePoint *10009*
                2. **RelativePoint** *10008*
        2. SemanticObjectReference *10014*
    2. 0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3*
        1. **1**:1 Annotation.*Uses*.**AnnotationArgument** 0:N *10059*
        2. **1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N *10056*
        3. 0:N Edge.*IsAttachedTo*.GraphicalElement 0:2 *10055*
        4. 0:N **EdgeElement**.*HasEnd*.Point **2**:2 *10057*
        5. 0:N **GraphicalElement**.*AppearsOn*.Diagram **1**:1 *10052*
        6. 0:N Point.*IsDependentUpon*.PositionedElement 0:1 *10060*
        7. 0:N **Point**.*IsLocatedOn*.Diagram **1**:1 *10053*
        8. 0:N **PositionedElement**.*HasCenter*.Point **1**:1 *10054*
        9. 0:N PresentationInformationObject.*Represents*.-SemanticObjectReference 0:N *10051*
        10. 0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1 *10058*

Figure 7: Generalization Hierarchy of the Metamodel „Presentation, Location and Connectivity".

2.2.2.5 The Hypothetical EIA/CDIF Metamodel „Integrated Metamodel"

The hypothetical „Integrated Metamodel" is created by means of a union of all defined MetaObjects, which can be used as a central EIA/CDIF repository, which also may serve as a dictionary for EIA/CDIF based modeling. Among other applications, it may help to ensure that modelers are able to re-use as much of the standardized MetaObjects as possible, either by detecting that needed ones exist already and therefore one merely needs to refer to them, or by specializing existing MetaObjects, thereby re-using all of their definitions (and specializations).

In the work carried out for [23] all standardized EIA/CDIF metamodels, including „Foundation" were extracted from their original text, parsed and transferred to a relational database (Oracle 7.3) for additional analysis. In this project the conceptual „Integrated Metamodel" (abbreviated: IMM) was created and published for the first time in public.

As a result it has become possible to indicate the structure of the IMM metamodel. It consists of a total of 292 *CollectableMeta-Objects*; specifically: 1 *AttributableMetaObject*, 62 *MetaEntity*, 60 *MetaRelationship* and 169 *MetaAttributes*. It may be interesting to anecdotically note that over all EIA/CDIF metamodels practically the same number of meta-entities and meta-relationship have been defined.

2.3 An Example: a Metamodel for Exchanging a Model

This section shall demonstrate how one can define a metamodel with EIA/CDIF and a model based upon it, which both get exchanged using EIA/CDIF's Syntax.1 and clear text encoding Encoding.1 for transferring model data.[10]

As most people may be acquainted with the entity relationship modeling paradigm, a primitive ("bare bone") entity relationship modeling language will get defined as depicted in figure 8 and which will be called "Bare_Bone_ERM". This particular modeling language hence consists of some entity type "EntType" and some relationship type "RelType", which both can be denoted with an attribute called "Name". "RelType"s are binary and they allow for denoting the minimum and maximum number of entities participating in relationships of the given type at both ends, dubbed the "source" and the "target". The arrow connecting the participating entity types with a given relationship type will possess an arrow head pointing at the "target".

1. RootObject *1*
    1. RootEntity *2*
        1. EntType *ME_1*
    2. 0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3*
        1. 0:N EntType.*RelType*.EntType 0:N *MR_1*

Figure 8: Metamodel „Bare_Bone_ERM" (an Example of a Metamodel).

Any EIA/CDIF compliant metamodel one devises must use the EIA/CDIF metamodel "Foundation" as its root as mentioned above, so "EntType" and "RelType" need to be defined as subtypes of "RootEntity" and "IsRelatedTo", respectively.

As is the case with the EIA/CDIF metamodel "Foundation" the types themselves including their attributes need to be created by instantiating the appropriate entity types in the meta-metamodel, "EntType" will be an instance of M3's "MetaEntity", "RelType" of M3's "MetaRelationship", and the "Name" attributes as well as the "RelType" attributes "MinSourceCardinality", "MaxSource-Cardinality", "MinTargetCardinality" and "MaxTarget-Cardinality" will be instances of M3's "MetaAttribute". Instances of the M3 relationship type "IsLocalAttributeOf" will get used to assign the attributes to their types and instances of the M3 relationship type "HasSubtype" will document the fact that "EntType" is a subtype of "RootEntity" and "RelType" is a subtype of "IsRelatedTo". Figure 9 shows the metamodel "Foundation" and how the "Bare_Bone_ERM" metamodel relates to it.

---

[10] This section follows thoroughly the EIA/CDIF standards as defined in [8], [9] and [10]. Specific examples of applying CDIF in the context of simulation and electronic systems can be found e.g. in [3], [4] and [19].

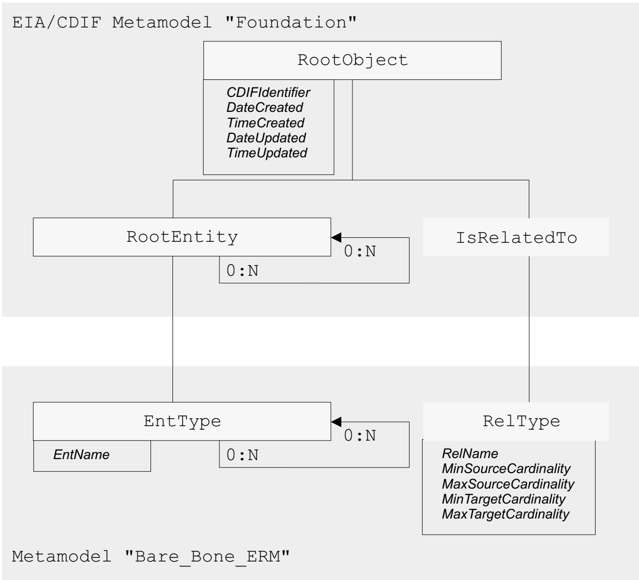

Figure 9: The Metamodel tree with the EIA/CDIF compliant „Bare_Bone_ERM“ Metamodel.

Given a simple entity relationship model expressed with the “Bare_Bone_ERM” depicting a simple automobile special interest group (abbrev.: ASIG), figure 10 represents the following facts about it: “a member may own one or more vehicles” and “a vehicle must be owned exactly by one member”. It follows therefore that there are members allowed to own no vehicle at all.

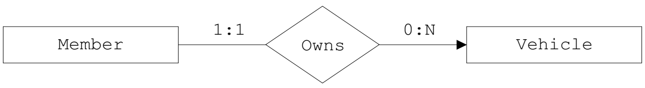

Figure 10: An ASIG-Model Utilizing the „Bare_Bone_ERM“ Metamodel.

Figure 11 shows a transfer of this metamodel with the means of EIA/CDIF’s Syntax.1 [9] and Encoding.1 [10] standard.

```
CDIF,SYNTAX "SYNTAX.1" "02.00.00",ENCODING "ENCODING.1" "02.00.00"


#|  comment: prologue ended, header section starts |#
(:HEADER
    (:SUMMARY
        (ExporterName                                  "ACAT – A
CDIF Aware Tool")
        (ExporterVersion                               "01.21")
        (ExportDate "2001/11/26")
        (ExportTime "00:21:54") ) )

#|  metamodel section |#
(:META-MODEL
#| refer to an EIA/CDIF standardized metamodel |#
    (:SUBJECTAREAREFERENCE Foundation
        (:VERSIONNUMBER "01.00") )

#|  define new metamodel |#
    (SubjectArea SA_1
        (Name *Bare_Bone_ERM*)
        (VersionNumber "01.00")
        (Description #[A Metamodel example.]#) )

#|  define new MetaEntity |#
    (MetaEntity ME_1
        (Name *EntType*)
        (Description #[Represents an entity type.]#) )

#|  MetaEntity is a subtype of "RootEntity" |#
    (   AttributableMetaObject.HasSubtype.
        AttributableMetaObject 2 ME_1)

#|  define new MetaAttribute |#
    (MetaAttribute MA_1
        (Name *Name*)
        (Description #[Stores the name.]#)
        (DataType <String>)
        (Length "64")
        (IsOptional -False-) )

#| assign new MetaAttribute to MetaEntity |#
    (   MetaAttribute.IsLocalMetaAttributeOf.
        AttributableMetaObject MA_1 ME_1)

#|  define new MetaRelationship |#
    (MetaRelationship MR_1
        (Name *RelType*)
        (Description #[Represents a relationship type.]#)
        (MinSourceCard "0")
        (MaxSourceCard "N")
        (MinDestCard "0")
        (MaxDestCard "N") )

#|  MetaRelationship is a subtype of "IsRelatedTo" |#
    (   AttributableMetaObject.HasSubtype.
        AttributableMetaObject 3 MR_1)

#|  MetaRelationship relates two „EntType“s |#
    (MetaRelationship.HasSource.MetaEntity MR_1 ME_1)
    (MetaRelationship.HasDestination.MetaEntity
        MR_1 ME_1)

#| define new MetaAttributes |#
    (MetaAttribute MA_2
        (Name *Name*)
        (Description #[ Stores the name.]#)
        (DataType <String>)
        (Length "64")
        (IsOptional -False-) )

    (MetaAttribute MA_3
        (Name *MinSourceCardinality*)
        (Description
            #[Stores the minimum source cardinality.]# )
        (DataType <String>)
        (Length "10")
        (IsOptional -False-) )

    (MetaAttribute MA_4
        (Name *MaxSourceCardinality*)
        (Description
            #[Stores the maximum source cardinality.]# )
        (DataType <String>)
        (Length "10")
        (IsOptional -False-) )

    (MetaAttribute MA_5
        (Name *MinTargetCardinality*)
        (Description
            #[Stores the minimum target cardinality.]# )
        (DataType <String>)
        (Length "10")
        (IsOptional -False-) )

    (MetaAttribute MA_6
        (Name *MaxTargetCardinality*)
        (Description
            #[ Stores the maximum target cardinality.]# )
        (DataType <String>)
        (Length "10")
        (IsOptional -False-) )

#|  assign new MetaAttributes to MetaRelationship |#

    (MetaAttribute.IsLocalMetaAttributeOf.
        AttributableMetaObject MA_2 MR_1)
    (MetaAttribute.IsLocalMetaAttributeOf.
        AttributableMetaObject MA_3 MR_1)
    (MetaAttribute.IsLocalMetaAttributeOf.
        AttributableMetaObject MA_4 MR_1)
    (MetaAttribute.IsLocalMetaAttributeOf.
        AttributableMetaObject MA_5 MR_1)
    (MetaAttribute.IsLocalMetaAttributeOf.
        AttributableMetaObject MA_6 MR_1)

#|  assign new MetaEntity, MetaRelationship and
        MetaAttributes to the newly defined metamodel |#
    (CollectableMetaObject.IsUsedIn.SubjectArea ME_1 SA_1)
    (CollectableMetaObject.IsUsedIn.SubjectArea MR_1 SA_1)
    (CollectableMetaObject.IsUsedIn.SubjectArea MA_1 SA_1)
    (CollectableMetaObject.IsUsedIn.SubjectArea MA_2 SA_1)
    (CollectableMetaObject.IsUsedIn.SubjectArea MA_3 SA_1)
    (CollectableMetaObject.IsUsedIn.SubjectArea MA_4 SA_1)
    (CollectableMetaObject.IsUsedIn.SubjectArea MA_5 SA_1)
    (CollectableMetaObject.IsUsedIn.SubjectArea MA_6 SA_1)
)
```

```
#| Model section |#
(:MODEL
       (EntType ET_1 (Name
    "Member") )
       (EntType ET_2 (Name
    "Vehicle") )
       (EntType.RelType.EntType R_1 ET_1 ET_2
           (Name                                        "Owns")
           (MinSourceCardinality                        "1")
           (MaxSourceCardinality                        "1")
           (MinTargetCardinality                        "0")
           (MaxTargetCardinality                        "N")
       )
)
```

Figure 11: Defining the „Bare_Bone_ERM" Metamodel and Exchanging the ASIG Model Based Upon it.[11]

3. CONCLUSION

The standardization work of EIA/CDIF [18] was concluded at the end of 1998. All the resulting EIA/CDIF standards are available and can be purchased from EIA [17]. This way the results of the years long work of highly regarded industry experts and the wealth of information contained in these standards remain accessible for the years to come.

HTML renderings of the original EIA/CDIF *AttributableMeta-Object* definitions can be found on the World-Wide-Web at [37].

The baselines (standards) of EIA/CDIF were transferred to ISO/IEC JTC1/SC7 WG11 which has been actively developing an international standard of CDIF, since November 2000 the working group 11 merged with the newly created working group 19, "ODP and Modelling Languages",. Once this work on CDIF is completed, a slightly changed but international set of CDIF standards will evolve.

Many of the EIA/CDIF contributors have been actively working on co-developing OMG standards in the context of the „Meta Object Facility" (abbreviated: MOF, a meta-metamodel, [34] [32]), of the „Unified Modeling Language" (abbreviated: UML, a metamodel, [35]) and the „XML Metadata Interchange" (abbreviated: XMI, a stream based transfer format, [22], [36]). Therefore it may be expected that the EIA/CDIF standards and the experiences gained while developing them are used as input for developing standards for modeling domains in addition to UML. As a matter of fact, if an axiomatic mapping from EIA/CDIF's meta-metamodel to MOF was done, then deriving XMI DTDs for them would be a straight forward process.

In the realm of XML an additional and interesting development could be undertaken by creating a SYNTAX.2 and ENCODING.2 CDIF exchange utilizing the work on „XML Schema" which became a W3C recommendation in May 2001 [38]. It would be possible to define appropriate *XML schemas* for the M3-, M2- and M1-layer and by doing so, taking advantage of all the applications (tools) which have been developed for manipulating XML Schema based data to employ CDIF data exchanges.

Independently of future work on EIA/CDIF, the available set of standards contains a wealth of thoroughly devised definitions, information and concepts, which one can put to work right away, if the exchange of model data is of importance to companies and the academia.

[11] Due to space constraints some words were broken up right after a delimiting point. In such a case the immediately following line would have to be abutted to the trailing dot beginning with the first non space character of the following line.

4. REFERENCES

[1] ANSI: „Information Resource Dictionary System (IRDS)", American National Standard Institute, New York 1989.

[2] Batini C., Ceri S., Navathe S.: „Conceptual Database Design", The Benjamin/Cummings Publishing, Redwood City 1992.

[3] Burst A., Spitzer B., Wolff M., Müller-Glaser K.D.: „On Code Generation for Rapid Prototyping Using CDIF", OOPSLA Workshop #25, „Model Engineering, Methods and Tools Integration with CDIF", Vancouver 1998.

[4] Burst A., Wolff M., Kühl M., Müller-Glaser K.D.: „Using CDIF for Concept-Oriented Rapid Prototyping of Electronic Systems", working paper, Institute for Information Processint Technology (ITIV), University of Karlsruhe 1998: : http://www-itiv.etec.uni-karlsruhe.de/FORSCHUNG/f_rpt/v_rsp98/rsp98.html

[5] ECMA: „Portable Common Tool Environment (PCTE) – Mapping from CASE Data Interchange Format (CDIF) to PCTE", ECMA-270, ECMA 1997. http://www.ecma.ch/stand/Ecma-270.htm

[6] EIA: „CDIF – CASE Data Interchange Format – Overview", Interim Standard, EIA/IS-106, EIA 1994.

[7] EIA: „CDIF – Framework for Modeling and Extensibility", Interim Standard, EIA/IS-107, EIA 1994.

[8] EIA: „CDIF – Transfer Format – General Rules for Syntaxes", Interim Standard, EIA/IS-108, EIA 1994.

[9] EIA: „CDIF – Transfer Format – Transfer Format Syntax – SYNTAX.1", Interim Standard, EIA/IS-109, EIA 1994.

[10] EIA: „CDIF – Transfer Format – Transfer Format Encoding – ENCODING.1", Interim Standard, EIA/IS-110, EIA 1994.

[11] EIA: „CDIF – Integrated Meta-model, Foundation Subject Area", Interim Standard, EIA/IS-111, EIA 1994.

[12] EIA: „CDIF – Integrated Meta-model, Common Subject Area", Interim Standard, EIA/IS-112, EIA 1995.

[13] EIA: „CDIF – Integrated Meta-model, Data Modeling Subject Area", Interim Standard, EIA/IS-114, EIA 1996.

[14] EIA: „CDIF – Integrated Meta-model, Data Flow Subject Area", Interim Standard, EIA/IS-115, EIA 1996.

[15] EIA: „CDIF – Integrated Meta-model, Presentation Location and Connectivity Subject Area", Interim Standard, EIA/IS-118, EIA 1996.

[16] EIA: „CDIF Transfer Format – OMG IDL Bindings", EIA/IS-734, EIA 1997.

[17] EIA: WWW homepage of the „Electronic Industries Alliance (EIA).
http://www.eia.org

[18] EIA: WWW homepage of the „Electronic Industries Association (EIA), CASE Data Interchange Format (CDIF) committee".
http://www.eigroup.org/cdif/index.html

[19] Ernst J.: „Contributions to the Integration of Tools and Techniques for the Development of Heterogeneous Embedded Real-Time Systems“, Forschungsbericht des Forschungszentrums für Informatik (FZI), Karlsruhe 1998.

[20] Flatscher R.G.: „An Overview of the Architecture of EIA's CASE Data Interchange Format (CDIF)“, in: “, in: Rundbrief der Gesellschaft für Informatik – Fachausschuß 5.2 (Informationssystem Architekturen), 3. Jahrgang, Heft 1.

[21] Flatscher R.G.: „Federating Meta-model and Model Data with EIA/CDIF's CORBA Compliant MIDDLEWARE.1“, in: Rundbrief der Gesellschaft für Informatik – Fachausschuß 5.2 (Informationssystem Architekturen), 4. Jahrgang, Heft 1, 1997.

[22] Flatscher R.G.: „Exchange of UML-Models with EIA/CDIF“, in: Schader M., Korthaus A. (Hrsg.): „The Unified Modeling Language – Technical Aspects and Applications“, Physica-Verlag, Heidelberg 1998.

[23] Flatscher R.G.: „Metamodellierung in EIA/CDIF“, ADV Verlag, Wien 1998. http://www.wu-wien.ac.at/wi/rgf/adv/.

[24] ISO/IEC: „Information Technology (IT), Conceptual Modelling Facilities (CSMF)“, Committee Draft SC21 WG3 N2039 for International Standard ISO/IEC 14481, January 1997.

[25] ISO/IEC: „Information Technology – CDIF Framework – Part 1: Overview“, committee draft CD 15474-1, ISO/IEC JTC 1/SC 7 N, 1540R, ISO/IEC 1998.

[26] ISO/IEC: „Information Technology – CDIF Framework – Part 2: Modelling and Extensibility“, committee draft CD 15474-2, ISO/IEC JTC 1/SC 7 N, 1541R, ISO/IEC 1998.

[27] ISO/IEC: „Information Technology – CDIF Transfer format – Part 1: General Rules for Syntaxes and Encodings“, committee draft CD 15475-1, ISO/IEC JTC 1/SC 7 N, 1542R, ISO/IEC 1998.

[28] ISO/IEC: „Information Technology – CDIF Transfer format – Part 2: Syntax SYNTAX.1“, committee draft CD 15475-2, ISO/IEC JTC 1/SC 7 N, 1543R, ISO/IEC 1998.

[29] ISO/IEC: „Information Technology – CDIF Transfer format – Part 3: Encoding ENCODING.1“, committee draft CD 15475-3, ISO/IEC JTC 1/SC 7 N, 1544R, ISO/IEC 1998.

[30] ISO/IEC: „Information Technology – CDIF Semantic Metamodel – Part 1: Foundation“, committee draft CD 15476-1, ISO/IEC JTC 1/SC 7 N, 1545R, ISO/IEC 1998.

[31] ISO/IEC: „Information Technology – CDIF Semantic Metamodel – Part 2: Common“, committee draft CD 15476-2, ISO/IEC JTC 1/SC 7 N, 1546R, ISO/IEC 1998.

[32] ISO/IEC: „Information Technology – CDIF Semantic Metamodel – Part 4: Data Models“, committee draft CD 15476-4, ISO/IEC JTC 1/SC 7 N, 1548R, ISO/IEC 1998.

[33] ISO/IEC: „Information Technology – CDIF Semantic Metamodel – Part 5: Data Flow Models“, committee draft CD 15476-5, ISO/IEC JTC 1/SC 7 N, 1549R, ISO/IEC 1998.

[34] OMG: „Meta Object Facility (MOF) – Specification“, version 1.0, OMG standard. ftp://ftp.omg.org/pub/docs/ad/97-10-02.pdf

[35] OMG: „Unified Modeling Language (UML)“, version 1.1, OMG standard. http://www.omg.org/techprocess/meetings/schedule/ Technology_Adoptions.htm#UML_Specification“.

[36] OMG: „XML Metadata Interchange (XMI)“, version 1.0, OMG standard. http://www.omg.org/cgi-bin/doc?ad/98-10-05.pdf

[37] WU Wien: HTML-renderings of ***all*** of the EIA/CDIF metamodels (standards as well as drafts), derived directly from the electronic versions of the standards and drafts. http://wwwi.wu-wien.ac.at/cdif

[38] W3C: „XML Schema“, version 1.0, W3C recommendation: http://www.w3c.org/XML/Schema