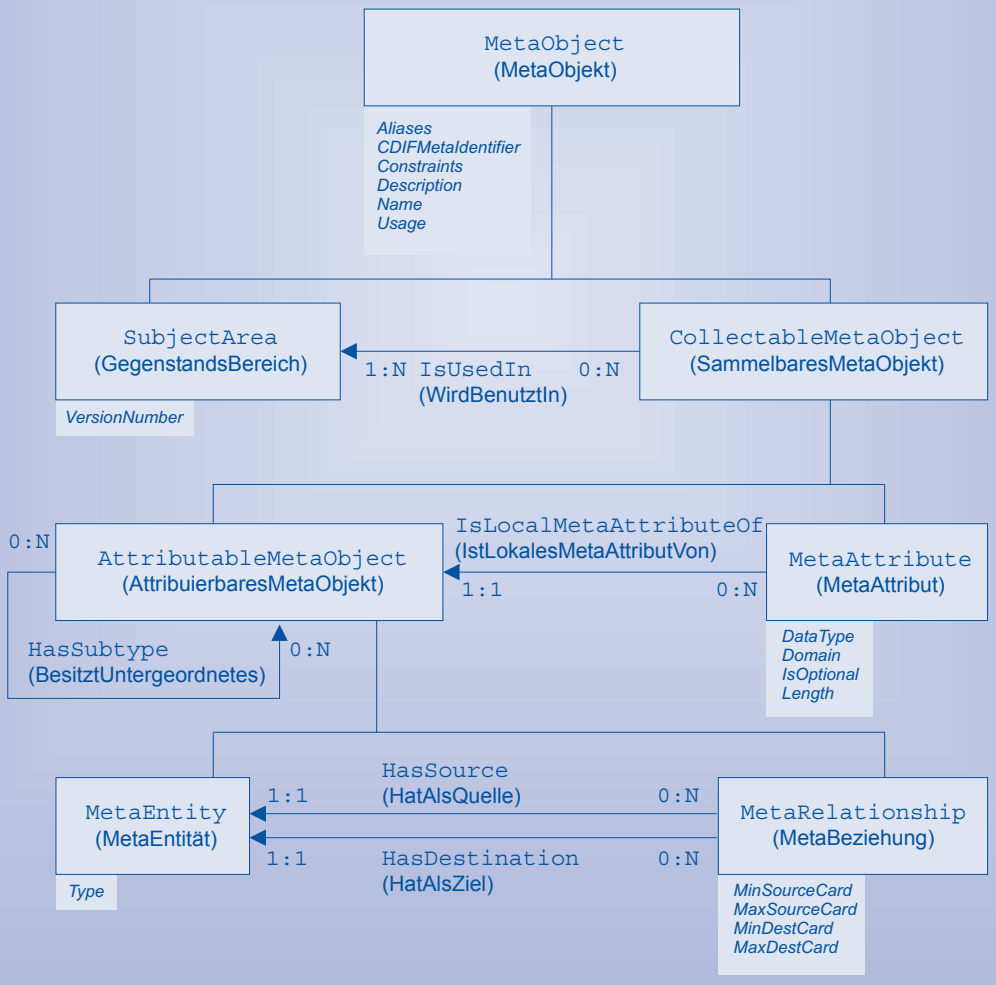
Meta-Modellierung in EIA/CDIF

Rony G. Flatscher



ADV

Rony G. Flatscher

Meta-Modellierung in EIA/CDIF

Mit 28 Abbildungen,

mit 276 Tabellen (davon 123 Tabellen für die vollständige Darstellung des „Integrierten EIA/CDIF-Meta-Modells“),

mit 14 Programmcode-Auflistungen,

mit 12 SQL-Tabellenspezifikationen,

mit 7 SQL-Viewspezifikationen,

mit einem Abkürzungsverzeichnis und

mit einem kleinen Wörterbuch für die Übersetzung der wichtigsten Begriffe (Englisch/Deutsch und Deutsch/Englisch).

ADV

ADV-Verlag

Trattnerhof 2

A-1010 Wien/Vienna

Mag. Dr. Rony Georges Flatscher
Institut für Informationsverarbeitung und Informationswirtschaft
Abteilung für Wirtschaftsinformatik (http://www**i**.wu-wien.ac.at)

Wirtschaftsuniversität Wien (http://www.wu-wien.ac.at)

Augasse 2-6
A-1090 Wien

Österreich/Austria

ISBN 3-901198-09-1, ADV-Verlag, Wien: 1998.

Medieninhaber: ADV Handelsgesellschaft mbH (http://www.adv.at)
Trattnerhof 2
A-1010 Wien/Vienna

Telefon: (01) 533 09 13
Telefax: (01) 533 09 13 77

Hersteller: Druckerei Riegelnik Ges.m.b.H.
Piaristeng. 17-19
A-1080 Wien

Verlags- und Herstellungsort: Wien

Vorwort

Die vorliegende Arbeit setzt sich folgende Ziele:

- erstmals den seit 1987 entwickelten EIA/CDIF-Standard systematisch und umfassend in zusammenhängender Form darzustellen, mit entsprechenden Quellen zu verknüpfen und damit einer wissenschaftlichen Diskussion zuzuführen,
- die Detailliertheit der Darstellung der EIA/CDIF-Standards so zu wählen, daß wissenschaftliche Einrichtungen und Unternehmen imstande sind, EIA/CDIF-Meta-Modelle selbständig zu erstellen beziehungsweise zu evaluieren,
- erstmals das hypothetische „Integrierte EIA/CDIF-Meta-Modell“ auf Basis der per 1. Jänner 1998 standardisierten EIA/CDIF-Meta-Modelle zu erstellen und im Rahmen dieser Arbeit zu dokumentieren; es wurde dabei Wert darauf gelegt, daß dieser Abschnitt auch als Referenz für sämtliche standardisierten EIA/CDIF-Meta-Objekte, also Meta-Entitätstypen, Meta-Beziehungstypen und ihre Meta-Attribute dient und somit für alle, die sich mit EIA/CDIF und ISO/CDIF auseinandersetzen, eine wertvolle Zusatzhilfe darstellt,
- erstmals Auswertungen über sämtliche bis 1. Jänner 1998 standardisierte EIA/CDIF-Meta-Modelle so aufzubereiten, daß sie einerseits Einsichten in die Struktur geben und andererseits auch für weitere Arbeiten und Auswertungen weiterverwendbar sind,
- erstmals ein EIA/CDIF-konformes Meta-Modell zu definieren, das demonstrieren soll, wie man strukturelle Informationen auf der Meta-Modellebene ohne Änderung am Meta-Meta-Modell erfassen und auswerten kann,
- Spezifikationen des EIA/CDIF-Standards für die Verteilung von EIA/CDIF-Meta-Modelldaten und Modelldaten mit Hilfe von OMG’s CORBA beziehungsweise der OMG IDL so einzuarbeiten, daß sie im Kontext der entsprechenden Modellierungsschichten dargestellt und verstanden werden können,

– erstmals Spezifikationen für relationale Datenbankverwaltungssysteme für die Abbildung des EIA/CDIF-Meta-Meta-Modells zu entwerfen und öffentlich zu dokumentieren, um damit

  - die Grundlagen für eine ingenieurmäßige Erstellung, Verwaltung und Pflege von Meta-Modelldefinitionen für EIA/CDIF zu erarbeiten, um von der seit 1987 bis heute gegenwärtigen, fehleranfälligen textverarbeitungsbezogenen Verarbeitung wegzukommen,
  - die Struktur von Meta-Modellen abzulegen und sie einer Auswertung für spezialisierte Werkzeuge bis hin zu direkten SQL-Anweisungen zugänglich zu machen,

– erstmals Spezifikationen für die objektorientierte und interpretierte Programmiersprache Object Rexx für die Abbildung des EIA/CDIF-Meta-Meta-Modells zu geben, um damit die Repräsentation der Konzepte direkt in Form von Klassenhierarchien und den zu den Klassen gehörenden Attributmethoden zu ermöglichen,

– sowohl auf aktuelle als auch auf mittelfristige Weiterentwicklungen des EIA/CDIF-Standards sowie auf Standards hinzuweisen, die auf EIA/CDIF bezug nehmen, insbesondere werden folgende Entwicklungslinien thematisiert:

  - weitere EIA/CDIF-Entwürfe von Meta-Modellen,
  - STEP/EXPRESS-Abbildungen von EIA/CDIF,
  - die Adaptierung von EIA/CDIF in einen „ISO/CDIF“-Standard und
  - die Entwicklungen rund um OMG’s „SMIF“ RFP, mit dessen Hilfe UML-Modelldaten beziehungsweise allgemeiner, MOF-basierte Meta-Modelldaten und Modelldaten tauschen zu können, wobei als Ausgangspunkt EIA/CDIF und STEP/EXPRESS vorausgesetzt werden,

– beispielhafte Implementierungen (im Anhang) für relationale Datenbankverwaltungssysteme inklusive der Erstellung von Triggern für das konsistente Löschen von Meta-Objekten aus der Datenbank auf Basis der erarbeiteten Spezifikationen vorzunehmen und zu dokumentieren, wobei dafür

als Zielsystem das relationale Datenbankverwaltungssystem der Firma ORACLE eingesetzt wird,

- beispielhafte Implementierungen (im Anhang) in Object Rexx darzustellen und zu dokumentieren, mit denen EIA/CDIF-Meta-Modelldefinitionen in relationale Datenbanken übertragen werden können, wobei hier die relationale Datenbank ORACLE als Zielsystem fungiert,
- erstmals Kreuztabellen (im Anhang) zu verfassen, mit deren Hilfe sämtliche EIA/CDIF-Meta-Objektdefinitionen aller mit Anfang 1998 standardisierter EIA/CDIF-Meta-Modelle anhand ihrer Bezeichner und ihrer Surrogate (Wert des Meta-Meta-Attributs „CDIFMetaIdentifier“) schnell aufgefunden werden können.

Dieses Buch nutzt drei verschiedene Darstellungs- beziehungsweise Strukturmittel:

1. Zunächst wird im Haupttext der Kern der Arbeit umgesetzt, wobei dafür die Begriffe der EIA/CDIF-Standards benutzt und häufig auch die deutschen Übersetzungen dafür verwendet werden. Im Anhang ist deshalb ein kleines Lexikon der entsprechenden englischen und deutschen Übersetzungen eingerichtet.

   So verwirrend manchmal der Einsatz der Begriffe erscheinen mag, so wird in dieser Arbeit versucht, sie genauso wie in den originalen EIA/CDIF-Standards zu verwenden (bis hin zur schriftlichen Darstellung der Begriffe), sodaß nach der Durcharbeitung dieses Buches die Beschäftigung mit den Originalstandards jedenfalls kein begriffliches Problem mehr darstellen sollte.

2. Fußnoten werden in dieser Arbeit nicht nur für Quellenverweise benutzt, sondern auch für Diskussionen, deren Dokumentation im Haupttext nach Meinung des Autors von der Vermittlung der EIA/CDIF-Standards zu stark ablenken würde.

   Es sollte daher möglich sein, den Haupttext ohne Zuhilfenahme der Fußnoten zu lesen und zu verstehen. Wann immer Zusammenhänge oder Hintergründe während des Lesens interessant erscheinen, wird auf die entsprechenden Fußnoten verwiesen.

3. Der Anhang bietet zunächst das übliche Quellenverzeichnis, Verzeichnis der Abbildungen und Tabellen, aber auch Verzeichnisse über Programmcode und SQL-Anweisungen. Darüber hinaus finden sich Abschnitte über Implementierungen des EIA/CDIF-Meta-Meta-Modells.
   Die Abschnitte über die Implementierung des EIA/CDIF-Meta-Meta-Modells in der relationalen Datenbankverwaltungssoftware ORACLE sowie die Implementierung in der objektorientierten, interpretierten Programmiersprache Object Rexx sind in sich abgeschlossen.
   Für das Verstehen der ORACLE-Implementierung sind grundlegende Kenntnisse über SQL und Triggers sowie fortgeschrittene Kenntnisse in ORACLE's PL*SQL für die Programmierung von gespeicherten Prozeduren notwendig.
   Für das Verständnis von Object Rexx sind allgemeine grundlegende Kenntnisse des objektorientierten Programmierparadigmas nach Smalltalk und C++ hilfreich, die Syntax der Sprache selbst ist sehr einfach und damit benutzerfreundlich, sodaß zum Teil Programmabschnitte wie Pseudocode wirken mögen (der Nachrichtenoperator in Object Rexx ist die Tilde '~').

Weitere Lesehinweise:

- Für die meisten, der in dieser Arbeit verwendeten, Abkürzungen wurden Einträge im Abkürzungsverzeichnis vorgenommen. Es befindet sich im Anhang auf Seite 493ff.
- Beim erstmaligen Auftreten von EIA/CDIF-Begriffen werden die deutschen Übersetzungen angegeben und in weiterer Folge synonym verwendet. Zudem wurden die Übersetzungen in ein entsprechendes Wörterbuch aufgenommen, das sich im Anhang findet (Englisch/Deutsch auf Seite 497ff und Deutsch/Englisch auf Seite 502ff).

Hinter einer derartigen Arbeit stehen neben viel Forschungsarbeit, Ausdauer und Geduld, viele weitere Väter und Mütter, die direkt oder indirekt wesentlich zum Gelingen beigetragen haben und denen daher mein Dank gilt:

- zuvorderst Professor Hans Robert Hansen, der dem Verfasser wirklich in vielen Belangen ein Vorbild ist und ganz wesentlich das Zustandekommen ermöglicht hat, dann die Kolleginnen und Kollegen an der Abteilung für

Wirtschaftsinformatik, die mit Zuspruch und auch mit „banalen“ Arbeiten wie dem wichtigen Korrekturlesen mitgeholfen haben,

– Woody Pidcock, dem Präsidenten von EIA/CDIF und dem „Convenor“ von ISO/IETC JTC1/SC7 mit der Arbeitsgruppe 11, die in jahrelanger Arbeit EIAs CDIF bis zum Herbst 1998 zu einem ISO-Standard macht, und

– Herrn Johannes Ernst, der als „EIA/CDIF-Technical Officer“ als hervorragender Fachmann immer wieder bei Fragen und Problemen per E-Mail und Diskussionslisten zur Verfügung gestanden hat.

Der Autor möchte sich besonders für die jahrelange Geduld und das jahrelange Verständnis seiner Frau herzlich bedanken, ohne die die Familie wohl zerbrochen wäre (oftmalige 12- bis 16-stündige Arbeitstage sieben Tage die Woche belasten immens).

Wien, im Juni 1998

Inhaltsverzeichnis – Übersicht

Inhaltsverzeichnis

1 Problemstellung

In den Forschungsvorhaben der Wirtschaftsinformatik, die sich als Wissenschaft zur Gestaltung rechnergestützter Informationssysteme in der Wirtschaft versteht[1], ist die methodische Erzeugung von Modellen[2] von grundlegender Bedeutung. Unter den vielen verschiedenen Methoden der Modellierung spielt die konzeptionelle Datenmodellierung mit Hilfe der auf [Chen76] zurückführbaren Entity-Relationship-Modellierung (abgekürzt: „ERM“) eine hervorragende Rolle, die daher auch aus keinem Lehrbuch zur Wirtschaftsinformatik wegzudenken ist.[3]

1.1 Konzeptionelle Modellierung mit der ERM-Methode

In der konzeptionellen Modellierung mit der ERM-Methode werden Entitäten (englisch: „entity“), das heißt voneinander unterscheidbare konkrete oder abstrakte Dinge, und die Beziehungen (englisch: „relationship“) zwischen ihnen aus einer zuvor abgegrenzten Diskurswelt[4] („Gegenstandsbereich“) mit Hilfe der Abstraktion verallgemeinert und gemeinsam mit Eigenschaften in Form von Attributen in Typen festgehalten. Sämtliche Entitätstypen und Beziehungstypen stellen als Ergebnis die Intension der Diskurswelt dar, die einzelnen Dinge und einzelnen Beziehungen die dazugehörende Extension.

---

1 Vgl. hierzu die Definition in [Hans96], Seite 86 oben, sowie die Ausführungen über den interdisziplinären Charakter zwischen Betriebswirtschaftslehre und Informatik. Diese erfolgten in Übereinstimmung mit der Charta der Mitglieder der Wissenschaftlichen Kommission Wirtschaftsinformatik im Verband der Hochschullehrer für Betriebswirtschaft e.V., wie sie mittlerweile auch in [W3WI] veröffentlicht ist. Untersuchungsgegenstand sind dementsprechend „Informations- und Kommunikationssysteme (IKS) in Wirtschaft und Verwaltung“, die soziotechnische Systeme darstellen, die ihrer Natur nach komplex sind.

2 Dies läßt sich auch anhand einer Analyse der Lehrbücher der Wirtschaftsinformatik erkennen, exemplarisch sei hier etwa [FerSin98] genannt.
Die hervorragende Bedeutung der Modellierung für die Wirtschaftsinformatik läßt sich auch anhand von [LeMaHi95] ermessen, das in der Darstellung der theoretischen Grundlagen der Wirtschaftsinformatik über 90 Seiten – fast ein Viertel des Buches – ausschließlich dem Begriff „Modell“ und der „Modellierung“, also der Erstellung von Modellen, widmet.

3 Beispielsweise wird die Methode der Entity-Relationship-Modellierung in folgenden (exemplarisch angeführten) wirtschaftsinformatikbezogenen Lehrbüchern im deutschen Sprachraum vermittelt: [BiMuRu97], [FerSin98], [Jank93], [Hans96], [Hild95], [LeAuBa91], [MeBoKö95], [Öste95a], [Sche95], [StaHas97].

4 Vgl. z.B. [LeMaHi95], Seite 289 oben, [FerSin98], Seiten 4ff und 103ff.

Durch die Erweiterung der grundlegenden Methode um die Möglichkeit, Generalisierungshierarchien von Entitätstypen zu definieren, konnte die Erstellung von semantischen Datenmodellen weiter verfeinert werden.[5] In der graphischen Darstellung in Form von Knoten- und Kantendiagrammen werden Entitätstypen üblicherweise als Knoten und Beziehungstypen sowie die hierarchischen Über- und Unterordnungen der Entitätstypen in erweiterten ERM als Kanten repräsentiert.

[BaCeNa92] führen folgende qualitativen Eigenschaften für konzeptionelle Modelle im allgemeinen[6] und für ERM[7] im besonderen an:

– Ausdruckskraft (englisch: „expressiveness“): ist sehr hoch aufgrund der zur Verfügung stehenden Abstraktionsmechanismen zur Klassifikation[8] (Typbildung), Aggregation[9] (Eigenschaftsbildung) und Verallgemeinerung[10] (Generalisierungshierarchie).

– Einfachheit (englisch: „simplicity“): ein konzeptionelles Modell soll einfach sein, damit es leicht verständlich ist und als Grundlage für unterschiedliche Anwendungen dienen kann. Dem gegenüber steht die Ausdruckskraft, die wie im Falle von ERM hoch ist und daher dazu führen kann, daß damit erstellte konzeptionelle Modelle nicht einfach verständlich sind.[11]

---

5 Diese Arbeit bezieht sich auf die Darstellung in [ElmNav89] und [BaCeNa92].

6 Vgl. [BaCeNa92], Seiten 29 bis 30.

7 Vgl. a.a.O, Seiten 47 bis 48.

8 Es werden Klassen von Entitäten, Beziehungen und Attributen unterschieden, die zur Bildung von Entitätstypen, Beziehungstypen und zu Attributstypen herangezogen werden.

9 In diesem Kontext wird eine nähere Beschreibung beziehungsweise sogar die Bildung von Entitäten, Beziehungen und zusammengesetzten Attributen durch Attribute verstanden. Versteht man unter Eigenschaften auch Verhalten in Form von Funktionen/Methoden, mit oder ohne Vor- und Nachbedingungen, geht die Modellierung in eine objektorientierte über.

10 Im englischen wird von „generalization“ gesprochen und beinhaltet üblicherweise die Bildung eines hierarchischen Klassifikationsbaumes von Entitätstypen, der die Eigenschaft aufweist, daß untergeordnete Typen sämtliche Eigenschaften der übergeordneten über das Konzept der Vererbung erhalten. Zugleich weisen die Entitätstypen beim Durchschreiten Richtung Wurzeltyp auf jeder Ebene verallgemeinerte Eigenschaften auf. Dies führt in der Regel zu Wurzeltypen die omnipotent in dem Sinne sind, daß sie *beliebige* Eigenschaften annehmen können beziehungsweise in beliebiger Art und Weise spezialisierbar sind.

11 Dies trifft vor allem für ERM-Varianten zu, die die Modellierung mit Hilfe von nichtbinären (mehrwertigen) Beziehungstypen zulassen oder auch Beziehungstypen zwischen Beziehungstypen selbst („komplexe Beziehungstypen“) ermöglichen.

– Minimalität (englisch: „minimality“): kein Konzept soll durch andere Konzepte beziehungsweise durch eine Kombination von anderen Konzepten ausdrückbar sein. Im ERM sind die Konzepte „Entität“, „Beziehung“, „Attribut“, und im EERM die „Generalisierungshierarchie“ minimal.

– Formalität (englisch: „formality“): damit ein konzeptionelles Modell beziehungsweise ein ERM formal bearbeitet werden kann, müssen sämtliche Konzepte, die für die Bildung der Modelle benutzt werden, eine einzigartige, präzise und wohldefinierte Bedeutung haben.

Aus ER-Modellen lassen sich unter anderem unterschiedliche Datenbankschemata direkt mit Hilfe von einfachen Regeln ableiten, die auf hierarchische, netzwerkorientierte oder die in der betriebswirtschaftlichen Umwelt dominierenden relationalen Datenbankverwaltungssysteme abgestellt sind.[12]

Die weltweite hohe Akzeptanz der ERM-Methode läßt sich auch daran bemessen, daß es eine regelmäßig organisierte, internationale Konferenz[13] für neuere Entwicklungen um diese Methode gibt, sowie aus der Tatsache ableiten, daß im Rahmen der internationalen Standardisierungsorganisation ISO/IEC eine Arbeitsgruppe die grundlegenden Begriffe und Konzepte der konzeptionellen Datenmodellierung[14] zu standardisieren versucht hat.[15, 16]

---

12 Vgl. [ElmNav89], Kapitel 15, „Advanced Data Modeling Concepts“, wo mit Hilfe der semantischen Datenmodellierung auch die Ableitungsregeln angeführt sind. In diesem Zusammenhang sind auch die Arbeiten in [Neum94] interessant, wo einerseits entsprechende Abbildungen zwischen (E)ERM und NIAM mit Hilfe von PROLOG definiert werden und andererseits ein entsprechendes, interessantes Quellenverzeichnis gefunden werden kann.

13 Die erste Konferenz fand 1979 in Los Angeles statt, bis 1985 wurde sie im Zweijahresrhythmus abgehalten, seitdem im jährlichen Abstand. So wird 1998 die mittlerweile 17. Konferenz im Herbst in Singapur unter der Bezeichnung „17th International Conference on Conceptual Modeling (ER'98)“ abgehalten.

14 Einen Überblick unter anderem über verschiedenen Datenmodellierungssprachen findet sich in [Hars94].

15 Ein interessanter und lesenswerter Entwurf davon findet sich in [ISO97a].

16 Vgl. in diesem Zusammenhang beispielsweise auch die Diskussion über objektorientierte Modellierungssprachen in [Fran98], in der auch auf EIA/CDIF Bezug genommen wird.

1.1.1 Erstellung konzeptioneller Modelle mit ERM in der Wirtschaftsinformatik des deutschsprachigen Raumes

In den vergangenen Jahren wurden im Kontext der Wirtschaftsinformatik zahlreiche Modelle mit Hilfe der ERM-Methode verfaßt, wobei die verschiedenen Modellierer voneinander leicht unterschiedliche Varianten der Methode nutzen. Im folgenden findet sich eine Auswahl von drei bekannteren Ansätzen in der deutschsprachigen Wirtschaftsinformatik:

– Ferstl/Sinz gehen in [FerSin98] zunächst vom erweiterten ERM[17] aus, das sie dann in das strukturierte ERM[18] (abgekürzt: „SERM“) weiterentwickeln, wobei Beziehungstypen gleichzeitig als Entitätstypen darstellbar sind.[19] Diese erarbeiteten Konzepte können in weiterer Folge in der von beiden Autoren entwickelten Modellierungsmethode SOM[20] (Akronym für: „Semantisches Objektmodell“) eingesetzt werden,

– Österle beschreibt in [Öste95a] für die in St. Gallen entwickelte Methode „PROMET“[21] eine erweiterte Form von ERM für die konzeptionelle Datenmodellierung, die bereits in früheren[22] Arbeiten aus St. Gallen eingesetzt wird.

– Scheer benutzt in [Sche91] und [Sche95] eine erweiterte ERM-Variante, in der Beziehungstypen gleichzeitig als Entitätstypen darstellbar sind,[23] wo-

17 Vgl. [FerSin98], Seite 127ff, aber auch [BaCeNa92], Kapitel 2, „Data Modeling Concepts“, Seite 15ff.

18 Vgl. a.a.O., Seite 142ff.

19 Das „Camouflieren“ von Beziehungstypen als „assoziative“ Entitätstypen läßt sich bereits bei [Codd79] finden und wird bis heute angewandt. Somit können komplexe Beziehungstypen in der Modellierung eingesetzt werden, indem Entitätstypen zu als assoziative Entitätstypen camouflierten Beziehungstypen in Beziehung setzbar sind.

20 Vgl. dazu den Übersichtsartikel in [FerSin95] sowie die detaillierte Einführung in SOM in [FerSin98], Seite 176ff.

21 Diese Methode wurde im Rahmen der St. Gallener Kompetenzzentren „Prozeßentwicklung“ (Kurzbezeichnung: „PRO“) und „Rechnergestütztes Informationsmanagement“ (Kurzbezeichnung: „RIM“) entwickelt und bezieht sich in der Ausprägung „PROMET BPR“ auf die Geschäftsprozeßmodellierung.

22 Vgl. z.B. [ÖstGut92a].

23 Es ist in diesem Zusammenhang bemerkenswert, daß beispielsweise in [Sche98b] begonnen wird, statt der konzeptionellen Modelle mit Hilfe von EERM Klassendiagramme aus UML (vgl. hierzu [UML97c], [UML97d] und [UML97e]) einzusetzen, ohne daß es dafür einen zwingenden Grund gäbe. Im Vorwort von [Sche98b] bemerkt Scheer dazu:

Fortsetzung folgt auf der nächsten Seite.

bei in [Sche91] als Beilage ein etwa 60cm mal 90cm großes Plakat beigefügt ist, das das konzeptionelle Datenmodell für ARIS (Akronym für: „Architektur integrierter Informationssysteme“) selbst dokumentiert.[24]

Ein wichtiges Ziel in der Wirtschaftsinformatik ist es, mit Hilfe der konzeptionellen Datenmodellierung zu einem unternehmensweiten Datenmodell, dem sogenannten Unternehmensdatenmodell[25] (abgekürzt: „UDM“) zu kommen, das idealerweise sämtliche Anwendungsdatenmodelle integriert[26]. Neben vielen Aspekten, die damit verbunden sind, kommen häufig auch grundlegende Überlegungen in bezug auf erwartete Nutzenpotentiale hinzu, wie sie in der folgenden Abbildung 1-1 aus [Hars94] generell für die konzeptionelle Datenmodellierung prägnant dargestellt sind, getrennt nach der Verwendung für den „Entwurf von DV-Systemen“ beziehungsweise für das „Informationsmanagement“.[27]

„Wegen der zu erwartenden Standardisierung von UML werden die ARIS-Metamodelle als Klassendiagramme nach UML dargestellt. Inhaltlich ergeben sich daraus aber keine Änderungen zu der Darstellung als Entity Relationship Model (ERM) der ersten zwei Auflagen.“

24 Damit liegt wie weiter unten noch begründet wird, bereits ein umfangreiches Meta-Modell in Form eines konzeptionellen ERM vor.

25 Vgl. beispielsweise die Ausführungen dazu in [Sche90], in [LeMaHi95] oder prägnant in [LeAuBa91]. Es findet sich auch als wichtiges Schlagwort im Lexikon der Wirtschaftsinformatik wieder (vgl. [Sche97]).
In [Maie98] findet sich eine auf [Maie96] zurückgehende Auswertung, die besagt, daß 41 Prozent der antwortenden Unternehmen im deutschsprachigen Raum UDMs führen beziehungsweise entwickeln, wobei Dienstleistungsunternehmen signifikant mehr als Industrieunternehmen UDM erstellen.

26 Darin liegt auch einer der wesentlichen Vorteile von großen standardisierten, betriebswirtschaftlichen Anwendungssystemen, wie sie von Firmen wie Baan oder SAP mit großem Erfolg vertrieben werden, sofern die Handhabung des gesamten Datenmodells beherrscht wird.

27 Im Rahmen dieser Arbeit wird beispielsweise der Nutzen von Meta-Modellen hauptsächlich in der Klärungsfunktion und in der Dokumentationsfunktion im Rahmen des Informationsmanagements gesehen.

	Entwurf von DV-Systemen	Informationsmanagement (Nutzung des Produktionsfaktors Information)
Klärungs-funktion	- Anforderungsspezifikationen - Integration heterogener Systeme - Reengineering	- Begriffserklärung - Informationsbedarfsanalyse
Dokumentations-funktion	- Dokumentation der DV-Systeme - Kommunikation mit dem Anwender	- Globales Informationsverzeichnis - Zugriffsmedium - Begriffsdokumentation (Organisationshandbuch)
Gestaltungs-funktion	- Projektabgrenzung - Schnittstellendefinition - Trennung fachlicher und DV-technischer Aspekte (Wiederverwendbarkeit) - Generieren von Datendefinitionen - Auswahl von Software - Konfiguration	- Schwachstellenanalyse - Informationsbewertung - Verantwortlichkeit für Information - Reintegration von Funktionen

Abbildung 1-1: Nutzenpotentiale von konzeptionellen Datenmodellen[28]

1.1.2 (Meta-)Meta-Modellierung: Erstellung konzeptioneller (ERM) Modelle für Modellierungsmethoden

Die im vorhergehenden Abschnitt angeführten deutschsprachigen Wirtschaftsinformatiker haben ihre Varianten der ERM dazu benutzt, ihre eigenen Modellierungsmethoden konzeptionell zu modellieren. Damit wird es möglich, die grundlegenden Konzepte, mit deren Hilfe Modelle gebildet werden, selbst zu formalisieren.[29]

---

28 Entnommen aus [Hars94], Seite 29, im Kontext des Abschnittes 2.3, „Nutzen semantischer Datenmodelle“.

29 Nachdem mit Hilfe von (E)ERM konzeptionelle Modelle gebildet werden, die festlegen, welche syntaktischen Elemente („Modellierungsbausteine“) zulässig sind und wie diese zueinander angeordnet werden dürfen (Beziehungen zwischen den Modellbausteinen), wird in diesem Zusammenhang auch von „Sprachen“ beziehungsweise „Modellierungssprachen“ gesprochen. Vgl. in diesem Zusammenhang auch die „Unified Modeling Language“ (abgekürzt: „UML“) in [UML97a] bis [UML97e], sowie [W3UML] als Ausgangspunkt weiterer Recherchen.

So definieren Ferstl/Sinz in [FerSin98] unter anderem die SOM-Methode für die Erstellung von Geschäftsprozeßmodellen[30] formal mit Hilfe eines konzeptionellen ERM. Österle/Gutzwiller spezifizieren in [ÖstGut92a] formal ein „Referenz-Metamodell für die Analyse und das System-Design“[31] mit ERM, zudem finden sich in [ÖsBrHi92] eine Reihe von Meta-Modellen zur Dokumentation der vorgeschlagenen IS-Architektur[32]. Scheer hat bereits, wie weiter oben erwähnt, in [Sche91] die ARIS-Architektur formal in Form von Meta-Modellen mit ERM dokumentiert und veröffentlichte 1998 ein Buch ([Sche98b]) mit dem Begriff „Meta-Modell“ im Titel: „ARIS – Modellierungsmethoden, Metamodelle, Anwendungen“[33].

Wenn mit Hilfe von Meta-Modellen Modellierungsmethodologien konzeptionell und formal mit ERM modelliert werden können, so liegt es nahe, daß man Meta-Modelle auch dazu heranzieht, unterschiedliche Modellierungsmethodologien vergleichbar zu machen. Dies versucht beispielsweise Hess in [Hess96], indem er vierzehn[34] verschiedene Methoden zum Entwurf betrieblicher Prozesse analysiert und mit Hilfe von Meta-Modellen in ERM formal dokumentiert.

Voraussetzung für das Verständnis derartiger Meta-Modelle ist einerseits ein einheitliches Meta-Meta-Modell, das die Modellbauelemente und Beziehungen zwischen ihnen für die Erstellung von Meta-Modellen festlegt, sowie eine kon-

---

30 Vgl. [FerSin98], Seite 194 bis 195.

31 In [ÖsGuSp94] werden die aus [ÖstGut92a] entnommenen Meta-Modelle für die „Analyse“ und das „System-Design“ dargestellt und erlautert. [ÖstGut92a] definiert unter anderem weitere Meta-Modelle in ERM für die Funktionssicht, Kommunikationssicht (beinhaltet Datenflußsicht), Datenverwendungssicht, Datensicht sowie Verhaltenssicht. In [ÖstGut92b] wird mit Hilfe eines umfangreichen Beispiels gezeigt, wie die Modellierung entsprechend den Meta-Modellen von [ÖstGut92a] erfolgt.

32 Vgl. in diesem Zusammenhang beispielsweise in [ÖsBrHi92] auf Seite 156 das Meta-Modell in Form eines ERM zur Beschreibung der Modellbausteine und Beziehungen untereinander für den Gegenstandsbereich („Teilsicht“) „Organisation“.

33 Wie eingangs erwähnt, benutzt Scheer beginnend mit [Sche98b] für die Darstellung des konzeptionellen Datenmodells die Notation von UML-Klassendiagrammen statt wie in früheren Arbeiten eine ERM-Notation. Die Diagramme werden als Folge davon grundsätzlich umfangreicher, die Kardinalitäten werden entsprechend der Gegenüberstellung in [Sche98b], Seite 6, getauscht und die übliche Notation für Beziehungstypen (auf dem Kopf stehende Raute) durch ein Klassen-Rechteck ersetzt, das selbst von der entsprechenden Assoziation durch strichlierte Linien abgesetzt ist .Der Buchstabe „n“, der bei Scheer’s ERM für „viele“ in Kardinalitäten steht, wird durch das Sternzeichen („*“) ersetzt.

34 Es sind dies die Methoden von „Action Inc.“, „Boston Consulting Group“, „Davenport“, „Diebold Deutschland GmbH“, „Eversheim“, „Ferstl/Sinz“, „Hammer“, „Harrington“, „IBM Unternehmensberatung GmbH“, „Johansson“, „Malone“, „McKinsey & Company“, „Ploenzke AG“ und „Scheer“.

sistente Benutzung von Begriffen für Meta-Entitätstypen[35] und Meta-Beziehungstypen über alle Meta-Modell hinweg. In der Regel wird als Meta-Meta-Modell eine bestimmte Variante von ERM benutzt, die sich aber strukturell und semantisch voneinander unterscheiden können:[36] bei Ferstl/Sinz kommt deren ERM-Methode zum Einsatz[37], im Falle von [Hess96] kommt die St. Gallener ERM-Methode zum Einsatz, im Falle von [Sche91] die Scheer'sche Ausprägung der ERM-Methode[38].

1.2 CASE-Systeme und Meta-Modelle

Meta-Modelle wurden bereits sehr früh im Rahmen von CASE[39]-Systemen eingesetzt, da sie die (zulässige) Struktur für die Erstellung von Modellen dokumentieren und festlegen. Wenn derartige Meta-Modelle in Form von konzeptionellen ER-Modellen vorliegen, liegt es nahe, daraus auch mit den üblichen Regeln[40] die physischen Datenstrukturen abzuleiten. Unter anderem können die Daten, die erstellte Modelle repräsentieren, bei Verwendung von relationalen Datenbankverwaltungssystemen in den entsprechend den Meta-Modellen abgeleiteten Tabellen gespeichert werden.

---

35 So kann etwa bei [Hess96] davon ausgegangen werden, daß die Benennung eines Meta-Entitätstyps mit „Process" semantisch über alle Meta-Modelle hinweg in etwa dasselbe Konzept bezeichnet.

36 So ermöglicht das Meta-Meta-Modell von Ferstl/Sinz (vgl. [FerSin98], Seite 122 oben) eine Verfeinerung von Beziehungstypen in „is_a" (Generalisierungsbeziehung, Symbol: Dreieck), „has" (Aggregationsbeziehung, „Attribut-Zuordnungsbeziehung", Symbol: strichlierte Linie) und „connects" (Assoziationsbeziehung, Symbol: durchgehende Linie). Das Meta-Meta-Modell ist reflexiv und wird daher selbst mit der in ihr definierten Notation dargestellt.

37 Vgl. beispielsweise das Meta-Modell für die Geschäftsprozeßmodellierung nach dem Ferstl/Sinz-SOM-Ansatz in [FerSin98], Seite 195 oben, das ebenda auf Seite 122 definierte Meta-Meta-Modell zur Konstruktion benutzt.

38 In [Sche98b], Seiten 120 bis 125, wird auf die Meta-Meta-Modellebene eingegangen, wobei es in Abbildung 67 ebenda auf Seite 123 aus zwei Klassen („Objekttyp" und „Anwendungsobjekt") und zwei Assoziationen besteht. Das Abstratkionsniveau erscheint so hoch, daß es aufgrund der Darstellung nicht ableitbar ist, aus welchen Modellbausteinen Meta-Modelle tatsächlich bestehen dürfen. Es ist bemerkenswert, daß [Sche98b] für den Begriff Meta-Meta-Modell die Bezeichnung „Meta^2-Modell" verwendet.

39 Vgl. in diesem Zusammenhang auch die Definitionen und Ausführungen in [Hans96], Seiten 858 bis 861. Beispielsweise zählt das Scheer'sche ARIS-Toolset hierbei zu den Upper-CASE-Systemen.

40 Vgl. beispielsweise [ElmNav89], Abschnitt 15.2, „Mapping EER Model Concepts", Seite 427ff.

1.2.1 Meta-CASE-Systeme

Da Meta-Modelle die zur Verfügung stehenden Modellbausteine und die Beziehungsregeln zwischen diesen formal festlegen, liegt es nahe, zunächst mit Hilfe eines Meta-Meta-Modells die Erstellung von Meta-Modellen zu standardisieren. Unter Kenntnis des Meta-Meta-Modells wird es möglich, die Meta-Modelldefinitionen auszuwerten, unabhängig davon für welche Modellierungsmethode sie erstellt sind. In weiterer Folge könnten dann CASE-Systeme geschaffen werden, die als Modelleditoren für beliebige Methoden einsetzbar sind. In so einem Fall spricht man von „Meta-CASE-Systemen“. [41]

Im deutschen Sprachraum ist beispielsweise das Projekt „KOGGE“ am Institut für Softwaretechnik an der Universität Koblenz-Landau in diesem Zusammenhang zu sehen und seit 1994 dokumentiert[42]. In Finnland wird mit internationaler Beteiligung am „MetaPHOR“-Projekt[43] gearbeitet, das relativ gut dokumentiert ist und aus dem auch ein kommerzielles Produkt entstanden ist, das sich „MetaEdit+“[44] nennt.[45]

---

41 Folgt man den Ausführungen in [Sche98b], Seiten 120 bis 125, so läßt sich das ARIS-Toolset auch als Meta-CASE-System verstehen.

42 Ein Überblick und eine Einführung wird in [CarEbe94] gegeben, weitere interessante Veröffentlichungen finden sich unter anderem in [EbSuUh97] sowie [Ebe97]. Ein Überblick über dieses Projekt (und gleichzeitig ein Ausgangspunkt für weitere Recherchen) findet sich unter [W3KOGGE].
Die graphische Komponente GRAL wird in [Fran97] in Form eines Referenzhandbuchs dokumentiert (vgl. zu GRAL auch [EbWiDa96a] und in einer ausführlicheren Variante in [EbWiDa96b]). In GRAL werden gerichtete, typisierbare Graphen („TGraphen“) verarbeitet, wobei auf Z-basierende Spezifikationen (vgl. [Spiv88]) dafür angegeben werden können. Graphklassen werden in diesem Zusammenhang mit EERM-Diagrammen spezifiziert. Ein Überblick über dieses Projekt (und zugleich ein Ausgangspunkt für weitere Recherchen) findet sich unter [W3GRAL].

43 In [LyyKer94] findet sich in Form eines Abschlußberichtes ein Überblick über das MetaPHOR-Projekt. Für die Implementierung wurde ein eigenes Meta-Meta-Modell entwickelt, das als GOPPR, „Graph-Object-Property-Role-Relationship“ bezeichnet wird. Ein Überblick über dieses Projekt (und gleichzeitig ein Ausgangspunkt für weitere Recherchen über aktuelle Forschungspläne) findet sich unter [W3METAPHOR].

44 Vgl. in diesem Zusammenhang [KeLyRo96], sowie die im WWW verfügbare Beschreibung in [W3METAEDIT] mit Verweisen auf entsprechende, weiterführende Literatur sowie zur Vertriebsfirma.

45 [W3CASETOOLS] stellt eine interessante, wenngleich unvollständige Übersicht über CASE-Tools dar, die zum überwiegenden Teil mit Meta-Modellen arbeiten. Diese Quelle empfiehlt sich als ein Ausgangspunkt zur weiteren Recherche von Meta-CASE-Systemen.

1.2.2 Austausch von Modelldaten

Sofern Unternehmen und Organisationen mit Hilfe von CASE-Systemen Datenflußmodelle, Datenmodelle, Kommunikationsmodelle etc. erstellen, werden sie in der einen oder anderen Form als Modelldaten gespeichert. Mit Hilfe dieser Modelldaten werden die einzelnen Modelle vollständig dokumentiert. Nachdem der Modellierungsvorgang selbst ein sehr kostenintensiver Vorgang sein kann, der entsprechende Ressourcen verbraucht, sollen einmal erstellte Modelle solange wie möglich weiterbenutzbar bleiben.[46] Durch den natürlichen Fortschritt in einem technologiegetriebenen und daher einem hohen Innovationsdruck ausgesetzten Fach wie der Wirtschaftsinformatik ist es zu erwarten, daß CASE-Werkzeuge beziehungsweise CASE-Systeme selbst einer entsprechend schnellen Veralterung ausgesetzt sind, sodaß sie in relativ kurzen Abständen durch neuere oder andere Werkzeuge und Systeme abgelöst werden.

Eine äußerst wichtige Frage, die es in diesem Kontext zu lösen gilt, lautet: „Wie können bereits (und möglicherweise sehr kostenintensiv) erstellte Modelle für verschiedene betriebswirtschaftliche Informationssysteme weiterverwendet werden, damit die bereits erfolgten Investititonen nicht wieder von neuem (für die Wiedererstellung der Modelle in einer neuen computerunterstützten Modellierungsumgebung) getätigt werden müssen?“ Des weiteren kann es unabhängig davon sinnvoll sein, daß durch Investititonen in spezialisierte CASE-Werkzeuge beziehungsweise CASE-Tools begründet, von Fremdsystemen[47] aus der Zugriff auf die Modelldaten möglich ist. So ist es vorstellbar, daß für ein Geschäftsprozeßmodellierungswerkzeug der Zugriff auf ein mit einem auf die Aufbauorganisation spezialisierten Werkzeug bereits erstelltes Aufbaumodell

---

46 Hier werden daher erstellte Modelle als Kapital für Unternehmen und Organisationen im Zusammenhang mit der Wartung und Weiterentwicklung von bestehenden beziehungsweise der strategischen Planung im Kontext von betriebswirtschaftlichen Informationssystemen angesehen.

47 Unter „Fremdsystemen“ werden hier beliebige, weitere CASE-Werkzeuge verstanden, für die bereits erstellten Modelldaten aus technischen Gründen nicht zugänglich sind. Dies kann durch unbekannte Datenbankverwaltungssysteme, undokumentierte dateibezogene Speicherformate bis hin zu Formaten und Strukturen, deren Elemente in der Bedeutung nicht zweifelsfrei bekannt sind, begründet sein.

eines Unternehmens zurückgegriffen werden soll, um dann entsprechend die Ablaufmodelle in Beziehung damit zu setzen.[48]

Nachdem die Konstruktion von Modellen ressourcenaufwendig ist, wäre es allein aus Kostenüberlegungen wünschenswert, bereits erzeugte Modelldaten zwischen den unterschiedlichen CASE-Systemen und CASE-Werkzeugen tauschen zu können. Dazu ist es aber notwendig, daß sich die Hersteller dieser unterschiedlichen Werkzeuge zumindest auf folgendes einigen:

- ein Austauschformat, das die Syntax und die Verkodierung der Daten beschreibt,
- Meta-Modelle, die unabhängig von den einzelnen CASE-Systemen und CASE-Werkzeugen die Konzepte beschreiben, die für die unterschiedlichen Austäusche grundlegend zur Verfügung stehen; beispielsweise muß im Falle des Austausches von Datenflußmodellen entsprechend ein akkordiertes Meta-Modell der Datenflußmodellierung definiert werden, das alle Hersteller gleichermaßen zum Austausch benutzen,[49]
- der Austausch von Modelldaten erfolgt so, daß die entsprechenden Konzepte des Meta-Modells[50] instanziiert werden,
- für den Zweck eines Austausches kann es notwendig sein, daß die Meta-Modelle für den Austausch von Modelldaten nach einheitlichen Regeln er-

---

48 Im Rahmen der Wirtschaftsinformatik zählen auch computerunterstützte Werkzeuge zur Aufbau- und Ablaufmodellierung zu den CASE-Werkzeugen im weiteren Sinn. Dies läßt sich auch anhand von CASE-Systemen wie ORACLE's „CASE Designer/2000" erkennen, wo für die (Geschäfts-) Prozeßmodellierung entsprechende Stellen definiert werden können, für die Teile eines zu konzipierenden, betriebswirtschaftlichen Informationssystems gedacht sind beziehungsweise damit unterstützt werden.

49 Damit können Meta-Modelle, die für einen bestimmten Gegenstandsbereich sämtliche erlaubten Konzepte festlegen und in ihrer Bedeutung beschreiben, nicht nur als Taxonomien, sondern auch als Ontologien interpretiert werden, mit deren Hilfe voneinander unabhängige Hersteller von CASE-Tools die Konzepte der zu tauschenden Modelldaten einheitlich in das Meta-Modell einordnen können. Vgl. in diesem Zusammenhang auch die Rolle von Ontologien in der Unternehmensmodellierung in [Vern96], Seiten 110 bis 111, sowie die Ausführungen und Diskussionen zu Ontologien allgemein, ebenda, Abschnitt „Ontologies", Seiten 376 bis 378.

50 Im Beispiel der Datenflußmodellierung wird ein Meta-Modell das Konzept „Fluß" aufweisen, sodaß für den Austausch konkreter Flüsse Instanzen gebildet werden, die die Struktur des auf Meta-Modellebene definierten Konzeptes „Fluß" aufweisen.

weiterbar sind, sodaß in weiterer Folge Importeure die Austauschdaten fehlerlos einlesen und bearbeiten können.[51]

Darüber hinaus ist es weiterhin wünschenswert, wenn Modelldaten auch mit Hilfe von CORBA[52] verteilbar wären, sodaß sie in vernetzten Systemen von unterschiedlichen CASE-Systemen und CASE-Werkzeugen aus benutzbar sind. Eine entsprechend allgemein gültige Schnittstellendefinition, die für alle möglichen Meta-Modelle gilt, setzt voraus, daß auch für die Konstruktion der Meta-Modelle ein Modell definiert wird, das sämtliche Modellbausteine und die Beziehungen untereinander dafür festlegt[53]. Meta-Modelle können dann als Instanzen des Meta-Meta-Modells aufgefaßt werden und dadurch selbst verteilbar sein.[54]

1.3 Electronic Industries Alliance's „CASE Data Interchange Format (EIA/CDIF)"

Das „CASE Data Interchange Format" (abgekürzt: „CDIF"[55]) von der amerikanischen "Electronic Industries Alliance"[56] (abgekürzt: "EIA") ist ein seit 1987 kontinuierlich entwickelter Industriestandard zum Austausch von Modelldaten.

1.3.1 Die Electronic Industries Alliance

EIA ist die amerikanische Interessensvertretung und Handelsorganisation der elektronischen Industrie. Sie gliedert sich schwerpunktmäßig in die Sektoren „Komponenten", „Konsumentenelektronik", „Elektronische Informationstechnik", „Elektronikindustrie", „Regierung" und „Telekommunikation", wofür eigenständi-

---

51 Nach einem erfolgreichen Einlesen dürfen die importierenden Werkzeuge jene Modelldaten verwerfen, deren Konzepte ihnen unbekannt sind.

52 Vgl. hierzu beispielsweise die Definitionen des entsprechenden Standards von OMG in [CORBA98], sowie die Ausführungen über Middleware im allgemeinen in [ÖsRiVo96].

53 Entsprechend handelt es sich um ein Meta-Meta-Modell, vgl. auch [LeMaHi95] im Abschnitt „Information auf der Meta-Meta-Ebene", Seiten 225 und 226, sowie [FerSin98] im Abschnitt „Meta-Ebenen von Modellsystemen", auf den Seiten 121 und 122.

54 Dieser Ansatz wurde beispielsweise von EIA/CDIF in [CDIF97b] gewählt.

55 Es gibt Initiativen, das Akronym umzuinterpretieren, beispielsweise in „Common Data Interchange Facility", oder gar nicht mehr auszuschreiben, sondern als eigenständigen Begriff für sich selbst zu belassen, es also für ein eigenständiges Konzept anzusehen, das nicht übersetzt zu werden braucht. Damit soll erreicht werden, daß mit dem Begriff „CDIF" unbefaßte Personen nicht schließen, daß die EIA/CDIF-Standards ausschließlich für CASE-Werkzeuge und CASE-Standards einsetzbar sind. Vgl. in diesem Zusammenhang auch [W3CDIF].

56 Bis Ende 1997 lautete die Bezeichnung: „Electronic Industries Association", vgl. [W3EIA].

ge Gruppen innerhalb von EIA geschaffen werden. EIA verfügt über Marktforschungseinrichtungen als auch Standardisierungsabteilungen, die mit ANSI und auch ISO/IETC zusammenarbeiten.

Die Gruppe "Elektronische Informationstechnik" (englisch: "Electronic Information Group", abgekürzt: "EIG") wurde 1993 gegründet, um die informationstechnischen Initiativen und Arbeiten von EIA innerhalb einer eigenständigen Organisation zu verwalten. Unter anderem findet sich hier die "CASE Data Interchange Format Division" (abgekürzt: "CDIF"), die seit 1987 an der Entwicklung von unterschiedlichen Austauschformaten für Modelldaten arbeitet. Die EIA/CDIF-Standards wurden in der Vergangenheit unter anderem von Firmen wie Boeing, Ford, Fujitsu, IBM, Lucent, ORACLE, Platinum und Rational mitentwickelt.[57]

Ein wesentliches Bestreben bei der Schaffung von EIA-Industriestandards ist, sowohl Hersteller als auch potentielle Anwender von informationstechnischen Standards für die Entwicklung der Standards selbst heranzuziehen. Dadurch soll eine hohe Akzeptanz der EIA-Industriestandards erreicht werden, die durch institutionalisierte Zusammenarbeit mit den Standardisierungsorganisationen der amerikanischen ANSI und der internationalen ISO/IETC noch verstärkt werden soll.

1.3.2 EIA/CDIF

Das „CASE Data Interchange Format“ umfaßt systematisch über Jahre hinweg definierte Standards für ein Meta-Meta-Modell[58], Meta-Modelle für verschiedene Gegenstandsbereiche[59], ein Austauschformat mit Syntax- und Verkodierungsdefinitionen[60], sowie einen Standard zur Verteilung[61] des Meta-Meta-Modells

---

57 Eine Liste der aktiven Mitglieder bei der Entwicklung von EIA/CDIF findet sich unter [W3CDIF_MA], eine Liste von ehemals aktiven Mitgliedern unter [W3CDIF_ME].

58 Vgl. [CDIF94b]: „Framework for Modeling and Extensibility“.

59 Vgl. [CDIF94f]: „Foundation“, [CDIF95]: „Common“, [CDIF96b]: „Data Modeling“, [CDIF96c]: „Data Flow Modeling“, [CDIF96d]: „Presentation Location and Connectivity“, sowie Entwürfe mit unterschiedlich hoch ausgearbeiteten Detaillierungsgrad wie [CDIF96a]: „State/Event Modeling“, [CDIF96e]: „Business Process Modeling“, [CDIF96f]: „Object-oriented Analysis and Design Core“, [CDIF96g]: „Project Management Planning And Scheduling“, [CDIF96h]: „Data Definition“, [CDIF98]: „Computer Aided Control Systems Design“.

60 Vgl. [CDIF94c]: „General Rules for Syntaxes“, [CDIF94d]: „Transfer Format Syntax – SYNTAX.1“ sowie [CDIF94e]: „Transfer Format Encoding – ENCODING.1“.

und die davon abgeleiteten Meta-Modelle, die als Instanzen die zu verteilenden (auszutauschenden) Modelldaten besitzen.

Es ist interessant zu beobachten, daß dieser Satz amerikanischer Industriestandards, der seit über 10 Jahren kontinuierlich entwickelt wird, zwar in zahlreichen Werkzeugen[62] Eingang gefunden hat, aber auf akademischen Boden, von wenigen Ausnahmen[63] abgesehen, bis dato so gut wie unbekannt und damit auch unerforscht ist! Hierfür mögen folgende Gründe besonders verantwortlich sein:

– Die EIA/CDIF-Standards sind urheberrechtlich geschützt und ein Zutritt dazu ist bis dato nur möglich, wenn man sie direkt bei EIA/CDIF kauft.[64]

– Die einzelnen EIA/CDIF-Standards sind in einer so prägnanten Art und Weise verfaßt, daß man sie nur mit außergewöhnlich hohem Aufwand inhaltlich erarbeiten kann.[65]

---

61 Vgl. [CDIF97b]: „OMG IDL Bindings".

62 Beispielsweise hat die Firma ORACLE (siehe [W3ORACLE]) bereits 1992 (vgl. [ORAC92]) das Data-Dictionary des ORACLE-CASE-Systems als EIA/CDIF-Austausch aufbereitet und verlegt, und bietet unter anderem auch über Dritte die Möglichkeit an, mit dem ORACLE-CASE-System „Designer/2000" Modelldaten über EIA/CDIF zu tauschen. In [ORAC97] stellt ORACLE einen Vergleich ihres CASE-Repositories mit dem von Microsoft an und führt unter anderem im Abschnitt „Repository" als wichtiges Abgrenzungs- und Alleinstellungsmerkmal an, daß „The Oracle Repository is open to a wide variety of existing CASE vendors through the CDIF (*C*ommon *D*ate *I*nterchange *F*acility) standard and automated exchange tools provided by Oracle.".
Ein weiteres Beispiel dafür ist das CASE-System „Paradigm Plus" (siehe [W3PARADIGM]) der Firma Platinum Technology (siehe [W3PLATINUM]), die EIA/CDIF für den Austausch von Modelldaten einsetzt.

63 Hierzu zählt beispielsweise das „Forschungszentrum Informatik" der Universität Karlsruhe (siehe [W3FZI]), das auch EIA/CDIF erforscht. [W3CASETOOLS] beinhaltet einen Verweis auf EIA/CDIF erst seit 1997, sodaß davon ausgegangen werden kann, daß die Mitarbeiter am MetaPHOR-Projekt während der Erarbeitung ihrer eigenen Grundlagen nicht auf EIA/CDIF-Arbeitsergebnisse und Definitionen zurückgreifen konnten.

64 Nachdem die EIA/CDIF-Standards Mitte 1998 auf akademischen Boden noch immer so gut wie unbekannt sind, kommt ein Kauf vielleicht der sprichwörtlichen „Katze im Sack" gleich.
Informationen zum Kauf der EIA/CDIF-Standards finden sich unter [W3CDIF_STANDARDS]. Ab 1998 werden die Standards vorzugsweise in elektronischer Form vertrieben und zwar als Adobe PDF-Dateien, sodaß man selbst Ausdrucke in der Zahl davon produzieren darf, die entsprechend der gekauften Lizenz erlaubt sind. Es ist zu erwarten, daß neue EIA/CDIF-Standards nur mehr in elektronischer Form erworben werden können.

65 In diesem Zusammenhang sei auf eine E-Mail von einem in die Vereinigten Staaten emigrierten Deutschen, Dr. Johannes Ernst, verwiesen, der als technischer Beauftragter (englisch: „technical officer") an EIA/CDIF seit Jahren mitarbeitet und über ein ausgezeichnetes und umfassendes Wissen über die EIA/CDIF-Standards verfügt (vgl. beispielsweise [Ernst98]). Er schrieb am 1996-02-05 über die EIA/CDIF Diskussionsliste

Fortsetzung folgt auf der nächsten Seite.

Die folgenden Unterabschnitte sollen zeigen, welche wichtige Rolle die EIA/CDIF-Standards seit Jahren im Rahmen von weiteren Standardisierungsbemühungen spielen. Im Umkehrschluß bedeutet dies nichts anderes, als daß Standardisierungsinstitutionen wie ECMA, ISO/IETC oder OMG auf die Ergebnisse der EIA/CDIF-Arbeiten zurückgreifen und sich damit zunutze machen.

1.3.2.1 EIA/CDIF, ECMA-149 und ECMA-270

Die 1961 gegründete „European Computer Manufacturers Association“[66] (abgekürzt: „ECMA“) setzt sich folgende Ziele:

- „To develop, in co-operation with the appropriate National, European and International organizations Standards and Technical Reports in order to facilitate and standardize the use of ICT systems.
- To encourage the correct use of Standards by influencing the environment in which they are applied.
- To promulgate the various Standards that it produces.

To this end, all ECMA Standards and Technical Reports are made available free of charge to all interested parties without restriction.“ [67]

„cdif-general@cdif.org“ (siehe [W3CDIF]) unter dem Betreff „*Re: Rational, Metadata Coalition, and Standardization ?!?*“ unter anderem:

> *„... There is one aspect which is often forgotten by long-term-CDIFans, but maybe crucial:*
> *CDIF is a considerably large and complex matter. To understand a meaningful part of it, considerable time is necessary on the part of the reader. (In my case it took me about one year to get a reasonable understanding about the whole thing -- although far less might be necessary to, say, implement a CDIF-interface). ...“.*

Gründe dafür sind unter anderem: fehlende Erläuterungen und Quellenverweise im Kontext der Definitionen; Aufteilung von zusammengehörenden Definitionen auf verschiedene Standards ([CDIF94c], [CDIF94d] und [CDIF94e] für den Syntax- und Verkodierungsteil, zusätzlich sind für das Verständnis [CDIF94a] und [CDIF94b] Voraussetzung), sodaß der Zusammenhang im Detail verloren gehen kann (beispielsweise muß ein „MetaObjectName“ vom EIA/CDIF-Datentyp „Identifier“ sein, darf aber als zusätzliche Bedingung keine Ziffern aufweisen); fehlende oder zuwenig Verweise auf weitere EIA/CDIF-Standards, die im Kontext eines bestimmten Standards eine Rolle spielen; fehlende integrierte Darstellung der EIA/CDIF OMG IDL-Standards zum EIA/CDIF-Meta-Meta-Modell und zum EIA/CDIF-Meta-Modell „Foundation“ (erklärlich einerseits dadurch, daß [CDIF97b] drei Jahre nach [CDIF94b] und [CDIF94f] verabschiedet wurde, andererseits ist es auch sinnvoll, die einzelnen Standards nach Schwerpunkten getrennt zu formulieren).

66 Siehe [W3ECMA].

67 Entnommen aus URL (Stand: 1998-06-01): „http://www.ecma.ch/memento/aims.htm“.

Seit der Gründung von ECMA[68] gibt es eine institutionalisierte Zusammenarbeit mit ISO/IEC, die ursprünglich über das technische Komitee ECMA-TC97 realisiert wurde. 1987 ging dann das Komitee ECMA-TC97 in ISO/IEC JTC 1 auf.

Unter den zahlreichen Standards[69] findet sich in der vierten Auflage auch [ECMA-149], der die Bezeichnung „Portable Common Tool Environment“ (abgekürzt: „PCTE“) trägt und folgenden Zweck erfüllen soll:[70] „It specifies the interface supported by any conforming implementation as a set of abstract operation specifications, together with the types of their parameters and results.“. Unter anderem werden für den Austausch von Daten sogenannte „Schema Definition Sets“ (abgekürzt: „SDS“) vorgesehen. Die Standardisierungsarbeiten von ECMA-PCTE wurden in eine ISO/IETC-Arbeitsgruppe, der ISO/IETC JTC 1/SC 22 WG 22, übernommen, um daraus einen internationalen Standard zu erarbeiten.

Nach einer Präsentation[71] von EIA/CDIF im Jahre 1991 für das technische Komitee ECMA-TC33 wurden Arbeiten initialisiert, die es zum Ziel hatten, Abbildungungen sämtlicher EIA/CDIF-Meta-Modelle auf PCTE-SDS durchzuführen. 1996 erfuhr auf Initiative von Fujitsu und ICL das Projekt einen neuen Schwung, der dazu führte, daß bis Dezember 1997 der Standard „ECMA-270“ verstärkt entwickelt und schließlich erfolgreich verabschiedet wurde.

In ECMA-270 werden die Regeln festgelegt, nach denen die standardisierten EIA/CDIF-Meta-Modelle in ECMA-149-SDS übergeführt werden können. Zusätzlich sind auch sämtliche EIA/CDIF-konformen Meta-Modelle – auch solche, die EIA/CDIF-konforme Erweiterungen an den Meta-Modellen vornehmen – nach diesen Regeln in PCTE-SDS überführbar. Im Anhang von ECMA-270 finden sich die PCTE-SDS für die standardisierten EIA/CDIF-Meta-Modelle „Foundation“, „Common“, „DataDefinition“ und „DataModelling“.

---

68 Vgl. zu diesen Ausführungen auch den URL (Stand: 1998-06-01): „http://www.ecma.ch/memento/history.htm“.

69 Entsprechend dem URL (Stand: 1998-06-01) „http://www.ecma.ch/memento/preface.htm“ wurden bis Mitte 1998 226 ECMA Standards und 69 Technische Berichte (englisch: „Technical Reports“) verfaßt.

70 Entnommen aus [ECMA-149] oben.

71 Vgl. zu den Ausführungen über das Entstehen des ECMA-270 Standards, den Abschnitt „Brief History“ am Anfang des Standards selbst, der über [ECMA-270] aus dem Internet frei bezogen werden kann.

In einer Kurzbeschreibung der wesentlichen Eigenschaften des ECMA-270-Standards finden sich folgende Aussagen dazu:

„Application of this mapping to a CDIF subject area generates a derived PCTE SDS that is semantically equivalent to the CDIF subject area. Such derived SDSs provide

- a means of exchanging models between CASE tools and a PCTE implementation;
- a means of realising models defined according to the corresponding CDIF subject areas in a PCTE installation;
- hence a basis for standard SDSs for systems engineering subject areas.“[72]

Allerdings werden in bezug auf ECMA-149 folgende Einschränkungen angeführt:

„The derived SDSs are not sufficient

- to define all the properties needed for efficient use of the models within a PCTE installation;

for the faithful transfer of the models between different PCTE installations.“ [73]

Die Arbeiten an diesem Standard wurden auch mit der Unterstützung von ISO/IETC JTC 1/SC 22 WG 22 und ISO/IEC JTC 1/SC 7 WG 11 durchgeführt.

1.3.2.2 EIA/CDIF und SEDDI (ISO/IEC JTC 1/SC 7 WG 11)

1992 mündeten verschiedene Initiativen zur Gründung des ISO[74]/IETC Projektes 7.28, „Software Engineering Data Description and Interchange“ (abgekürzt: „SEDDI“), das es sich zum Ziel setzt, die unterschiedlichen Initiativen und Aktivitäten zu vereinheitlichen, die auf die Beschreibung und den Austausch von Modelldaten gerichtet sind. Als wichtiger Auslöser für dieses Projekt gilt die

---

72 Entnommen aus [ECMA-270] unten.

73 Vgl. ebenda.

74 Die internationale Standardisierungsorganisation (abgekürzt: „ISO“) wurde 1947 als internationale, regierungsunabhängige Organisation gegründet. Die WWW-Homepage findet sich unter [W3ISO] und erlaubt unter anderem das Recherchieren von aktuellen Standardisierungsinitiativen.

weiter oben erwähnte Präsentation von EIA/CDIF 1991 für das technische Komitee ECMA-TC33. Das Projekt 7.28, das ISO/IEC JTC 1/SC 7 WG 11, „Information technology/Software engineering“, übertragen wurde, entwickelt aus EIA/CDIF einen internationalen Standard.

Gleichzeitig damit wurde mit EIA/CDIF eine Zusammenarbeit begonnen, mit dem Ziel, daß weitere Standardisierungen von EIA/CDIF mit ISO/IEC JTC 1/SC 7 WG 11 koordiniert werden.[75] Parallel dazu werden die EIA/CDIF-Standards übernommen, überarbeitet und dem üblichen Abstimmungsprozeß von ISO/IETC unterzogen. Als Ergebnis davon wird aus dem amerikanisch basierten, ein Satz von internationalen Standards erarbeitet.[76] Das Projekt 7.28, „Software Engineering Data Description and Interchange (SEDDI)“, setzt sich mit Mitte 1998[77] aus folgenden Teilprojekten zusammen:

– Projekt 7.28.01, „Overview & Architecture“,[78]

– Projekt 7.28.02, „Interchange Formats“, [79]

– Projekt 7.28.03, „Abstract Model“[80] und

– Projekt 7.28.04.02, „PCTE Schema Definition Sets“[81].

---

75 Diese Zusammenarbeit führte beispielsweise dazu, daß die aktuell gültigen, die grundlegende Architektur EIA/CDIFs festlegenden Standards, 1994 mit ISO/IEC JTC 1/SC 7 WG 11 akkordiert wurden.

76 Nachdem üblicherweise nationale Standardisierungsorganisationen Mitglied der ISO/IEC sind, können einerseits Standards darüber bezogen werden, andererseits wird es dadurch für Interessierte möglich, als nationaler Vertreter daran aktiv mitzuarbeiten.

77 Vgl. hierzu die Struktur der Projekte von ISO/IEC JTC 1/SC 7 WG 11, wie sie unter folgendem URL (Stand: 1998-06-01) aufgelistet werden: „http://saturne.info.uqam.ca/Labo_Recherche/Lrgl/sc7/wg11.html“.

78 Die 1998 zur Abstimmung gelangten Standardisierungsentwürfe entsprechen den Unterprojekten „7.28.01.01 – Overview“ in [ISOCDIF98a] sowie „7.28.01.02 – Framework“ in [ISOCDIF98b].

79 Die 1998 zur Abstimmung gelangten Standardisierungsentwürfe entsprechen den Unterprojekten „7.28.02.01 – General Rules“ in [ISOCDIF98c], „7.28.02.02 – Syntax“ in [ISOCDIF98d] sowie „7.28.02.03 – Encoding“ in [ISOCDIF98e].

80 Die 1998 zur Abstimmung gelangten Standardisierungsentwürfe entsprechen den Unterprojekten „7.28.03.04 – Foundation Subject Area“ in [ISOCDIF98f], „7.28.03.05 – Common Subject Area“ in [ISOCDIF98g], „7.28.03.08 – Data Modeling Subject Area“ in [ISOCDIF98h] sowie „7.28.03.07 – Data Flow Model Subject Area“ in [ISOCDIF98i]. Für die Entwürfe zu den Unterprojekten „7.28.03.02 – Presentation Location And Connectivity Subejct Area“, „7.28.03.06 – Data Definitions Subject Area“ sowie „07.28.03.09 – State/Event Model Subject Area“ konnten keine zitierbaren ISO/IEC-bezogenen Quellen (Stand: 1998-06-01) aufgefunden werden.

Die im Rahmen des SEDDI-Projekts von ISO/IEC JTC 1/SC 7 WG 11 erstellten Standards werden in dieser Arbeit mit „ISO/CDIF“[82] bezeichnet.[83]

1.3.2.3 EIA/CDIF und STEP/EXPRESS (ISO/IEC TC 184/SC 4)

1984 wurde ein ISO/IEC Projekt begonnen, das sich im wesentlichen mit allen Aspekten des Austausches von Produktdaten auseinandersetzt und unter dem Akronym „STEP“[84] bekannt geworden ist.[85] Sämtliche bisher vom Subkomitee ISO/IEC TC 184/SC 4, „Industrial automation systems and integration/Industrial data“, beziehungsweise dessen Arbeitsgruppen[86] erarbeiteten Standards sind unter der ISO-Familie 10303[87] zusammengefaßt.

Im Zusammenhang mit EIA/CDIF sind vor allem die Standards ISO 10303-1:1994, „Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles“, ISO 10303-11:1994, „Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual“ und ISO 10303-21:1994, „Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure“ von Interesse, da sie sich mit dem Austausch von (Modell-) Daten auseinandersetzen.

---

81 Dieses Unterprojekt baut auf ECMA-270 auf, mit Stand 1998-06-01 konnte keine zitierbare ISO/IEC-bezogene Quelle aufgefunden werden. Zudem ist es wahrscheinlich, daß dafür eine neue ISO/IETC-Nummer vergeben wird.

82 Aufgrund des Akronyms „CDIF“ in der Bezeichnung „ISO/CDIF“ kommt der unmittelbare Bezug zu EIA/CDIF zum Ausdruck.

83 Die ISO/CDIF-Entwürfe unterscheiden sich mittlerweile in Teilen von den EIA/CDIF-Standards. Allerdings sind bei entsprechendem Verständnis von EIA/CDIF die Änderungen, etwa am Meta-Meta-Modell oder im Meta-Modell „Common“, leicht zu verstehen und nachvollziehbar.

84 „STEP“ stellt das Akronym für „Standard for the Exchange of Product and process model data“ dar.

85 Vgl. hierzu auch die Ausführungen in [Vern96], Seiten 359 bis 364, sowie die Kurzübersicht in [Ernst98], Seiten 60 bis 61.

86 Unter dem URL (Stand: 1998-06-01) „http://www.iso.ch/meme/TC184SC4.html“ können die jeweils aktiven Arbeitsgruppen abgerufen werden.

87 Unter dem URL (Stand: 1998-06-01) „http://www.iso.ch/liste/TC184SC4.html“ können die jeweils verfügbaren ISO-STEP-Standards abgerufen werden.

Ähnlich wie im Abschnitt 1.3.2.1, „EIA/CDIF, ECMA-149 und ECMA-270“[88], wird im Rahmen der Arbeiten von ISO/IEC JTC 1/SC 7 WG 11 versucht, eine Abbildung von EIA/CDIF-Meta-Modellen mit Hilfe von EXPRESS durchzuführen. Es werden ähnliche Ziele verfolgt, wie bei ECMA-270 und es gibt bereits Entwürfe, wie EIA/CDIF-Meta-Modelle mit Hilfe von EXPRESS repräsentiert werden können.[89]

1.3.2.4 EIA/CDIF und OMG (UML und SMIF)

Die „Object Management Group“[90] (abgekürzt: „OMG“) wurde 1989 mit dem Ziel gegründet, Standards für verteilte Objekttechnologien zu erstellen.[91] Mit Anfang Juni 1998 zählen laut OMG über 800 Firmen und Organisationen zu ihren Mitgliedern.

Im Rahmen der „Analysis and Design Task Force“ (abgekürzt: „ADTF“) wurden verschiedene „Request for Proposals“ veröffentlicht, wobei in diesem Zusammenhang zwei von Bedeutung sind:

– RFP für die Analyse und Design, das als Ergebnis im Herbst 1997 den OMG Standard „UML“ erbrachte.[92] Der erste Satz an UML-Entwürfen der Firma Rational von Anfang 1997 beinhaltete neben einem wenig beachteten Anhang, der das Meta-Meta-Modell von UML beschreibt[93], auch einen, der beschreibt, wie man UML-Modelle mit Hilfe von EIA/CDIF tauschen könnte.[94, 95]

---

88 Vgl. Seite 17ff oben.

89 Mit 1998-06-01 existiert vom Britischen Normungsinstitut ein nichtöffentlicher Entwurf dazu, der als ISO/IEC Standard veröffentlicht werden soll. Der Autor gelangte aufgrund seiner freien Mitgliedschaft in einer geschlossenen Listserver-Diskussionsliste für ISO/IEC JTC 1/SC 7 WG 11 zur Kenntnis dieses Entwurfes.

90 Siehe auch [W3OMG].

91 Der bekannteste OMG-Standard ist bis dato sicher der „CORBA“-Standard, Akronym für: „Common Object Request Broker Architecture“. Die Spezifikationen der Version 2.2 mit Stand Februar 1998 können von [CORBA98] über das WWW bezogen werden.

92 Die endgültigen Standards finden sich in [UML97c], [UML97d] und [UML97e].

93 Vgl. hierzu [UML97b].

94 Vgl. hierzu [UML97f].

95 Vgl. in diesem Zusammenhang beispielsweise [CDIF97a] oder [Flat98a], die den Austausch von UML-Modellen mit Hilfe von EIA/CDIF grundsätzlich diskutieren.

– Der SMIF-RFP, „Stream-based Model Interchange Format“[96] (abgekürzt: „SMIF“), wurde im Zusammenhang mit den Abstimmungsarbeiten zum OMG-Standard „UML“ erstellt. Damit wurde dem Austausch von Modellen unabhängig von UML ganz allgemein ein besonderes und eigenständiges Gewicht im Rahmen der OMG-Standardisierungen verliehen. Neben der Forderung dem OMG Meta-Meta-Modell „Meta Object Facility“ (abgekürzt: „MOF“) zu entsprechen, sollen die Vorschläge nachweisen, daß es damit möglich ist, Modelldaten zu exportieren und ohne Informationsverlust wieder zu importieren. Abschnitt 6.4 in [OMG97], „Related Documents and Standards“, führt als Dokumente und Standards CDIF und STEP EXPRESS an. In Abschnitt 6.6[97], „Optional Requirements“, wird ausführlich auf EIA/CDIF eingegangen, wobei auch eine Abbildung des EIA/CDIF-Meta-Meta-Modells auf OMG’s MOF angesprochen wird. Hier wird im besonderen darauf wert gelegt, daß SMIF-Vorschläge über Vorkehrungen verfügen, die aus EIA/CDIF-Austauschdaten valide SMIF-Austauschdaten erzeugen.[98] Aus dem SMIF RFP soll bis März 1999 ein neuer OMG Standard entstehen.

Somit sind auch die Arbeiten an den EIA/CDIF-Standards im Rahmen der Standardisierungsbemühungen der OMG relevant.

1.4 Ziele der Arbeit

In diesem ersten Kapitel wurde versucht, ausgehend von der Notwendigkeit Meta-Modelle für unterschiedliche Zwecke zu erstellen, nachzuweisen, daß im Rahmen der deutschsprachigen Wirtschaftsinformatiker am Beispiel von Ferstl/Sinz, Österle und Scheer der Einsatz von Meta-Modellen im Rahmen von betriebswirtschaftlichen Problemstellungen systematisch erfolgt. Bis dato hat sich hier allerdings kein einheitlicher Standard zur Erstellung von Meta-Modellen herauskristallisiert, sodaß die erstellten Meta-Modelle der deutschsprachigen Wirtschaftsinformatiker unterschiedlichen Modellierungsregeln gehorchen.

---

96 Vgl. [OMG97].

97 Vgl. in diesem Zusammenhang auch die Ziele, die auf Seite 2 von [OMG97] angegeben sind.

98 Dieselbe Forderung wird auch bezüglich STEP EXPRESS-Austauschdaten aufgestellt.

Insoferne wird das Einarbeiten in die unterschiedlichen Meta-Modelle erschwert, da dies zunächst voraussetzt, daß die entsprechenden Meta-Meta-Modelle im Detail zuvor erarbeitet und verstanden werden.

Es wurde des weiteren versucht, die seit 1987 entwickelten EIA/CDIF-Standards für den herstellerunabhängigen Austausch von Modelldaten vorzustellen und ihre Relevanz dadurch zu begründen, indem aufgezeigt wurde, in welchen weiteren Standardisierungsorganisationen sie Eingang gefunden haben. In diesem Zusammenhang ist die Entwicklung eines ISO/CDIF Standards, dessen Verabschiedung für November 1998 erwartet wird, von Interesse, sowie jene Arbeiten, die die CDIF-Standards für ECMA-PCTE sowie für STEP EXPRESS erschließen (sollen). Es kann auch erwartet werden, daß die CDIF-Standards im Rahmen des OMG RFP „SMIF“ eine wichtige Rolle spielen und ihren Niederschlag in den dafür zu erarbeiteten Standard im Frühjahr 1999 finden.

Im weiteren Verlauf dieser Arbeit soll in Form eines Diskurses[99] erstmals eine systematische, zusammenhängede Aufbereitung der EIA/CDIF-Standards erfolgen. Damit soll die weiter oben erwähnte hohe Einarbeitungszeit – und damit auch ein hoher Kostenfaktor – auf ein Minimum für Interessierte und Fachkundige reduziert werden. In weiterer Folge soll dadurch die Möglichkeit geschaffen werden, die EIA/CDIF-Standards einer wissenschaftlichen Diskussion zuzuführen. Dies bedeutet aber gleichzeitig, daß auch die ISO/CDIF-Standards beziehungsweise weitere Standards, die sich auf EIA/CDIF beziehen, selbst einem wissenschaftlichen Diskurs zugänglich gemacht werden. Zudem sollen die Darstellungen so detailliert erfolgen, daß es grundsätzlich möglich wird, eigenständig EIA/CDIF-konforme Meta-Modelle zu erstellen.

Es wäre aus Sicht des Autors wünschenswert, würde es zumindest unter den deutschsprachigen Wirtschaftsinformatikern dazu kommen, daß sie für die Entwicklung ihrer eigenen Meta-Modelle ein einheitliches Meta-Meta-Modell zugrundelegen. Nachdem das EIA/CDIF-Meta-Meta-Modell in Form einer erwei-

99 Vgl. hierzu beispielsweise die Ausführungen in [LeMaHi95], Seite 29ff.

terten Entity-Relationship-Modellierungsmethode definiert ist, erscheint es dafür als besonders geeignet und wird daher auch im Detail hier vorgestellt.[100]

Im Rahmen dieser Arbeit sollen jedenfalls die folgenden Ziele weiterverfolgt werden:

- integrierende Darstellung[101] des EIA/CDIF-Meta-Meta-Modells und sämtlicher, standardisierter EIA/CDIF-Meta-Modelle,
- Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells in relationale Datenbanken (Kapitel 2) und darauf aufbauende Implementierungen (Anhang) zur Demonstration,
- Spezifikation für die Abbildung des EIA/CDIF-Meta-Meta-Modells in einer objektorientierten Programmiersprache (Kapitel 2) und darauf aufbauende Implementierungen (Anhang) zur Demonstration,
- erstmalig die physische Erstellung des sogenannten „Integrierten EIA/CDIF-Meta-Modells“ durchzuführen und so zu dokumentieren, daß es auch als Referenz benutzt werden kann,
- sämtliche Definitionen des Integrierten Meta-Modells daraufhin zu überprüfen, ob sie selbst den Regeln der EIA/CDIF-Standards entsprechen,
- die Strukturen der Meta-Modelle in Form von (weiterverarbeitbaren) Auswertungen darzustellen,
- ein Meta-Modell zur Umgehung einiger Einschränkungen des minimalen EIA/CDIF-Meta-Meta-Modells erstellen, das Informationen, die für Meta-CASE-Systeme wichtig sein können, aufnehmen kann,
- sämtliche Surrogate von Meta-Objekten aus dem Integrierten Meta-Modell für das Meta-Meta-Attribut „CDIFMetaIdentifier“ in Form von Tabellen mit den entsprechenden Meta-Objekten als Referenz darzustellen, und

---

100 Damit könnten auch über zehn Jahre an intensiver Entwicklungsarbeit zunutze gemacht werden.

101 Beispielsweise sollen die OMG IDL Standards dort eingearbeitet werden, wofür sie erstellt wurden, nämlich bei der Diskussion des Meta-Meta-Modells selbst und bei der Diskussion des fundamentalen Meta-Modells. Soweit dies sinnvoll erscheint, werden Querverweise über Meta-Modellgrenzen hinweg angegeben, sowie zu den Meta-Modellen passende Quellen angegeben, die nicht in den EIA/CDIF-Standards selbst enthalten sind, aber für ein weiteres Studium als geeignet erscheinen.

– sämtliche qualifizierte Bezeichner der Meta-Objekte aus dem Integrierten Meta-Modell in Form von Tabellen mit den entsprechenden Surrogatwerten des Meta-Meta-Attributs „CDIFMetaIdentifier“ als Referenz darzustellen.

Im übrigen wird in der gesamten Arbeit bewußt die Diktion aus den EIA/CDIF-Standards benutzt, um den/die Leser/in in diese Sprachwelt einzugewöhnen. Um einerseits Wortwiederholungen zu vermeiden und andererseits den Sinn oder Unsinn[102] von deutschen Übersetzungen vor Augen zu führen, werden die englischen Bezeichner und ihre deutschen Übersetzungen austauschbar eingesetzt.

102 Sofern die deutschen Übersetzungen vom Autor in ihrer Bedeutung gleich geraten sind, wie ihre englischen Originale, sollte der Einsatz der deutschen Begriffe zu keinen unsinnigen Sätzen führen, sondern im Gegenteil, Aussagen manchmal leichter verständlich machen.

2 Überblick über EIA/CDIF

In diesem Abschnitt wird ein Überblick über die grundlegende Architektur, das Vier-Schichten-Modell, von EIA/CDIF gegeben sowie grundlegende Definitionen für die EIA/CDIF-Begriffe. Es schließt mit einem Beispiel eines Austausches von Modelldaten ab, wobei für diesen Zweck die Erweiterbarkeit von EIA/CDIF ausgenutzt wird, indem zuvor ein eigenständiges Meta-Modell vollständig definiert wird, das anschließend durch die Modelldaten instanziiert wird. Das Beispiel in Abschnitt 2.3.2 auf Seite 40 demonstriert gleichzeitig die Anwendung der EIA/CDIF-Standards „SYNTAX.1“ und „ENCODING.1“.

2.1 Das „CASE Data Interchange Format“ (CDIF)

Das „CASE[103] Data Interchange Format“ (Abkürzung: „CDIF“) wird im Rahmen einer Standardisierungsgruppe der „Electronic Industries Alliance“ (Abkürzung: „EIA“) seit 1987 entwickelt. Diese amerikanische Industriestandardisierungsorganisation von Anbietern und Kunden versucht in ihrer Arbeit, auch Anwender von Technologien in die Entwicklung ihrer Standards einzubeziehen, um dadurch die Marktakzeptanz zu erhöhen. EIA ist unter anderem mit dem „American National Standard Institute“ (Abkürzung: „ANSI“) und der „International Organization for Standardization“ (Abkürzung: „ISO“) assoziiert.

Ausgangspunkt der Entwicklung von CDIF war die Beobachtung, daß unterschiedliche Hersteller von CASE-basierten Werkzeugen und Systemen zwar bestimmte Methodologien wie beispielsweise die Entity-Relationship-Modellierung unterstützten, die erstellten Modelle aber nicht in die Werkzeuge und Systeme anderer Hersteller übertragbar waren. Diese mangelhafte Übertragbarkeit beziehungsweise *Austauschbarkeit* von Modellierungsergebnissen wurde einerseits durch inkompatible, undokumentierte und proprietäre Datenformate

---

103 „CASE“ stellt die Abkürzung für „Computer Aided Software Engineering“ dar (vgl. z.B. [Balz91] beziehungsweise eine Charakterisierung des Begriffs „CASE“ in [Hans96], Seite 858ff).

der verschiedenen Hersteller der CASE-Systeme verursacht, andererseits durch den unterschiedlichen Einsatz von *Methodologievarianten*[104].

Sofern ein CASE-Anwender das verwendete CASE-System durch spezialisierte CASE-Werkzeuge ergänzen oder das eingesetzte CASE-Produkt durch das eines anderen Herstellers ersetzen wollte, müssen sämtliche entworfenen Modelle für das neue Werkzeug beziehungsweise System neu erfaßt, kontrolliert und überarbeitet werden. Eine zeitaufwendige, fehleranfällige und damit kostentreibende Tätigkeit für CASE-Anwender. Für die CASE-Anbieter auf der anderen Seite entsteht dadurch eine Situation, in der die Kunden abhängig beziehungsweise in jedem Fall in ihrer Mobilität durch bereits erfolgte Investitionen von Ressourcen in die Erstellung von (größeren) Modellen eingeschränkt werden, wodurch die Herstellerbindung entsprechend steigt. Daher liegt es im Interesse der Anwender mit Hilfe von Standards dieser Herstellerabhängigkeit zu entkommen.

Ein herstellerunabhängiger Standard für den Austausch von Modelldaten hilft den Anwendern und kann aus unterschiedlichsten Gründen auch für die Anbieter von CASE-Tools und CASE-Systemen interessant sein:

- Ein CASE-System, das unterschiedliche Modellierungsmethodologien unterstützt, kann die Integration von unterschiedlichen Modellen einfacher durchführen, wenn die Modelle von Grund auf *nach denselben Konstruktionsprinzipien* erstellt wurden.[105]
- Es wird möglich, daß die Modellierergebnisse von CASE-Werkzeugen von Drittherstellern, in die bestehenden Modelle beziehungsweise CASE-Systeme *integriert* werden. Umgekehrt werden die in einem bestimmten CASE-System erstellten Modelle in Systemen von Drittherstellern weiterverarbeitbar.

---

104 Man denke nur an die verschiedenen Variationen der Entity-Relationship-Modellierung, beginnend mit den Anfängen in [Chen76] bis hin zu aktuelleren Ausprägungen, wie sie etwa in [BaCeNa92] dargestellt sind.

105 Mit anderen Worten, unterschiedliche Modellierungswerkzeuge benutzen für die Modellbildung dasselbe Meta-Modell. Sofern für unterschiedliche Methodologien Meta-Modelle zur Definition benutzt werden, ist hier dementsprechend gemeint, daß die Konstruktion derartiger Meta-Modelle auf Basis ein- und desselben Meta-Meta-Modells erfolgt.

– Gegenüber den Kunden der CASE-Systeme wird ausdrücklich auf die Bildung einer möglichen Herstellerabhängigkeit durch Offenheit und Kooperation verzichtet, sowie Verständnis für die Bedingungslage der Anwender signalisiert und insgesamt dadurch Vertrauen konstituiert.

EIA/CDIF nahm 1987 ihre Arbeit auf und setzt sich seitdem aus CASE-Tool-Hersteller und CASE-Tool-Anwender zusammen, die einander regelmäßig[106] vier bis sechs Mal pro Jahr für eine ganze Woche treffen. 1991 wurde der erste Satz an „EIA/CDIF-Interim Standards“ verabschiedet, der die Grundlagen für den Austausch von Modelldaten zwischen CASE-Werkzeugen beziehungsweise CASE-Systemen unterschiedlicher Hersteller ermöglicht und von den Beteiligten in weiterer Folge umgesetzt[107] wurde.

Die Grundlagen des EIA/CDIF-Austauschformats und somit auch der Definitionen für die Bildung von Austauschmodellen in Form von EIA/CDIF-Meta-Modellen[108], die als Basis für den Austausch der Modelldaten dienen, wurden weiterentwickelt. 1994 wurde ein weiterer, überarbeiteter Standard, verabschiedet, der die Version aus dem Jahre 1991 ablöste.[109]

Erst nachdem der 1994 Standard verabschiedet wurde, forcierte EIA/CDIF die offizielle Standardisierung der ersten *semantischen* EIA/CDIF-Meta-Modelle, wie „Data Flow Model Subject Area“[110], „Data Modeling Subject Area“[111] oder „Data Definition Subject Area“[112].[113]

---

106 Üblicherweise treffen die EIA/CDIF-Mitglieder einander sechs Mal im Jahr. 1996 wurde als weitere Kommunikationsform die Telefonkonferenz eingeführt, die zum Teil die persönlichen Treffen substituiert und es den Mitgliedern trotzdem ermöglicht, zu aktuellen Fragen Stellung zu nehmen oder an Abstimmungen teilzunehmen.

107 Die Firma ORACLE hat beispielsweise als Folge der Verabschiedung der 1991-er „EIA/CDIF-Interim Standards“ in einem Buch (vgl. [ORAC92]) das Meta-Modell des ORACLE*CASE-Dictionaries in Forme eines „EIA/CDIF-Austausches“ dokumentiert.

108 Auch hier und in weiterer Folge wird aus Gründen der besseren Lesbarkeit statt „Metamodell“ und „Metametamodell“ die Schreibweise mit Bindestrichen zwischen den verschiedenen Wörtern eingeführt, somit die Synonyme „Meta-Modellen“ und „Meta-Meta-Modellen“ dafür verwendet.

109 Somit wurde über einen Zeitraum von sieben (!) Jahren von EIA/CDIF darauf verwendet, die *Grundlagen* für den Austausch von Modelldaten zu schaffen: das EIA/CDIF-Meta-Meta-Modell, die CDIF-Syntax für den Austausch und die CDIF-Verkodierung der Austauschdaten. Dies stellt einen interessanten Indikator für die Komplexität der zugrundeliegenden Problemstellung dar.

110 Vgl. [CDIF96a].

111 Vgl. [CDIF96b].

112 Vgl. [CDIF97a].

Der EIA/CDIF-Standard setzt sich aus einer Reihe von verschiedenen Standards zusammen:

1. Übersicht über das EIA/CDIF-Meta-Meta-Modell: „CDIF – Framework for Modeling and Extensibility“,[114]
2. Definition des fundamentalen Meta-Modells: „CDIF – Integrated Metamodel, Foundation Subject Area“,[115]
3. Übersicht über das Verschlüsseln und Austauschen von Daten: „CDIF – Transfer Format – General Rules for Syntaxes“,[116]
4. Definitionen für die Syntax des Austauschformates: „CDIF – Transfer Format Syntax – SYNTAX.1“,[117]
5. Definitionen für das Verschlüsseln von Daten: „CDIF – Transfer Format Encoding – ENCODING.1“,[118] sowie
6. weitere Meta-Modelle.

Die Bildung dieser unterschiedlichen Schwerpunkte soll die Benutzer des EIA/CDIF-Standards das systematische Einarbeiten erleichtern. Diese Reihenfolge stellt zugleich auch eine Ordnung dar, die zum Erarbeiten des Verständnisses anhand der originalen EIA/CDIF-Standards empfohlen wird.[119]

---

113 Das EIA/CDIF-Meta-Modell „Foundation Subject Area“ wurde 1994 gemeinsam mit den anderen konstituierenden Teilen des Standards verabschiedet. Es definiert die Wurzel („RootObject“) und die beiden Spezialisierungsformen „RootEntity“ (eine Meta-Entität) und „RootEntity.IsRelatedTo.RootEntity“ (eine Meta-Beziehung), die wiederum in weiterer Folge durch Meta-Entitäten und Meta-Beziehungen, die in weiteren Meta-Modellen definiert werden, spezialisiert werden müssen. Die „Foundation Subject Area“ zählt daher nicht zu den semantischen Meta-Modellen, die für den Austausch von Modelldaten für bestimmte Methodologien definiert werden. Eine systematischere Unterscheidung wird durch die Definition der Meta-Entität „SemanticInformationObject“ im EIA/CDIF-Meta-Modell „Common Subject Area“ möglich. Weitere Ausführungen und Diskussionen zu diesen beiden grundlegenden Meta-Modellen finden sich weiter unten (vgl. Abschnitt 4.1.1, Seite 155ff, und Abschnitt 4.1.2, Seite 176ff).

114 Vgl. [CDIF94b].

115 Vgl. [CDIF94f].

116 Vgl. [CDIF94c].

117 Vgl. [CDIF94d].

118 Vgl. [CDIF94e].

119 In dieser Arbeit wird nach Möglichkeit dieselbe Ordnung beziehungsweise Gliederung benutzt, wie in den EIA/CDIF-Standards selbst. Dadurch soll es für den Leser einfacher werden, von dieser Arbeit auf die Standards zu verzweigen und umgekehrt.

2.2 Die Architektur von EIA/CDIF

EAI/CDIF definiert für die Modellierung von Daten eine Variante der konzeptionellen (Daten-)Modellierung, wie sie ursprünglich von [Chen76] entworfen und später in [BaCeNa92], [ElmNav89] und [ISO97a] weiterentwickelt und dokumentiert wurde. Es nutzt die Konzepte[120] von attribuierbaren „Entitätstypen“ und „Beziehungstypen“, die in einer Generalisierungshierarchie angeordnet sind.

2.2.1 Das Vier-Schichten-Modell

Die Architektur des Austausches von Modelldaten mit Hilfe des EIA/CDIF-Standards wird in vier verschiedenen Ebenen angeordnet: drei Modellebenen[121] mit instanziierbaren Konzepten[122] sowie einer vierten, extensionalen Ebene („Benutzerdaten“). Das Vier-Schichten-Modell geht ursprünglich auf die Arbeiten im Zusammenhang mit ANSI-IRDS zurück und wird beispielsweise in [HabLey93][123] vorgestellt und diskutiert. Dieselbe Sichtweise der vier Schichten verwendet auch der Standard „Meta Object Facility“ (abgekürzt: MOF) der OMG. Tabelle 2-1 stellt diese vier Ebenen anhand von EIA/CDIF und von OMG/MOF dar.

---

120 Vgl. hierzu die Ausführungen zur konzeptionellen Datenmodellierung am Ende des Abschnitts 2.2.2.1, Seite 33ff weiter unten.

121 Unabhängig von EIA/CDIF und OMG/MOF unterscheidet auch [ISO97a] („Information Technology, Conceptual Modelling Facilities“) im Abschnitt 5.3, „Concepts of Use“, die drei Modellebenen „Meta-Meta Model“ („Defining Schema“), „Meta Model“ (die dem Meta-Modell „Normative Schema“ folgen müssen) und die Schicht „Modelling Schema“ (auch „Application Model“ genannt). Auch hier fungiert die übergeordnete Schicht als Sprache, die die Modellierungsmöglichkeiten beziehungsweise die Modellierungsausdruckmöglichkeiten festlegen.

122 Es handelt sich also um instanziierbare „Entitätstypen“ und „Beziehungstypen“, die in EIA/CDIF auch als „Meta-Typen“ bezeichnet werden, da ihre Instanzen selbst instanziierbare Typen sind. Im Vier-Schichten-Modell sind die Konzepte der M3- und M2-Schicht demgemäß MetaTypen.

123 Vgl. ebenda Abschnitt 5.2, „Die 4-Schichten-Architektur“, Seite 77ff.

EIA/CDIF[124]		OMG/MOF[125]
M3[126]	Meta-Meta-Modell	„meta-metamodel“: „MOF Layer“
M2	Meta-Modell	„metamodel“: „OA&DF (UML) Layer“
M1	Modell	„model“ („UserObject Model“)
M0	Benutzerdaten	„user objects“ („User Data)

Tabelle 2-1: Vier-Schichten-Modell von EIA/CDIF und von MOF

Die jeweils höhere Schicht legt „eine genaue und vollständige Definition der Instanzen der jeweils untergeordneten Schicht fest.“[127, 128] Die Konzepte, die das untergeordnete Modell festlegen, werden als *Instanzen* von Konzepten des übergeordneten Modells angesehen. Im Kontext der Modellierung definiert die übergeordnete Schicht eine Sprache (und Notation), mit deren Hilfe in der unmittelbar untergeordneten Schicht modelliert werden kann.[129]

Die Besonderheit der M0-Schicht im Vergleich mit der M1-, M2- und M3-Schicht liegt darin, daß es im Unterschied zu den anderen Schichten nicht über instanziierbare Konzepte verfügt.[130] Da EIA/CDIF sich mit dem Austausch von Model-

124 Vgl. die Definition für „Model Layers“ in den Glossaren zu den EIA/CDIF-Standards, beispielsweise [CDIF94b], Seite 93, oder die Ausführungen hierzu in [CDIF97e].

125 Vgl. Tabelle 1 auf Seite 3.3 in [MOF97a].

126 Die dem Buchstaben „M“ folgende Ziffer gibt die Anzahl der „M“s an, mit denen die einzelnen Wörter des (zusammengesetzten) Begriffs beginnen. So weist beispielsweise „M3“ darauf hin, daß es für den zusammengesetzten Begriff „**M**eta- **M**eta- **M**odell“ steht.

127 Vgl. hierzu den Glossareintrag „Model Layers“ in [CDIF94b], Seite 93.

128 Grundsätzlich kann eine höhergelegene Schicht als „Intension“ und die direkt untergeordnete Schicht als die entsprechende „Extension“ angesehen werden. Damit können hier drei Intensions-/ Extensionspaare angenommen werden: <M3,M2>, <M2,M1>, <M1,M0>. Vgl. in diesem Zusammenhang auch die entsprechenden Diskussionen im Rahmen des ANSI/IRDS-Vier-Schichtenmodell in [HabLey93], beispielsweise auf den Seiten 61ff, 77 Mitte, 79 Mitte. Ebenda auf Seite 80 findet sich eine Übersichtsabbildung der unterschiedlichen Schichten, wobei die Benennung vom Meta-Meta-Modell mit der Nummer 1 beginnt und als „L1“ (englische Abkürzung für „Ebene 1“: „**L**evel **1**“) bezeichnet wird. Diese Schichteinteilung endet dementsprechend mit der Ebene 4 (Kurzbezeichnung: „L4“) und entspricht in EIA/CDIF und OMG/MOF der Schicht „M0“.

129 Vgl. die Spalte „Description“ in Tabelle 1 auf Seite 3.3. in [MOF97a], wo für die M3-Schicht festgelegt wird: „The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.“, für die M2-Schicht: „An instance of a meta-metamodel. Defines the language for specifying a model.“, für die M1-Schicht „An instance of a metamodel. Defines a language to describe an information domain“ und schließlich für die M0-Schicht: „An instance of a model. Defines a specific information domain.“.

130 Damit sind M0-Daten atomar, das heißt nicht weiter instanziierbar beziehungsweise zerlegbar.

len – also der M1-Schicht – auseinandersetzt, wird die M0-Ebene in den verschiedenen EIA/CDIF-Standards nicht direkt abgehandelt und in dieser Arbeit daher nicht weiter ausgeführt.

2.2.2 EIA/CDIF-Definitionen

In diesem Abschnitt werden die in dieser Arbeit benutzten Schreibweisen und die Definitionen von EIA/CDIF für die wichtigsten EIA/CDIF-Begriffe[131] vorgestellt.

2.2.2.1 Begriffsbildung

Im Rahmen dieser Arbeit werden folgende Begriffe häufig verwendet:

- Entität, Entitätstyp, Meta-Entität, Meta-Entitätstyp, Meta-Meta-Entität, Meta-Meta-Entitätstyp,
- Beziehung, Beziehungstyp, Meta-Beziehung, Meta-Beziehungstyp, Meta-Meta-Beziehung, Meta-Meta-Beziehungstyp,
- Attribute, Meta-Attribute, Meta-Meta-Attribute.

Die Begriffe „Meta-Meta-Entitätstyp“ („Meta-Meta-Beziehungstyp“; englisch: „Meta-Meta-Entity-Type“ respektive „Meta-Meta-Relationship-Type“) und „Meta-Meta-Entität“ („Meta-Meta-Beziehung“; englisch: „Meta-Meta-Entity“ respektive „Meta-Meta-Relationship“) werden synonym verwendet und gehören der M3-Schicht an.[132] Eigenschaften von Meta-Meta-Entität(styp)en werden in Form von „Meta-Meta-Attributen“ (englisch: „Meta-Meta-Attribute“) angegeben. „Meta-Meta-Objekt“ („englisch: „Meta-Meta-Object“) ist ein generischer Begriff für Meta-Meta-Entitäten und Meta-Meta-Beziehungen.

Die Begriffe „Meta-Entität“ („Meta-Beziehung“; englisch: „Meta-Entity“ respektive „Meta-Relationship“), „MetaEntität“ („MetaBeziehung“; englisch: „MetaEntity“

---

131 Die EIA/CDIF-Begriffsdefinitionen wurden bis Anfang 1998 in Form eines Glossars (englisch: „Glossary“) sämtlichen EIA/CDIF-Interim-Standards im Anhang beigegeben. In diesem Abschnitt wird das Glossar von [CDIF94b] referenziert, es hätten aber genauso gut die identischen Glossare etwa in [CDIF95] oder in [CDIF94a] benutzt werden können.

132 Dies folgt aus der Überlegung, daß das EIA/CDIF-Meta-Meta-Modell selbst mit Hilfe des EIA/CDIF-Meta-Meta-Modells definiert wird. Daher können die im EIA/CDIF-Meta-Meta-Modell enthaltenen Meta-Meta-Enitätstypen *und* Meta-Meta-Beziehungstypen zugleich auch als Meta-Meta-Entitäten und Meta-Meta-Beziehungen angesehen werden.

respektive „MetaRelationship“), „Meta-Entitätstyp“ („Meta-Beziehungstyp“; englisch: „Meta-Entity-Type“ respektive „Meta-Relationship-Type“) werden synonym verwendet und gehören der M2-Schicht an.[133] Eigenschaften von Meta-Entität(styp)en und Meta-Beziehung(styp)en werden in Form von „MetaAttributen“ (englisch: „MetaAttribute“) beziehungsweise synonym mit „Meta-Attributen“ (englisch: „Meta-Attribute“) angegeben. „Meta-Objekt“ (englisch: „Meta-Object“) beziehungsweise „MetaObjekt“ (englisch: „MetaObject“) ist ein generischer Begriff für Meta-Entitäten und Meta-Beziehungen.

Die Begriffe „Entität“ („Beziehung“; englisch: „Entity“ respektive „Relationship“) und „Entitätstyp“ („Beziehungstyp“; englisch: „Entity-Type“ respektive „Relationship-Type“) gehören der M1-Schicht an. Eigenschaften von Entität(styp)en und Beziehung(styp)en werden in Form von „Attributen“ (englisch: „Attribute“) angegeben.[134]

Unabhängig davon werden die Begriffe „Entitätstyp“, „Entitäten“, „Beziehungstyp“, „Beziehung“ und „Attribut“ auch im Sinne und den Bedeutungen der konzeptionellen Modellierung verwendet:[135]

– Entität: ein (materielles oder immaterielles) Objekt, das von gleichartigen anderen Objekten eindeutig unterscheidbar ist.

– Entitätstyp: durch Verallgemeinerung (Abstraktion) von einzelnen Entitäten gelangt man zu Entitätstypen, die in diesem Zusammenhang zusätzlich durch Eigenschaften in Form von Attributen beschrieben werden können.[136]

---

133 Eine „MetaEntität“ der M2-Schicht wird als Instanz des Meta-Meta-Entitätstyps „MetaEntity“ (deutsche Übersetzung: „MetaEntität“) der M3-Schicht angesehen. Um die Lesbarkeit zu verbessern wird der Begriff „MetaEntität“ mit einem Bindestrich versehen: „Meta-Entität“. Eine „Meta-Entität“ der M2-Schicht stellt gleichzeitig einen „Meta-Entitätstyp“ aus der Sicht der M1-Schicht dar, nachdem „Entitäten“ die Extension von „Meta-Entitätstypen“ darstellen. Diese Ausführungen gelten analog für „MetaBeziehung“, „Meta-Beziehung“ und „Meta-Beziehungstyp“.

134 Eine „Entität“ („Beziehung“) der M1-Schicht wird als Instanz von M2-Schicht-Entitätstypen (Beziehungstypen) angesehen, gleichzeitig stellen sie selbst die „Entitätstypen“ („Beziehungstypen“) für die M0-Schicht dar.

135 Vgl. in diesem Zusammenhang beispielsweise [BaCeNa92], [ElmNav89], [Flat90], [ISO97a], [Vett91]. Die Definitionen folgen hier [Flat96b], Abschnitt 2.2.1 auf Seite 22ff.

136 Eine Entitäts*menge* umfaßt sämtliche Entitäten eines Entitäts*typs*, der Typ stellt insofern die Intension, die Menge die Extension dar. In [ISO97a] wird für Entitätsmenge der Begriff „Klasse“ benutzt.

– Beziehung: ein immaterielles Objekt, das mindestens zwei Entitäten miteinander assoziiert und das von gleichartigen, anderen Objekten eindeutig unterscheidbar ist.[137]

– Beziehungstyp: durch Verallgemeinerung (Abstraktion) von einzelnen Beziehungen gelangt man zu Beziehungstypen, die in diesem Zusammenhang zusätzlich durch Eigenschaften in Form von Attributen beschrieben werden können.

– Kardinalitäten: Verhältniszahl, die angibt, wieviele Entitäten eines Typs über einen bestimmten, binären Beziehungstyp mit Entitäten eines anderen Typs in Beziehung stehen. In EIA/CDIF werden die einzelnen Verhältniszahlen jeweils mit ihren Minimum- und – durch einen Doppelpunkt getrennt – Maximumwerten angegeben.[138]

Somit sind die EIA/CDIF „Meta-Meta-Entitätstypen“ („Meta-Meta-Beziehungstypen“), „Meta-Entitätstypen“ („Meta-Beziehungstypen“) auch „Entitätstypen“ („Beziehungstypen“) im konzeptionellen Sinn und „Meta-Meta-Entitäten“ („Meta-Meta-Beziehungen“) sowie „Meta-Entitäten“ („Meta-Beziehungen“) entsprechend „Entitäten“ („Beziehungen“).[139]

2.2.2.2 Begriff „EIA/CDIF Meta-Meta-Modell“

„Das EIA/CDIF-Meta-Meta-Modell definiert die Menge an Konzepten und Notationen, die dazu benutzt werden, Meta-Modelle zu erstellen. Im besonderen definiert das EIA/CDIF-Meta-Meta-Modell ein Entitäts-Beziehungs-Attributs-

---

137 [ISO97a] benutzt auch für Beziehungen den Begriff Entität, vgl. Abschnitt 6.1, „Fundamental Concepts for the Universe of Discourse“.

138 [BaCeNa92] vertauschen im Vergleich zur EIA/CDIF Darstellung die Kardinalitäten bei binären Beziehungstypen, vgl. hierzu [BaCeNa92] Seiten 31 bis 33. Damit ist es mit der Notation von [BaCeNa92] auch möglich, Aussagen über die Anzahl der minimalen und maximalen eingegangenen Beziehungen einzelner Entitäten eines bestimmten Entitätstyps in einem bestimmten Beziehungstyp anzugeben. Diese strukturellen Bedingungen werden häufig über das Konzept der „Rolle“ vermerkt, die Entitätstypen in bezug auf einen bestimmten Beziehungstyp einnehmen („spielen“). In EIA/CDIF werden für das Modellieren von Meta-Modellen ausschließlich binäre Beziehungstypen eingesetzt, sodaß auf das Rollenkonzept verzichtet wurde. Allerdings führt dies unter anderem auch dazu, daß einander ausschließende Beziehungstypen, die in EIA/CDIF zwar vorgesehen sind, im EIA/CDIF-Meta-Meta-Modell nur graphisch, nicht aber textuell ausgedrückt werden können. Vergleiche hierzu die Diskussion im Kapitel 0, auf Seite 129ff.

139 Die Begriffe „Entität“, „Beziehung“, „Attribut“, „Objekt“ werden häufig auch in ihren englischen Übersetzungen benutzt, also „Entity“, „Relationship“, „Attribute“ und „Object“. Eine „Meta-Meta-Entität“ kann daher auch als „Meta-Meta-Entity“ bezeichnet werden.

Modell, das für die Erstellung sowohl von Meta-Modellen als auch für das EIA/CDIF-Meta-Meta-Modell selbst benutzt wird.“[140]

Das EIA/CDIF-Meta-Meta-Modell ist reflexiv in dem Sinne, daß es für die Definition des EIA/CDIF-Meta-Meta-Modells selbst herangezogen werden kann.[141] Das EIA/CDIF-Meta-Meta-Modell setzt zur Modellierung ihrer Meta-Meta-Entitätstypen eine Generalisierungshierarchie ein.

2.2.2.3 Begriff „EIA/CDIF-Austausch“ („EIA/CDIF-Transfer“)

„Eine Kombination einer bestimmten Syntax, einer bestimmten Verkodierung dieser Syntax und ein Meta-Modell. Mit anderen Worten, eine vollständige Defintion des Formats und des Inhalts eines Austausches.“[142]

Die EIA/CDIF-Standards definieren eine Syntax „SYNTAX.1“ in [CDIF94d], eine Verkodierung „ENCODING.1“ in [CDIF94e] und ein fundamentales Meta-Modell „Foundation“[143] in [CDIF94f]. Ein EIA/CDIF-Austausch erfolgt mit Hilfe dieser Standards in Form eines Stroms von Zeichen, für gewöhnlich in Form einer Datei. Die Struktur umfaßt unter anderem einen Abschnitt für die Definition von Meta-Modellen beziehungsweise der Referenzierung von EIA/CDIF-standardisierten Meta-Modellen sowie einem Abschnitt für Modelldaten, die die Meta-Objekte des Meta-Modells instanziieren. Für einen EIA/CDIF-Austausch wird eine Erweiterungsmöglichkeit vorgesehen, indem referenzierte und standardisierte Meta-Modelle im Rahmen des Austausches durch eigene Meta-Objekte erweiterbar sind beziehungsweise unabhängig davon ein eigenständiges (proprietäres) Meta-Modell für einen EIA/CDIF-Austausch[144] definiert wird.

---

140 Übersetzung aus dem Englischen von [CDIF94b], Seite 89, Glossareintrag „CDIF Meta-Meta-Model“.

141 Diese Eigenschaft wird im EIA/CDIF-Standard [CDIF97b] – „CDIF Transfer Format – OMG IDL Bindings“ – ausgenutzt, der in Abschnitt 3.2, „Verteilung des EIA/CDIF-Meta-Meta-Modells mit Hilfe von CORBA“ auf Seite 98ff, sowie in Abschnitt 4.1.1.3, „OMG IDL-Definitionen für EIA/CDIF-Meta-Modelle“ auf Seite 171ff, vorgestellt wird.

142 Übersetzung aus dem Englischen von [CDIF94b], Seite 89, Glossareintrag „CDIF Transfer“.

143 Vgl. die Definitionen und Diskussionen dieses Meta-Modells in Abschnitt 4.1.1 auf Seite 155ff weiter unten.

144 In einem solchen Fall muß das im EIA/CDIF-Austausch definierte Meta-Modell zumindest den Meta-Entitätstyp „RootEntity“ und den Meta-Beziehungstyp „RootEntity.*IsRelatedTo*.-RootEntity“ aus dem standardisierten EIA/CDIF-Meta-Modell „Foundation“ (vgl. [CDIF94f]) benutzen, um einen EIA/CDIF-Transfer zu repräsentieren. Die Benutzung der

Fortsetzung folgt auf der nächsten Seite.

Grundsätzlich gilt für Export-Werkzeuge, die einen EIA/CDIF-Austausch erzeugen, die Regel der „maximalen Ausgabe“:[145] es sollen so viele Informationen wie nur möglich in den EIA/CDIF-Austausch aufgenommen werden, unabhängig davon, ob man von einem Import-Werkzeug die Verarbeitung sämtlicher Daten erwarten kann oder nicht.

Für Import-Werkzeuge gilt:[146] es muß sämtliche Meta-Modell-Definitionen und Modelldaten aus einem EIA/CDIF-Austausch extrahieren, auch solche Informationen, die für das Import-Werkzeug ohne Bedeutung sind. Nachdem das Einlesen des EIA/CDIF-Austausches erfolgreich war, kann das Import-Werkzeug jene Konzepte und dessen Instanzen verwerfen, die unbekannt sind.[147]

2.2.2.4 Begriff „Meta-Modell“

„Das Meta-Modell enthält detaillierte Definitionen von Meta-Entitäten, Meta-Beziehungen und Meta-Attributen, deren Instanzen einen aktuellen EIA/CDIF-Transfer repräsentieren.

Das ‘Integrierte EIA/CDIF-Meta-Modell’ (entsprechend der Menge an Definitionen, die die Familie der EIA/CDIF-Standards umfaßt) stellt die Definition sämtlicher Konzepte (Informationstypen) dar, die in einem EIA/CDIF-Austausch enthalten sein können, ohne daß es notwendig wird, den EIA/CDIF-Erweiterungsmechanismus[148] einzusetzen.“[149]

EIA/CDIF-konforme Meta-Modelle *müssen* das fundamentale Meta-Modell „Foundation“[150] einsetzen beziehungsweise referenzieren.[151] Jedes standardi-

---

beiden angeführten fundamentalen Typen kann dabei direkt oder über die Möglichkeit der Subtypbildung erfolgen.

145 Vgl. Kapitel 5, „Exporter Responsibilities“ in [CDIF94b], Seite 28.

146 Vgl. Kapitel 6, „Importer Responsibilities“ in [CDIF94b], Seite 29.

147 Eine solche Situation kann beispielsweise dann auftreten, wenn ein Export-Werkzeug Erweiterungen an einem standardisierten EIA/CDIF-Meta-Modell vorgenommen hat, deren Bedeutung einem Import-Werkzeug unbekannt ist. In einem solchen Fall kann das Import-Werkzeug zumindest jene Daten weiterverarbeiten, die dem standardisierten und daher ihm bekannten Meta-Modell entsprechen.

148 Dieser hier angesprochene Erweiterungsmechanismus erlaubt das Erweitern standardisierter oder das Definieren proprietärer Meta-Modelle in einem EIA/CDIF-Transfer.

149 Übersetzung aus dem Englischen von [CDFI94a], Seite 92, Glossareintrag „Meta-Model“.

150 Vgl. hierzu [CDIF94f].

151 Diese Forderung wird von EIA/CDIF in [CDIF94f], Seite 12, letzter Absatz formuliert. Ein EIA/CDIF-Austausch muß demnach als Minimalforderung das fundamentale Meta-Modell „Foundation“ benutzen beziehungsweise erweitern.

sierte, semantische EIA/CDIF-Meta-Modell definiert üblicherweise jene grundlegenden Konzepte, die einzelnen Modellierungsmethodologien[152] zugrundeliegen, beispielsweise für die Datenflußdiagrammodellierung[153] oder die Datenmodellierung.[154]

Das EIA/CDIF-Meta-Modell setzt zur Modellierung ihrer Meta-Entitätstypen und Meta-Beziehungstypen[155] eine Generalisierungshierarchie ein.

2.2.2.5 Begriff „Gegenstandsbereich“ („Subject Area“)

„Eine Sammlung von miteinander in Beziehung stehender Definitionen von Meta-Objekt-Instanzen. Gegenstandsbereiche überlappen einander, um insgesamt die Integration eines umfassenden Meta-Modells zu gewährleisten.[156] Werkzeuge brauchen nur jene Gegenstandsbereiche zu benutzen, deren Daten importiert beziehungsweise exportiert werden.“[157]

Für jeden Gegenstandsbereich wird ein Meta-Modell definiert.

2.2.2.6 Begriff „Integriertes EIA/CDIF-Meta-Modell“

„Die Beschreibung einer Menge an Konzepten und Notationen, die für die Definition eines Modells benutzt werden. Das ‘Integrierte EIA/CDIF-Meta-Modell’ (abgekürzt: „IMM“) definiert ein Entity-Relationship-Attribut-Modell, das für die Konstruktion und Definition von Modellen im Rahmen der Systementwicklung benutzt wird.“ [158]

Es handelt sich hierbei um ein hypothetisches Modell, das als Mengenvereinigung sämtlicher Definitionen für die Konzepte aller standardisierter EIA/CDIF-

---

152 EIA/CDIF spricht in allgemeiner Form von sogenannten „Subject Areas“ (deutsch: Gegenstandsbereiche), für die Meta-Modelle erstellt werden. Modellierungsgegenstand ist in der Regel eine Modellierungsmethodologie, sodaß in weiterer Folge die entsprechend erstellten Modelle über einen EIA/CDIF-Austausch transportiert werden können.

153 Vgl. den „Data Flow“ Standard in [CDIF96b].

154 Vgl. den „Data Model“ Standard in [CDIF96a].

155 Meta-Beziehungstypen werden als Instanz des Meta-Meta-Entitätstyps „MetaRelationship“ gebildet.

156 Diese Überlappung wird in jedem Fall durch die zwingend vorgeschriebene Nutzung des „Foundation“-Meta-Modells erreicht (vgl. [CDIF94f]).

157 Übersetzung aus dem Englischen von [CDFI94a], Seite 94, Glossareintrag „Subject Area“.

158 Übersetzung aus dem Englischen von [CDFI94a], Seite 88, Glossareintrag „CDIF Integrated Meta-Model“.

Meta-Modelle entsteht.[159] Sobald EIA/CDIF neue Meta-Modelle definiert, die über neu definierte Konzepte verfügen, erweitert sich das Integrierte EIA/CDIF-Meta-Modell entsprechend.

2.2.2.7 Begriff „Arbeits-Meta-Modell ('Working Meta-model')"

„Das Arbeits-Meta-Modell stellt die Definitionen von spezifischen Meta-Objekten dar, die im Modelldatenabschnitt eines EIA/CDIF-Austausches instanziiert werden dürfen. Das Arbeits-Meta-Modell umfaßt jene Meta-Objekte des Integrierten EIA/CDIF-Meta-Modells, die durch Referenzen auf bereits standardisierte EIA/CDIF-Meta-Modelle umfaßt werden sowie Meta-Objekte, die mit Hilfe von Erweiterungen im Meta-Modell-Abschnitt des EIA/CDIF-Austausches definiert wurden."[160]

2.2.2.8 Begriff „Modell"

„Eine bestimmte Sammlung von Software-Engineering-Daten. Dies wird als 'Modell' bezeichnet, da es üblicherweise ein Modell repräsentiert, das im Rahmen eines zu entwickelnden Softwaresystems erstellt wird." [161]

2.3 Austausch von Modelldaten (CDIF-Austausch)

Für den Austausch von Modelldaten, das sind Daten der M1-Schicht, wird einerseits die Kenntnis des entsprechenden Meta-Modells (der M2-Schicht) vorausgesetzt, sowie eine Syntax und Verkodierung der Daten.

In diesem Abschnitt wird ein EIA/CDIF-Austausch auf Basis von „SYNTAX.1" und „ENCODING.1" beispielhaft skizziert, die für den Austausch von Modelldaten in Form eines Stromes von Zeichen festgelegt wurden.[162]

---

159 Im Rahmen dieser Arbeit wird nach Kenntnisstand des Autors zum ersten Mal das „Integrierte EIA/CDIF-Meta-Modell" physisch erarbeitet sowie dokumentiert und zwar auf Basis sämtlicher EIA/CDIF-Meta-Modelle, die bis Anfang 1998 standardisiert wurden.

160 Übersetzung aus dem Englischen von [CDFI94a], Seite 94, Glossareintrag „Working Meta-Model".

161 Übersetzung aus dem Englischen von [CDFI94a], Seite 94, Glossareintrag „Model".

162 Üblicherweise erfolgt ein Austausch in diesem Zusammenhang in Form einer Datei. Vgl. zu diesem Abschnitt auch [CDIF94c], [CDIF94d] und [CDIF94e].

2.3.1 Struktur des EIA/CDIF-Austausches[163]

Ein EIA/CDIF-Austausch weist folgenden, grundlegenden Aufbau auf:

1. einen Prolog (EIA/CDIF-Originalbezeichnung: „Transfer Envelope“, auf deutsch: „Austauschumschlag“) mit Informationen über den EIA/CDIF-Austausch, bestehend aus der Zeichenkette ‘`CDIF`’ („CDIF Signature“), einem Komma, der Zeichenkette ‘`SYNTAX "SYNTAX.1" "02.00.00"`’ („Syntax Identifier“), einem Komma sowie der Zeichenkette ‘`ENCODING "ENCODING.1" "02.00.00"`’ („Encoding Identifier“),[164]
2. einen optionalen Kopfabschnitt („Header“), der für Informationen über die eingesetzten Werkzeuge benutzt werden kann,
3. einen Meta-Modell-Abschnitt, der sowohl über Referenzen auf standardisierte EIA/CDIF-Meta-Modelle als auch über eigenständige Definitionen (Erweiterungen) von Meta-Objekten verfügen kann, und
4. einen Abschnitt mit den Modelldaten, die die Meta-Entitäten beziehungsweise die Meta-Beziehungen des Meta-Modell-Abschnitts instanziieren.

2.3.2 Ein beispielhafter EIA/CDIF-Austausch

In diesem Beispiel wird ein einfaches Meta-Modell für die Entity-Relationship-Modellierung definiert, mit dessen Hilfe ein Modell spezifiziert wird, das folgenden einfachen Zusammenhang für einen Automobilverein abbilden soll: „Jedes Auto muß genau einem Mitglied zugeordnet sein. Ein Mitglied kann, muß aber nicht ein Auto besitzen. Es kann auch mehrere Autos besitzen.“

Der EIA/CDIF-Austausch muß das fundamentale, von EIA/CDIF standardisierte Meta-Modell „Foundation“ einsetzen. Das Beispiels-Meta-Modell mit der Bezeichnung „Mini_ERM“ spezialisiert Konzepte des fundamentalen Meta-Modells und soll über folgende Meta-Objekte verfügen, wobei sämtliche Meta-Attribute zwingend vorgeschrieben sind:[165]

---

163 Vgl. Kapitel 4, „General Structure of the EIA/CDIF-Transfer“, in [CDIF94c] auf Seite 8ff.

164 Die Apostrophe (‘, ’) sind nicht Bestandteil der vordefinierten Zeichenketten, sondern stellen lediglich Begrenzungszeichen dar.

165 EIA/CDIF hat in jahrelanger Arbeit ein mächtiges Meta-Modell für den Bereich der Datenmodellierung erarbeitet, das in [CDIF96a] dokumentiert ist. Mit der Kenntnis dieses Gegenstandbereiches wäre es in Wirklichkeit ausreichend, dieses standardisierte Meta-

Fortsetzung folgt auf der nächsten Seite.

- einen Meta-Entitätstyp „EntTyp“ mit einem Meta-Attribut „Bezeichnung“ und
- einem Meta-Beziehungstyp „BezTyp“ mit den Meta-Attributen „Bezeichnung“, „MinQuellKardinalitaet“, „MaxQuellKardinalitaet“, „MinZielKardinalitaet“ und „MaxZielKardinalitaet“. Ein „BezTyp“ ist eine „Meta-Beziehung“, die jeweils zwei Meta-Entitäten vom Typ „EntTyp“ miteinander assoziiert.[166]

„EntTyp“ soll durch Rechtecke repräsentiert werden, in die die Bezeichnung eingetragen wird. „BezTyp“ soll durch eine auf dem Kopf stehende Rauten dargestellt werden, wobei die Bezeichnung darin eingetragen wird. Darüber hinaus wird vom Rechteck des Quell-“EntTyp“s ein durchgehender Strich zur Raute „BezTyp“ gezogen und von dieser Raute ein Pfeil, dessen Spitze das Rechteck des Ziel-“EntTyp“s berührt. Auf dem Strich beziehungsweise dem Pfeil, werden die entsprechenden Minimum- und Maximumkardinalitäten, durch einen Doppelpunkt getrennt, gezeichnet.

Die Definitionen für das fundamentale Meta-Modell „Foundation“ und die für die Bildung des Meta-Modells „Mini_ERM“ stellt Abbildung 2-1 dar, dessen Meta-Objekte „EntTyp“ und „BezTyp“ direkt die entsprechenden Meta-Objekte des standardisierten und referenzierten EIA/CDIF-Meta-Modell „Foundation“ spezialisieren und mit Meta-Attributen versehen.[167]

Modell zu referenzieren, damit man sämtliche dort definierten Typen direkt auf Modellebene instanziieren kann. In diesem Abschnitt wird lediglich beispielhaft gezeigt, wie eine Meta-Modelldefinition im Meta-Modellabschnitt eines EIA/CDIF-Austausches erfolgen kann und wie auf Modellebene die definierten Meta-Objekte instanziierbar sind.

166 Da für die Kardinalitäten eigene Meta-Attribute vorgesehen werden, können diese im Modell ausdrücklich angegeben werden.

167 Wie weiter unten im Abschnitt 3.1, „Definition des EIA/CDIF-Meta-Meta-Modells“ auf Seite 49ff, erläutert, stellen Linien von oben nach unten angeordnet Hierarchiebeziehungen dar, Pfeile Beziehungstypen zwischen den damit verbundenen Entitätstypen. Die Kardinalitäten werden hierbei in der Minimum-/ Maximumnotation angegeben.

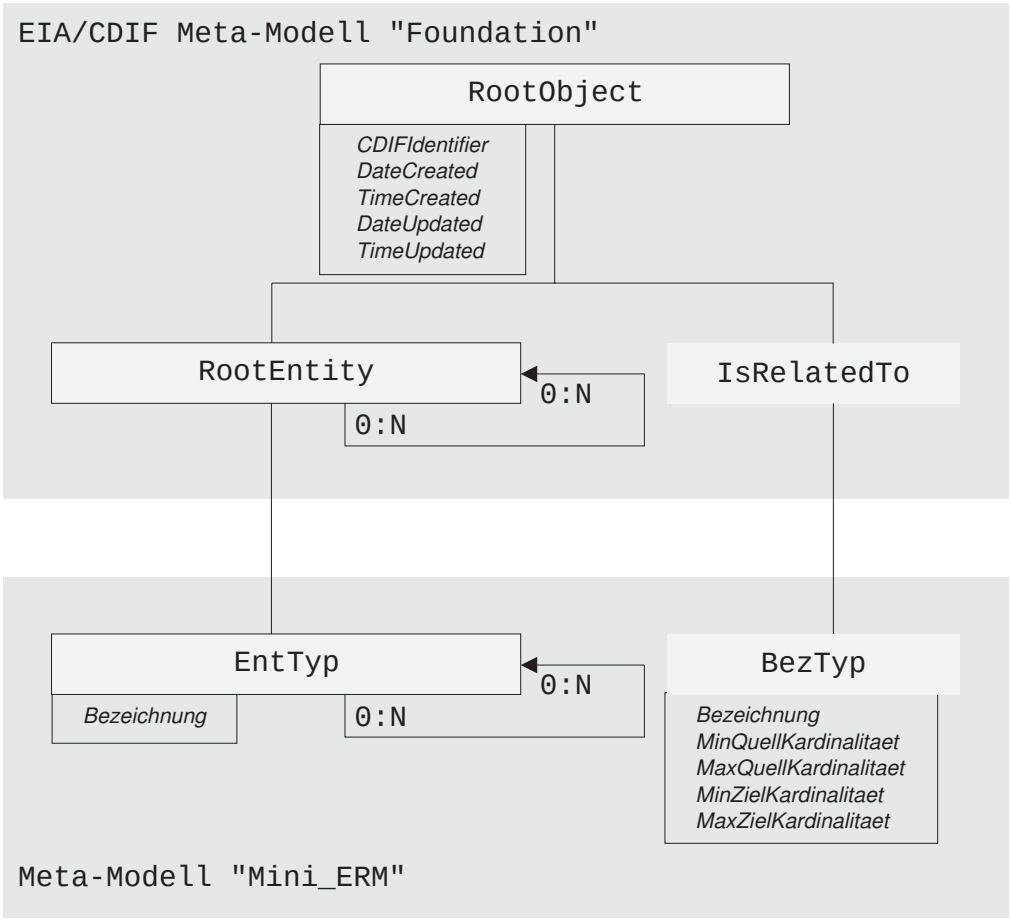


Abbildung 2-1: Das Meta-Modell „Mini_ERM“ und sein Bezug zum EIA/CDIF-Meta-Modell „Foundation“[168]

Das Modell soll folgende Instanzen aufweisen:

- Instanz von „EntTyp“ mit dem Wert „Mitglied“ im Meta-Attribut „Bezeichnung“,
- Instanz von „EntTyp“ mit dem Wert „Auto“ im Meta-Attribut „Bezeichnung“,
- Instanz von „BezTyp“ mit dem Wert „besitzt“ im Meta-Attribut „Bezeichnung“, wobei als Quelle „Mitglied“ und als Ziel „Auto“ dient; „MinQuell-Kardinalitaet“ weist den Wert „1“, „MaxQuellKardinalitaet“ den Wert „1“,

168 Üblicherweise wird in EIA/CDIF die Generalisierungsbeziehung zwischen Meta-Beziehungstypen im Unterschied zu denen zwischen Meta-Entitätstypen nicht graphisch dargestellt. Hier erfolgt diese orthogonale Darstellung aus Gründen des leichteren Verständnisses. Wäre das Meta-Modell „Mini_ERM“ sehr umfangreich, machte es der Übersichtlichkeit wegen auch Sinn, auf die Generalisierungsbeziehungen zwischen Meta-Beziehungstypen in der graphischen Darstellung zu verzichten.

„MinZielKardinalitaet“ den Wert „0“ und „MaxZielKardinalitaet“ den Wert „N“ auf.

Somit kann, entsprechend den oben angegebenen Regeln, das Modell wie in Abbildung 2-2 erstellt werden.

Abbildung 2-2: Modell des Automobilvereins

In der folgenden Abbildung 2-3[169] werden die Instanzen von den Meta-Meta-Objekten „SubjectArea“, „MetaAttribute“, „MetaEntity“ und „MetaRelationship“ durch Surrogate eindeutig gekennzeichnet[170] und im Meta-Meta-Attribut „CDIFMetaIdentifier“ gespeichert.[171] Diese Surrogate werden in weiterer Folge für die Instanziierung von Beziehungstypen als Kurzverweise benutzt. Kommentare werden mit der Zeichenkette ‘#/’ eingeleitet und mit ‘/#’ abgeschlossen.

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

[2]  #| Kommentar: Prolog beendet, Kopfabschnitt |#
     (:HEADER
          (:SUMMARY
            (ExporterName    "Werkzeug XYZ")
            (ExporterVersion "01.10")
            (ExportDate      "1998/04/30")
            (ExportTime      "14:20:54")
          )
     )

[3]  #| Meta-Modell-Abschnitt |#
     (:META-MODEL
     #| fundamentales und standardisiertes Meta-Modell "Foundation"
        wird referenziert; sämtliche entsprechenden Definitionen werden
        daher vorausgesetzt |#
          (:SUBJECTAREAREFERENCE Foundation (:VERSIONNUMBER "01.00") )
```

169 Die Numerierung in eckigen Klammern soll für den Leser das Auffinden der unterschiedlichen Abschnitte eines EIA/CDIF-Transfers mit SYNTAX.1 und ENCODING.1 erleichtern und ist daher nicht Bestandteil des beispielhaften EIA/CDIF-Austausches.

170 Es handelt sich um die Werte „SA_1“, „ME_1“, „MR_1“, „MA_1“, „MA_2“, „MA_3“, „MA_4“, „MA_5“ und „MA_6“.

171 Vgl. die Definitionen des Meta-Meta-Entitätstyps „MetaObject“ des EIA/CDIF-Meta-Meta-Modells in Abschnitt 3.1.4.1 auf Seite 59ff.

```
#| neues Meta-Modell wird definiert |#
     (SubjectArea SA_1
       (Name *Mini_ERM*)
       (VersionNumber "01.00")
       (Description #[Dies ist ein Beispiels-Meta-Modell.]#)
     )

#| neue Meta-Entität wird definiert |#
     (MetaEntity ME_1
       (Name *EntTyp*)
       (Description #[Repräsentiert einen Entitätstyp.]#)
     )
#| neue Meta-Entität ist Subtyp von "RootEntity"[172] |#
     (AttributableMetaObject.HasSubtype.AttributableMetaObject
     2 ME_1)

#| Meta-Attribut wird zunächst definiert und dann zugeordnet |#
     (MetaAttribute MA_1
       (Name *Bezeichnung*)
       (Description #[Nimmt den Namen auf.]#)
       (DataType <String>)
       (Length "64")
       (IsOptional -False-)
     )

#| neues Meta-Attribut gehört zu Meta-Entität "EntTyp" |#
     (MetaAttribute.IsLocalMetaAttributeOf.AttributableMetaObject
       MA_1 ME_1)

#| neue Meta-Beziehung wird defniert |#
     (MetaRelationship MR_1
       (Name *BezTyp*)
       (Description #[Repräsentiert einen Beziehungstyp]#)
       (MinSourceCard "0")
       (MaxSourceCard "N")
       (MinDestCard "0")
       (MaxDestCard "N")
     )
#| neue Meta-Beziehung ist Subtyp von
     "RootEntity.IsRelatedTo.RootEntity"[173] |#
```

[172] „RootEntity“ ist im referenzierten Meta-Modell „Foundation“ definiert und besitzt im Meta-Meta-Attribut „CDIFMetaIdentifier“ den Wert zwei, vgl. hierzu [CDIF94f], Seite 20, beziehungsweise die Ausführungen im Abschnitt 4.1.1.1.2 auf Seite 163ff.

[173] „RootEntity.*IsRelatedTo*.RootEntity“ ist im referenzierten Meta-Modell „Foundation“ definiert und besitzt im Meta-Meta-Attribut „CDIFMetaIdentifier“ den Wert drei, vgl. hierzu

Fortsetzung folgt auf der nächsten Seite.

```
     (AttributableMetaObject.HasSubtype.AttributableMetaObject
     3 MR_1)
#| neue Meta-Beziehung assoziiert zwei Meta-Entitäten "EntTyp" |#
     (MetaRelationship.HasSource.MetaEntity MR_1 ME_1)
     (MetaRelationship.HasDestination.MetaEntity MR_1 ME_1)

#| Meta-Attribute werden zunächst definiert und dann zugeordnet |#
     (MetaAttribute MA_2
       (Name *Bezeichnung*)
       (Description #[Nimmt den Namen auf.]#)
       (DataType <String>)
       (Length "64")
       (IsOptional -False-)
     )

     (MetaAttribute MA_3
       (Name *MinQuellKardinalitaet*)
       (Description
         #[Beinhaltet den Minimumwert der Quell-Kardinalität.]#
       )
       (DataType <String>)
       (Length "10")
       (IsOptional -False-)
     )

     (MetaAttribute MA_4
       (Name *MaxQuellKardinalitaet*)
       (Description
         #[Beinhaltet den Maximalwert der Quell-Kardinalität.]#
       )
       (DataType <String>)
       (Length "10")
       (IsOptional -False-)
     )

     (MetaAttribute MA_5
       (Name *MinZielKardinalitaet*)
       (Description
         #[Beinhaltet den Minimumwert der Ziel-Kardinalität.]#
       )
       (DataType <String>)
       (Length "10")
       (IsOptional -False-)
```

[CDIF94f], Seite 21, beziehungsweise die Ausführungen im Abschnitt 4.1.1.1.3 auf Seite 164ff.

```
        )

        (MetaAttribute MA_6
          (Name *MaxZielKardinalitaet*)
          (Description
            #[Beinhaltet den Maximalwert der Ziel-Kardinalität.]#
          )
          (DataType <String>)
          (Length "10")
          (IsOptional -False-)
        )

        (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)

    #| Meta-Entitäten, Meta-Beziehungen und Meta-Attribute werden
       dem neu definierten Meta-Modell zugeordnet|#
       (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)
    )  #| Ende Meta-Modell-Abschnitt |#

[4] #| Modell-Abschnitt |#
    (:MODEL
       (EntTyp ET_1
         (Bezeichnung "Mitglied")
       )

       (EntTyp ET_2
         (Bezeichnung "Auto")
       )

       (EntTyp.BezTyp.EntTyp R_1 ET_1 ET_2
         (Bezeichnung "besitzt")
```

```
            (MinQuellKardinalitaet "1")
            (MaxQuellKardinalitaet "1")
            (MinZielKardinalitaet "0")
            (MaxZielKardinalitaet "N")
          )
    )
```

Abbildung 2-3: EIA/CDIF-Austausch eines Modells des Automobilvereins mit Hilfe des Meta-Modells „Mini_ERM“ unter Einsatz der EIA/CDIF-Standards „SYNTAX.1“ und „ENCODING.1“

3 Das EIA/CDIF-Meta-Meta-Modell

In diesem Kapitel wird zunächst das EIA/CDIF-Meta-Meta-Modell entsprechend dem EIA/CDIF „Interim Standard“ vorgestellt und kommentiert. Daran anschließend wird diskutiert, wie das Meta-Meta-Modell einerseits in einem relationalen Datenbankmodell und andererseits in einer objektorientierten Programmiersprache spezifiziert und implementiert[174] werden kann.

3.1 Definition des EIA/CDIF-Meta-Meta-Modells

Das EIA/CDIF-Meta-Meta-Modell wird in Form eines erweiterten Entity-Relationship-Diagrammes[175] in Abbildung 3-1[176] dargestellt und repräsentiert ein konzeptionelles Datenmodell[177]. Drei grundlegende „Elemente“[178], manchmal auch als „Objekttypen“ bezeichnet, werden hierbei vorausgesetzt und sind daher fundamental für die Bildung des Meta-Meta-Modells:

1. Entitätstyp,
2. Beziehungstyp und
3. Attribut.

Ein Beziehungstyp ist als binäre Relation[179] definiert und assoziiert demgemäß zwei Entitätstypen. Jeder Entitätstyp und Beziehungstyp kann darüber hinaus Attribute aufweisen. Attribute erlauben die Assoziierung eines Bezeichners mit einem Wert. Die Ausprägungen, die Attribute annehmen können, werden in der

---

174 Die Dokumentationen von beispielhaften Implementierungen finden sich im Anhang in den Abschnitten 6.2.1, „Implementierung des EIA/CDIF-Meta-Meta-Modells in ORACLE“ auf Seite 424ff, und 6.2.2, „Implementierung des EIA/CDIF-Meta-Meta-Modells in Object Rexx“ auf Seite 449ff.

175 Vgl. beispielsweise [BaCeNa92], [ElmNav89] oder [Flat96b].

176 Vgl. Seite 53 weiter unten.

177 [LeMaHi95] bezeichnen dies als „konzeptuelle“ Datenmodelle. Vgl. ebenda auch die Ausführungen über die unterschiedlichen Modellbegriffe in der Wirtschaftsinformatik im Abschnitt 2.4, „Systementwicklungsorientierte Modelle“ auf Seite 120ff.

178 Es wird manchmal auch der Begriff „Konzept“ dafür benutzt.

179 Die Tatsache, daß für die Definition des Meta-Meta-Modells selbst und in weiterer Folge für Meta-Modelle lediglich binäre Beziehungstypen erlaubt sind, stellt keine Einschränkung in bezug auf den Tausch von nichtbinären Beziehungen innerhalb von Modelldaten dar, wenn diese in ihren entsprechenden Meta-Modellen derartige nicht-binäre Beziehungen vorgesehen haben. Ein Beispiel dafür stellt das standardisierte EIA/CDIF Meta-Modell „Datenmodellierung“ dar, wie es in [CDIF96b] festgelegt und weiter unten im Abschnitt 4.1.3.2, „DMOD – EIA/CDIF-Meta-Modell für die Datenmodellierung“ auf Seite 241ff, beschrieben ist.

Version der EIA/CDIF-Standards von 1994 in Form von „EIA/CDIF-Datentypen“[180] vorgegeben. Typen können instanziiert werden, sodaß als Ergebnis davon *Instanzen* beziehungsweise *Objekte* vom entsprechenden Typ verfügbar sind.[181]

Auf der Ebene des Meta-Meta-Modells werden diese drei Elemente des Modells als „Meta-Meta-Entität“ (englisch: „meta-meta-entity“), „Meta-Meta-Beziehung“ (englisch: „meta-meta-relationship“) und „Meta-Meta-Attribut“ (englisch: „meta-meta-attribute“) bezeichnet,[182] auf der Ebene von Meta-Modellen als „Meta-Entität“ (englisch: „meta-entity“), „Meta-Beziehung“ (englisch: „meta-relationship“) und „Meta-Attribut“ (englisch: „meta-attribute“), auf der Ebene der Modelldaten als „Entität“ (englisch: „entity“), „Beziehung“ (englisch: „relationship“) und „Attribut“ (englisch: „attribute“).[183, 184]

Als weitere Konsequenz werden im EIA/CDIF-Standard diese drei Elemente dokumentiert. Im *Meta-Meta-Modell* werden Meta-Meta-Entitäten, Meta-Meta-Beziehungen und Meta-Meta-Attribute mit den entsprechenden – in Tabelle 3-1 angeführten – Attributen beschrieben.

---

180 Vgl. den entsprechenden Abschnitt in 3.1.2 auf Seite 54ff.

181 Nachdem Typen – wie im vorhergehenden Kapitel ausgeführt – als Intension aufgefaßt werden, stellen ihre Instanzen beziehungsweise Objekte eine entsprechende Extension dar.

182 Nachdem das Meta-Meta-Modell reflexiv ist und es als Folge davon selbst damit instanzierbar wird, kommt es dazu, daß die darin definierten Meta-Meta-Entitätstypen (Meta-Meta-Beziehungstypen) selbst als Instanzen/Objekte verfügbar werden und demzufolge als „Meta-Meta-Entitäten“ (Meta-Meta-Beziehungen) bezeichnet werden. Diese reflexive Eigenschaft wird beispielsweise auch in der OMG IDL-Definition dafür ausgenutzt (vgl. Abschnitt 3.2, „Verteilung des EIA/CDIF-Meta-Meta-Modells mit Hilfe von CORBA“ auf Seite 98ff weiter unten).
Instanzen von Meta-Meta-Entitäten (Meta-Meta-Beziehungen) werden als „Meta-Entitäten“ („Meta-Beziehungen“) bezeichnet und sind selbst Typen, deren Extensionen die M1-Schicht bilden. Entsprechend heißen sie im EIA/CDIF-Standard auch „Meta-Entitätstypen“ („Meta-Beziehungstypen“).

183 Dies folgt der Konvention, die im EIA/CDIF-Standard eingeführt wurde; somit werden Entitäts*typen* und Beziehungs*typen* in der Folge auf die Begriffe „Entität“ und „Beziehung“ verkürzt. Im Standard werden die Bezeichnungen „Entität“ und „Beziehung“ im Zusammenhang mit den unterschiedlichen Modellierungsebenen verwendet, indem entsprechend der Modellierungsebene „Meta“ (für Meta-Modelle) oder „Meta-Meta“ (für das Meta-Meta-Modell) den Begriffen vorangestellt wird.

184 Eine generische Bezeichnung für Meta-Meta-Entitäten (Meta-Entitäten) und Meta-Meta-Beziehungen (Meta-Beziehungen) stellt der Begriff „Meta-Meta-Objekt“ (englisch: „meta-meta-object“; „Meta-Objekt“, englisch: „meta-object“) dar.

Name des Attributs[185]	**Bedeutung**	**Beschreibung für**
Name	Name	Meta-Meta-Entität, Meta-Meta-Beziehung, Meta-Meta-Attribut
SubtypeOf	Menge der Typen	Meta-Meta-Entität, Meta-Meta-Beziehung
SupertypeOf	Menge der untergeordnetenTypen	Meta-Meta-Entität, Meta-Meta-Beziehung
Description	Beschreibung des Zwecks	Meta-Meta-Entität, Meta-Meta-Beziehung, Meta-Meta-Attribut
Aliases	Alias-Namen, sofern verfügbar	Meta-Meta-Entität, Meta-Meta-Beziehung, Meta-Meta-Attribut
Usage	Verwendungshinweise	Meta-Meta-Entität, Meta-Meta-Beziehung, Meta-Meta-Attribut
Constraints	Einschränkungen, Nebenbedingungen	Meta-Meta-Entität, Meta-Meta-Beziehung, Meta-Meta-Attribut
Type	„Kernel“, „Characteristic“, „Associative“	Meta-Meta-Entität
MinSourceCard	Minimalwert für Quellkardinalität	Meta-Meta-Beziehung
MaxSourceCard	Maximalwert für Quellkardinalität	Meta-Meta-Beziehung
MinDestCard	Minimalwert für Zielkardinalität	Meta-Meta-Beziehung
MaxDestCard	Maximalwert für Zielkardinalität	Meta-Meta-Beziehung
DataType	CDIF-Datentyp	Meta-Meta-Attribut
Domain	Wertemenge, wenn „DataType“ vom Typ „Enumerated“ ist	Meta-Meta-Attribut

185 In dieser Arbeit werden sämtliche EIA/CDIF-Bezeichner in der Schreibweise so beibehalten, wie sie in den Standards angeführt sind. Dadurch soll ein entsprechend einfacher Einstieg in die EIA/CDIF-Standards ermöglicht werden.

Name des Attributs[185]	Bedeutung	Beschreibung für
Length	Länge, wenn „DataType“ vom Typ „String“ ist	Meta-Meta-Attribut
IsOptional	„Wahr“ (englisch: „True“), wenn Wert für Attribut optional ist, „Falsch“ (englisch: „False“) sonst	Meta-Meta-Attribut

Tabelle 3-1: Übersicht über die Attribute für die Beschreibung von Meta-Meta-Entitäten, Meta-Meta-Beziehungen und Meta-Meta-Attributen

Die für die Beschreibung der Meta-Meta-Entitäten, Meta-Meta-Beziehungen und Meta-Meta-Attribute benutzten Attribute entsprechen in ihrer Bezeichnung und in ihren Namen im großen und ganzen jenen Meta-Attributen, die für Meta-Entitäten und Meta-Beziehungen verwendet werden.[186]

In EIA/CDIF-Diagrammen werden die Entitätstypen als Rechtecke und die Beziehungstypen als gerichtete Pfeile eingezeichnet. Sowohl Entitätstypen als auch Beziehungstypen müssen Namen tragen und können Attribute aufweisen. Abbildung 3-1 stellt das EIA/CDIF-Meta-Meta-Modell in Form eines Diagramms dar. Die gerichteten Pfeile dienen dabei als Hinweise dafür, in welche Richtung die Beziehungstypen zu lesen sind, beispielsweise gilt für den Beziehungstyp „IsUsedIn“ in Abbildung 3-1 die Leseweise „CollectableMetaObject.*IsUsedIn.*-SubjectArea“.[187]

186 Eine Ausnahme stellt das Fehlen eines Surrogats vom EIA/CDIF-Datentyp „Identifier“ für die einzelnen Meta-Meta-Objekte des Meta-Meta-Modells dar und zum anderen das Fehlen von Super-/Subtypmengen für Meta-Meta-Beziehungen. Das Fehlen der letzteren im Standard ist vermutlich dadurch motiviert gewesen, daß im Meta-Meta-Modell Meta-Meta-Beziehungen nicht spezialisiert werden.

187 In diesem Beispiel wird die Meta-Meta-Entität „CollectableMetaObject“ auch als „Quell(e)“ (englisch: „Source“) und die Meta-Meta-Entität „SubjectArea“, auf die der Pfeil der Meta-Meta-Beziehung weist, als „Ziel“ (englisch: „Destination“) bezeichnet.

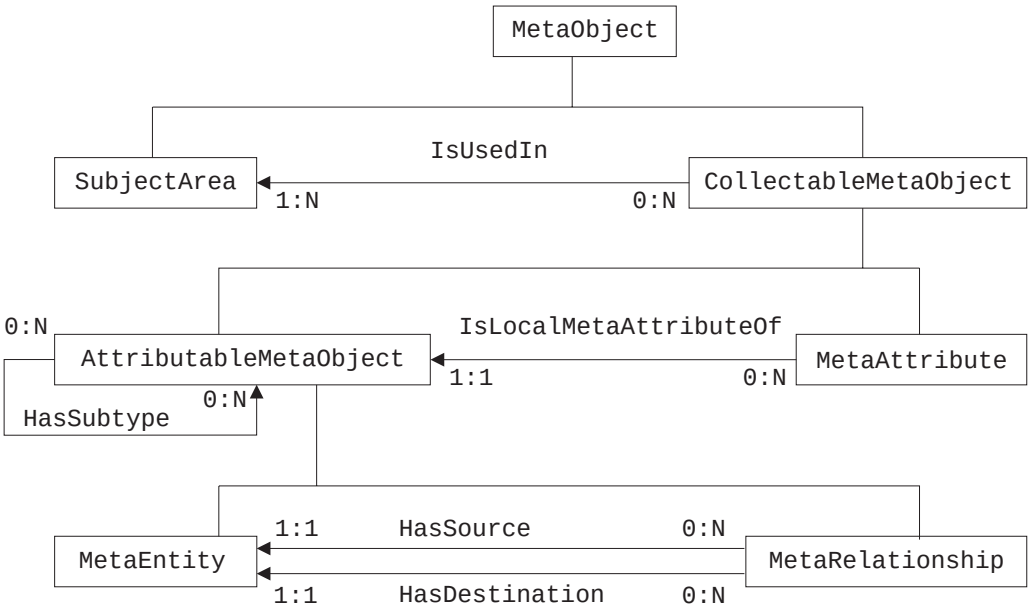


Abbildung 3-1: Das EIA/CDIF-Meta-Meta-Modell[188]

Der Wurzelentitätstyp des EIA/CDIF-Meta-Meta-Modells, in der Nomenklatur von EIA/CDIF eine „Meta-Meta-Entität“, wird durch den Entitätstyp „MetaObject“ gebildet. In der Folge wird diese Meta-Meta-Entität spezialisiert[189], wobei die dadurch entstandenen Meta-Meta-Entitäten zueinander über Meta-Meta-Beziehungen assoziiert werden können.

Im EIA/CDIF-Meta-Meta-Modell sind die Meta-Meta-Entitäten „MetaObject“ und „CollectableMetaObject“ abstrakt[190]. Alle anderen Meta-Meta-Entitäten[191] werden durch Meta-Modelle instanziiert und beinhalten entsprechend die Definitionen für die Meta-Modellbezeichnung (Instanz der Meta-Meta-Entität

188 Abbildung 6-1 auf Seite 423 beinhaltet auch die deutschen Übersetzungen für die dargestellten Meta-Meta-Objekte, die weiter unten für die englischen Begriffe eingeführt werden.

189 Spezialisierungsbeziehungen werden durch Linien dargestellt, die keine Pfeile aufweisen. Hierbei wird optisch die Hierarchie dadurch dargestellt, daß die Supertypen über den Subtypen angeschrieben werden.

190 „Abstrakt“ bedeutet in diesem Zusammenhang, daß diese Entitätstypen nicht *direkt* instanziiert werden. Es ist allerdings denkbar, daß manche Implementierungen des Meta-Meta-Modells ein Objektmodell benutzen (beispielsweise das von Smalltalk oder von Object Rexx), das indirekt auch diese Entitätstypen instanziiert. Auch der Implementierungsvorschlag – in dieser Arbeit für die Repräsentation des Meta-Meta-Modells in relationalen Datenbanksystemen weiter unten – instanziiert diese Entitätstypen.

191 Die Meta-Meta-Entität „AttributableMetaObject“ besitzt nur eine *einzige* Instanz, nämlich „RootObject“ und wird in [CDIF94f] definiert.

„SubjectArea“), Meta-Entitäten, Meta-Beziehungen und Meta-Attribute dieser Meta-Modelle[192].

3.1.1 Namenskonventionen

In den EIA/CDIF-Standards werden sämtliche Namen in Groß- und Kleinschreibung vergeben. Bei zusammengesetzten Wörtern wird der erste Buchstabe eines jeden Wortes mit einem Großbuchstaben begonnen, der Rest in Kleinbuchstaben geschrieben. Akronyme behalten ihre Großschreibung bei.[193]

Beziehungstypen werden voll qualifiziert, indem durch Punkte getrennt, einerseits dem Namen des Beziehungstyps der Name des Quellentitätstyps vorangestellt und andererseits der Name des Zielentitätstyps[194] angefügt wird.

Die Namen von Attributen müssen lediglich innerhalb von Entitäts- beziehungsweise Beziehungstypen eindeutig sein. Attribute können *vollqualifiziert* werden, indem durch einen Punkt getrennt der vollqualifizierte Name des entsprechenden Typs vorangestellt wird.

Die Namen von Meta-Objekten können nur in der vollqualifizierten Form Primärschlüsselcharakter aufweisen, sodaß die entsprechenden Meta-Objekte dadurch eindeutig identifizierbar sind.

3.1.2 EIA/CDIF-Datentypen

Für die Erstellung von Meta-Modellen und den Austausch von Modelldaten wurden eine Reihe von Datentypen vordefiniert, die für die Dokumentation des Meta-Meta-Modells, sowie für die Definition von Meta-Modellen und den Tausch

---

192 Somit werden in den EIA/CDIF-Standards beispielsweise Meta-Entitäten als Instanzen der Meta-Meta-Entität „Meta-Entity“ aufgefaßt. Andererseits stellen dieselben Meta-Entitäten in bezug auf die Modelldaten Entitätstypen dar. Ein Meta-Modell stellt daher einerseits eine Extension für die Intension des Meta-Meta-Modells dar und ist selbst gleichzeitig die Intension für die Modelldaten (Extension von Meta-Modellen). Vgl. hierzu beispielsweise die Ausführungen in [Wede92], insbesondere die Ausführungen über Meta- und Objektsprachen (z.B. Seite 80), sowie die Diskussion über „Sinn“ (Intension, Inhalt) und „Geltung“ (Extension, Umfang) (z.B. Seite 82ff, Seite 118ff).

193 EIA/CDIF-Namen sind vom vordefinierten Datentyp „Identifier“. Vgl. Seite 56. Bei Meta-Meta-Beziehungen und Meta-Beziehungen wird ein „vollqualifizierter“ Name dadurch gebildet, daß der Name der Quellentität durch einen Punkt mit dem Namen der Beziehung und durch einen weiteren Punkt mit dem Namen der Zielentität verbunden wird. Die „ZielEntität“ findet sich auf der Seite des abgebildeten Pfeilkopfes.

194 Der Zielentitätstyp findet sich auf der Seite des Pfeiles.

von Modelldaten eingesetzt werden. Für diese Datentypen wird auch eine entsprechende Syntax in [CDIF94d] und Kodierung in [CDIF94e] festgelegt.

Die einzelnen Datentypen werden in den folgenden Unterabschnitten in alphabetischer Reihenfolge vorgestellt und kurz charakterisiert.[195]

3.1.2.1 EIA/CDIF-Datentyp „Bitmap“

Der Datentyp „Bitmap“ besteht aus einem Feld von „RGB“-Triplets und einer Angabe über die Höhe und Breite. Ein „RGB“-Triplet setzt sich aus drei Werten zusammen, die für die Intensität der Farben Rot, Grün und Blau jeweils im Wertebereich von 0 bis 255 stehen.

3.1.2.2 EIA/CDIF-Datentyp „Boolean“

Der Datentyp „Boolean“ kann die Werte „Wahr“ (englisch: „True“) und „Falsch“ (englisch: „False“) annehmen.

3.1.2.3 EIA/CDIF-Datentyp „Date“

Der Datentyp „Date“ kann als Wert jedes gültige Datum annehmen. Das Datum kann als absolut oder relativ (positiv oder negativ) angesehen werden, abhängig von der Angabe eines entsprechenden Indikators.

3.1.2.4 EIA/CDIF-Datentyp „Enumerated“

Der Datentyp „Enumerated“ erlaubt die Definition einer Wertemenge.[196] Jeder Wert in der Menge muß vom EIA/CDIF-Datentyp „Identifier“ sein.

3.1.2.5 EIA/CDIF-Datentyp „Float“

Der Datentyp „Float“ erlaubt als Wert Fließkommazahlen, die über maximal 16 Ziffern verfügen dürfen und innerhalb des Wertbereichs von 10^{-1023} bis 10^{1023} liegen müssen.

---

195 Vgl. unter anderem auch [CDIF94b], [CDIF94d] und [CDIF94e].

196 Dieser Datentyp findet bei der Definition von Meta-Attributen Verwendung, deren Werteausprägungen aus einer Menge von frei aber vordefinierten Werten zu wählen ist.

3.1.2.6 EIA/CDIF-Datentyp „Identifier“

Der Datentyp „Identifier“ kann aus bis zu 32 Zeichen bestehen. Die Wertemenge für die Zeichen setzt sich aus den englischen Groß- und Kleinbuchstaben, sowie den Ziffern und zusätzlich dem Bindestrich („-“) sowie dem Unterstrich („_“) zusammen. Ein „Identifier“ muß mit einem alphanumerischen Zeichen beginnen.

3.1.2.7 EIA/CDIF-Datentyp „Integer“

Der Datentyp „Integer“ als Ganzzahl verfügt über einen definierten Wertebereich von -2.147.483.648 bis 2.147.483.647.

3.1.2.8 EIA/CDIF-Datentyp „IntegerList“

Der Datentyp „IntegerList“ setzt sich aus einer geordneten Liste von Daten vom Typ „Integer“ zusammen.

3.1.2.9 EIA/CDIF-Datentyp „Point“

Der Datentyp „Point“ setzt sich aus drei Werten vom Typ „Integer“ zusammen, die einen Punkt in einem dreidimensionalen Raum anhand der Koordinaten „x“, „y“ und „z“ festlegen.[197]

3.1.2.10 EIA/CDIF-Datentyp „PointList“

Der Datentyp „PointList“ setzt sich aus einer geordneten Liste von Daten vom Typ „Point“ zusammen.

3.1.2.11 EIA/CDIF-Datentyp „String“

Der Datentyp „String“ setzt sich aus einer Zeichenkette von „druckbaren“[198] Zeichen im Umfang von 1 bis 1024 Zeichen[199] zusammen.

---

197 Der Ursprung *(0, 0, 0)* im dreidimensionalen EIA/CDIF-Koordinatennetz findet sich links *(x=0)*, oben *(y=0)* und vorne *(z=0)*. Positive Einheiten von *x* bewegen die Koordinate vom Ursprung aus nach rechts, positive Einheiten von *y* nach unten und positive Einheiten von *z* nach hinten, negative Einheiten in die entsprechende Gegenrichtung.

198 Die exakte Definition findet sich in der BNF von [CDIF94e], Seite 18ff.

3.1.2.12 EIA/CDIF-Datentyp „Text“

Der Datentyp „Text“ setzt sich aus beliebigen Zeichen zusammen. Im Unterschied zum Datentyp „String“ wird die Länge (maximale Anzahl der Zeichen) nicht im vorhinein eingeschränkt.[200]

3.1.2.13 EIA/CDIF-Datentyp „Time“

Der Datentyp „Time“ kann jede gültige Uhrzeit annehmen. Die Zeit kann als absolut oder relativ (positiv oder negativ in bezug auf UTC[201] oder lokale Zeit) angesehen werden, abhängig von der Angabe eines entsprechenden Indikators.

3.1.3 Meta-Meta-Attribute

Meta-Meta-Attribute werden für Meta-Meta-Entitäten und Meta-Meta-Beziehungen definiert. Meta-Meta-Attribute werden in weiterer Folge mit den Attributen beschrieben, die in Tabelle 3-2 dafür angeführt sind.

Name des Attributs	Bedeutung
Name	Name des Meta-Meta-Attributs
Description	Beschreibung des Zwecks
Usage	Verwendungshinweise
Aliases	Alias-Namen, sofern verfügbar
Constraints	Einschränkungen, Nebenbedingungen
DataType	CDIF-Datentyp
Domain	Wertemenge, wenn „DataType“ vom Typ „Enumerated“ ist

199 Bei der Definition dieses Datentyps wird für die Angabe der maximalen Anzahl an erlaubten Zeichen das Meta-Meta-Attribut „Length“ benutzt.

200 Allerdings werden in [CDIF94d] und [CDIF94e] Daten vom Typ „Text“ für den Modelldatenaustausch in kleinere Einheiten, den sogenannten „TextStrings“ aufgeteilt, die maximal aus 1.024 Zeichen bestehen dürfen (vgl. BNF in [CDIF94d], Seite 13). Längere Texte werden durch eine Liste von „TextStrings“ dargestellt (vgl. BNF in [CDIF94c], Seite 40, Regeln für „TextValue“ und „TextValueTail“).

201 „UTC“ ist die französische Abkürzung für den englischen Begriff „Universal Coordinated Time“ der CCIR (englisch für: „Consultative Committee on Intenational Radio“) und entspricht der „Greenwich Mean Time“, abgekürzt: GMT.

Name des Attributs	Bedeutung
Length	Länge, wenn „DataType“ vom Typ „String“ ist
IsOptional	„Wahr“ (englisch: True), wenn Wert[202] für Attribut optional ist, „Falsch“ (englisch: False) sonst

Tabelle 3-2: Attribute für die Beschreibung von Meta-Meta-Attributen[203]

3.1.4 Meta-Meta-Entitäten

In diesem Abschnitt werden die Meta-Meta-Entitäten des Meta-Meta-Modells detailliert vorgestellt. Die Reihenfolge der Vorstellung erfolgt entsprechend der in Abbildung 3-1[204] abgebildeten Hierarchie, die Ebene für Ebene von der Wurzel ausgehend von oben nach unten (und von links nach rechts) durchwandert wird.[205]

Die Meta-Meta-Entitäten sind im EIA/CDIF-Standard mit denselben Attributen[206] dokumentiert, wie sie für Meta-Entitäten in der Meta-Meta-Entität „MetaEntity“ in Form von Meta-Meta-Attributen definiert werden. Tabelle 3-3 führt die Attribute an, die Meta-Meta-Entitäten beschreiben.

202 In weiterer Folge werden die englischen Bezeichnungen „True“ für Wahr und „False“ für Falsch benutzt.

203 Die Attribute werden in derselben Reihenfolge wie im Standard (vgl. [CDIF94b]) angeführt.

204 Siehe Seite 53.

205 Dies weicht von der Reihenfolge der EIA/CDIF-Standards ab, die immer eine alphabetische ist. Für das Meta-Meta-Modell wurde diese Reihenfolge gewählt, da dadurch die Vererbung von Meta-Meta-Attributen entlang der Generalisierungshierarchie leichter nachvollziehbar erscheint.

206 Die einzige Ausnahme bildet hier das nicht ausdrücklich angeführte Attribut zur eindeutigen Identifizierung der Meta-Meta-Entitäten. Für Meta-Entitäten ist dies das Meta-Meta-Attribut „CDIFMetaIdentifier“, das für die Meta-Meta-Entität „MetaObject“ definiert ist.
Für den Austausch von Modelldaten werden derartige eindeutige Bezeichner im fundamentalen Meta-Modell „Foundation Subject Area“ (vgl. [CDIF94f]) durch die Definition des Meta-Attributs „CDIFIdentifier“ für die Meta-Entität „RootObject“ ermöglicht. Dadurch ist für einen EIA/CDIF-Transfer gewährleistet, daß sämtliche getauschte Objekte über eindeutig identifizierende Werte verfügen. Nachdem das Meta-Meta-Modell selbst nie getauscht wird, sondern lediglich sämtlichen Meta-Modellen zugrundegelegt ist, benötigt man bei der Darstellung des Meta-Meta-Modells keine eindeutigen Bezeichner über die Namen der Meta-Meta-Objekte hinaus.

Name des Attributs	Bedeutung
Name	Name der Meta-Meta-Entität
SubtypeOf	Menge der übergeordneten Meta-Meta-Entitäten
SupertypeOf	Menge der untergeordneten Meta-Meta-Entitäten
Description	Beschreibung des Zwecks
Aliases	Synonyme, sofern verfügbar
Usage	Verwendungshinweise
Constraints	Einschränkungen, Nebenbedingungen
Type	Wert aus der Wertemenge {„Kernel“, „Characteristic“, „Associative“}[207]

Tabelle 3-3: Attribute für die Beschreibung von Meta-Meta-Entitäten[208]

In den folgenden Unterabschnitten werden die einzelnen Meta-Meta-Entitäten vorgestellt und es wird prägnant erklärt, welche Stellung sie im Meta-Meta-Modell einnehmen. Sofern für Meta-Meta-Entitäten Meta-Meta-Attribute[209] definiert sind, werden sie anschließend mit ihren detaillierten Definitionen vorgestellt.

Die Erläuterungen erfolgen im Vergleich zum Standard in kommentierter und prägnanter Form und sind daher nicht notwendigerweise vollständig.

3.1.4.1 Meta-Meta-Entität „MetaObject“

Die Meta-Meta-Entität „MetaObject“ [210] (deutsch:[211] „MetaObjekt“[212]) bildet die Wurzel des EIA/CDIF-Meta-Meta-Modells. Aufgrund der objektorientierten Ei-

207 Vgl. die Definition des Meta-Meta-Attributs „Type“ für die Meta-Meta-Entität „Meta-Entity“, die die Wertemenge für die Typisierung von Meta-Entitäten im Meta-Meta-Modell festlegt.

208 Die Attribute werden in derselben Reihenfolge wie im Standard (vgl. [CDIF94b]) angeführt.

209 Die Vorstellung allfälliger Meta-Meta-Attribute erfolgt alphabetisch, entsprechend dem Standard (vgl. [CDIF94b]).

210 Vgl. [CDIF94b], S.49ff.

211 Die deutschen Übersetzungen folgen den EIA/CDIF-Namenskonventionen.

212 Somit können sämtliche Meta-Meta-Entitäten im EIA/CDIF-Meta-Meta-Modell, die als Subklasse zu „MetaObject“ definiert werden, entsprechend den CDIF-Konventionen synonym als „MetaObjects“ beziehungsweise in deutsch als „MetaObjekte“ bezeichnet werden, dies unter anderem deshalb, da jede Spezialisierung eines Objekts in jedem Fall auch vom Typ des Objekts ist, das spezialisiert wurde.

genschaften des erweiterten Entity-Relationship Modells erben sämtliche Subklassen alle Meta-Meta-Attribute, die für diese fundamentale und omnipotente Meta-Meta-Entität definiert sind.

Name des Attributs	Bedeutung
Name	***MetaObject***
SubtypeOf	–
SupertypeOf	*CollectableMetaObject* *SubjectArea*
Description	*Wurzel des Meta-Meta-Modells, definiert jene Meta-Meta-Attribute, die allen Meta-Meta-Entitäten gemein sind.*
Usage	*Diese Meta-Meta-Entität ist abstrakt und soll daher nicht instanziiert werden.*
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 3-4: Meta-Meta-Entität „MetaObject“

Folgende Meta-Meta-Attribute sind für die Meta-Meta-Entität „MetaObject“ festgelegt:

1. „Aliases“,
2. „CDIFMetaIdentifier“,
3. „Constraints“,
4. „Description“,
5. „Name“ und
6. „Usage“.

3.1.4.1.1 Meta-Meta-Attribut „Aliases“

Das optionale Meta-Meta-Attribut „Aliases“ [213] ist vom EIA/CDIF-Datentyp „String“ und ist mit 1024 Zeichen begrenzt. Sofern Aliase, also Synonyme, exi-

213 Vgl. [CDIF94b], Seite 50.

stieren, müssen diese durch eine mit Kommata separierte Liste innerhalb der Zeichenkette angegeben werden. Jedes Synonym muß überdies einen Wert entsprechend dem EIA/CDIF-Datentyp „Identifier“ aufweisen.

Tabelle 3-5 beschreibt das Meta-Meta-Attribut „Aliases“.

Name des Attributs	**Bedeutung**
Name	***Aliases***
Description	*Gibt alternative Namen (Synonyme) an.*
Usage	*Damit können Namen, die semantisch äquivalent sind, erkannt werden. In EIA/CDIF-Transfers wird allerdings nur das Meta-Meta-Attribut „Name“ verwendet.*
Aliases	*Synonyms, AlternateNames, AKAs*[214]
Constraints	*Der Wert muß eine kommaseparierte Liste von Namen enthalten, die jeweils vom EIA/CDIF-Datentyp „Identifier“ sind.*
DataType	*String*
Domain	–
Length	*1024*
IsOptional	*True*

Tabelle 3-5: Meta-Meta-Attribut „Aliases“

3.1.4.1.2 Meta-Meta-Attribut „CDIFMetaIdentifier“

Das zwingend vorgeschriebene Meta-Meta-Attribut „CDIFMetaIdentifier“[215] ist vom EIA/CDIF-Datentyp „Identifier“ und identifiziert ein Meta-Objekt eindeutig. Damit wird durch dieses Meta-Meta-Attribut in einem EIA/CDIF-Transfer auf Meta-Modellebene (M2-Schicht) die Entitätsintegritätsbedingung gewährleistet. Die Identifier-Werte beginnen mit einer Ziffer, wenn die Instanzen[216] von

214 Das weit verbreitete Akronym „AKA“ leitet sich aus dem englischen „**A**lso **K**nown **A**s“ (deutsch: „auch bekannt unter“) ab.

215 Vgl. [CDIF94b], Seite 50.

216 Konkret handelt es sich um Instanzen der Meta-Meta-Entitäten „SubjectArea“, „Meta-Attribute“, „AttributableMetaObject“, „MetaEntity“ und „MetaRelationship“, die Subklassen von „MetaObject“ sind. Wie bereits weiter oben ausgeführt, sind alle anderen Meta-Meta-Entitäten abstrakt.

„MetaObject“ von EIA/CDIF im Rahmen der Definition von standardisierten EIA/CDIF-Meta-Modellen erfolgt ist, mit einem Buchstaben sonst[217].

Tabelle 3-6 beschreibt das Meta-Meta-Attribut „CDIFMetaIdentifier“.

Name des Attributs	Bedeutung
Name	***CDIFMetaIdentifier***
Description	*Enthält eindeutigen Wert.*
Usage	*CDIFMetaIdentifier werden für Meta-Modelle (M2-Schicht) in EIA/CDIF-Transfers benötigt, um Instanzen von „MetaObject“ beziehungsweise dessen Subtypen eindeutig identifizieren zu können.[218]*
Aliases	_ [219]
Constraints	*Die Werte in von EIA/CDIF-standardisierten Meta-Modellen beginnen mit einer Ziffer, mit einem Buchstaben sonst.*
DataType	*Identifier*
Domain	–
Length	–
IsOptional	*False*

Tabelle 3-6: Meta-Meta-Attribut „CDIFMetaIdentifier“

3.1.4.1.3 Meta-Meta-Attribut „Constraints“

Das Meta-Meta-Attribut „Constraints“ [220] ist vom EIA/CDIF-Datentyp „Text“ und beinhaltet eine *Beschreibung* von Einschränkungen und Nebenbedingungen *in*

217 Beispielsweise müssen Erweiterungen zu standardisierten Meta-Modellen beziehungsweise Definitionen von proprietären Meta-Modellen im Rahmen eines EIA/CDIF-Austausches Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“ aufweisen, die mit einem Buchstaben beginnen.

218 Somit existieren in CDIF zwei verschiedene Konzepte, um auf Meta-Modellebene (M2-Schicht) Meta-Entitäten voneinander zu unterscheiden: einerseits das für diesen Zweck definierte Meta-Meta-Attribut „CDIFMetaIdentifier“ und andererseits die vollqualifizierten Namen für Meta-Entitäten, Meta-Beziehungen und Meta-Attribute. Beide Identifikationskonzepte sind voneinander unabhängig.

219 Als Alias böte sich hier der Begriff „Surrogat“ an, vgl. z.B. [Codd79].

220 Vgl. [CDIF94b], Seite 51.

natürlicher Sprache. Beispielsweise sollen einander ausschließende[221] Beziehungen in dieser Form dokumentiert werden.

Tabelle 3-7 beschreibt das Meta-Meta-Attribut „Constraints“.

Name des Attributs	Bedeutung
Name	***Constraints***
Description	*Beschreibt Einschränkungen, die in bezug auf Instanzen von „MetaObject“ existieren.*
Usage	*Einschränkungen sollen unmißverständlich in natürlicher Sprache angegeben werden.*
Aliases	*Restriction, Rule, Obligation, Policy*
Constraints	–
DataType	*Text*
Domain	–
Length	–
IsOptional	*True*

Tabelle 3-7: Meta-Meta-Attribut „Constraints“

3.1.4.1.4 Meta-Meta-Attribut „Description“

Das zwingend vorgeschriebene Meta-Meta-Attribut „Description“ [222] ist vom EIA/CDIF-Datentyp „Text“ und beinhaltet eine Beschreibung des Zwecks des entsprechenden Meta-Objekts in natürlicher Sprache. Dieses Meta-Meta-Attribut soll im Unterschied zum Meta-Meta-Attribut „Constraints“ nicht dazu verwendet werden, Einschränkungen, Abhängigkeiten oder Nebenbedingungen zu formulieren.

Tabelle 3-8 beschreibt das Meta-Meta-Attribut „Description“.

---

221 Einander ausschließende Beziehungen (englisch: „mutual exclusive relationships“) werden in den Texten und Erläuterungen zum EIA/CDIF-Standard ausdrücklich unterstützt. Siehe auch die Diskussion über das EIA/CDIF-Meta-Meta-Modell weiter unten, in der darauf eingegangen wird.

222 Vgl. [CDIF94b], Seite 51.

Name des Attributs	Bedeutung
Name	***Description***
Description	*Beschreibt und charakterisiert ein Meta-Objekt.*
Usage	*Dieses Meta-Meta-Attribut soll **nicht** für die Definition von Einschränkungen und Abhängigkeiten benutzt werden, sondern lediglich das Wesen, den Zweck beschreiben.*
Aliases	*Definition*
Constraints	–
DataType	*Text*
Domain	–
Length	–
IsOptional	*False*

Tabelle 3-8: Meta-Meta-Attribut „Description“

3.1.4.1.5 Meta-Meta-Attribut „Name“

Das zwingend vorgeschriebene Meta-Meta-Attribut „Name“ [223] ist vom EIA/CDIF-Datentyp „Identifier“ und beinhaltet den Namen, mit dem ein Meta-Objekt bezeichnet wird.

Tabelle 3-9 beschreibt das Meta-Meta-Attribut „Name“.

Name des Attributs	Bedeutung
Name	***Name***
Description	*Der Name, mit dem ein Meta-Objekt bezeichnet und in weiterer Folge referenziert wird.*
Usage	*Namen sollen prägnant sein, aber nicht auf Kosten der Genauigkeit beziehungsweise Klarheit.*
Aliases	*Title, Identifier*
Constraints	*Der erste Buchstabe aller Namenskomponenten soll in Großbuchstaben geschrieben werden, die restlichen Buchstaben in Kleinbuchstaben, es sei denn, es handelt sich*

[223] Vgl. [CDIF94b], Seite 52.

Name des Attributs	Bedeutung
	um Wörter, die groß geschrieben werden, wie beispielsweise Akronyme.
DataType	*Identifier*
Domain	–
Length	–
IsOptional	*False*

Tabelle 3-9: Meta-Meta-Attribut „Name“

3.1.4.1.6 Meta-Meta-Attribut „Usage“

Das Meta-Meta-Attribut „Usage“[224] ist vom EIA/CDIF-Datentyp „Text“ und beinhaltet Erläuterungen zur Handhabung von Instanzen vom Typ „MetaObject“ beziehungsweise dessen Subtpyen in bezug auf EIA/CDIF-Transfers in natürlicher Sprache. Beispielsweise können hier Hinweise für Informationssystementwickler gegeben werden, die für den Export oder den Import von EIA/CDIF-Transfers relevant sind.

Tabelle 3-10 beschreibt das Meta-Meta-Attribut „Usage“.

Name des Attributs	Bedeutung
Name	***Usage***
Description	*Erläuterungen zur Handhabung des entsprechenden Meta-Objekts.*
Usage	*Kann dazu benutzt werden, um den Zweck eines Meta-Objekts näher zu beschreiben.*
Aliases	*Examples, Hints, Explanation*
Constraints	–
DataType	*Text*
Domain	–

224 Vgl. [CDIF94b], Seite 52.

Name des Attributs	Bedeutung
Length	–
IsOptional	*True*

Tabelle 3-10: Meta-Meta-Attribut „Usage“

3.1.4.2 Meta-Meta-Entität „SubjectArea“

Mit Hilfe der Meta-Meta-Entität „SubjectArea“[225] (deutsch: „Gegenstands-Bereich“) wird ein Gegenstandsbereich definiert, für dessen Konzepte ein Meta-Modell erzeugt wird. Die Meta-Meta-Beziehung „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ legt fest, welche Meta-Objekte in welchem Gegenstandsbereich benutzt werden.

Hierbei wird von der Vorstellung ausgegangen, daß sämtliche definierten Meta-Entitäten, Meta-Beziehungen und Meta-Attribute das sogenannte *Integrierte EIA/CDIF-Meta-Modell* bilden. Eine Instanz der Meta-Meta-Entität „SubjectArea“ definiert insoferne über die Meta-Meta-Beziehung „0:N **CollectableMeta-Object**.*IsUsedIn*.SubjectArea 1:N“ *eine Sicht* beziehungsweise *einen Ausschnitt aus dem integrierten Meta-Modell* und bildet gleichzeitig damit das Meta-Modell des entsprechenden GegenstandsBereiches ab.

Eine „SubjectArea“ beziehungsweise ein „GegenstandsBereich“ legt einerseits die grundlegenden Strukturen für den Austausch von Modelldaten fest, wie beispielsweise in der „Foundation Subject Area“[226] und definiert andererseits in Form von Meta-Modellen die Konzepte von unterschiedlichen Modellierungsmethodologien, deren Modelldaten getauscht werden sollen.[227]

---

225 Vgl. [CDIF94b], Seite 57ff.

226 Vgl. [CDIF94f].

227 Beispielsweise werden in [CDIF96c] sämtliche für die Modellierung von Datenflüssen relevanten Modelldaten und Modellbeziehungen definiert, soweit dies der entsprechenden EIA/CDIF-Arbeitsgruppe als notwendig und sinnvoll erschien. Bei EIA/CDIF-Meta-Modellen wird grundsätzlich versucht, sämtliche Varianten einer Modellierungsmethodologie zu berücksichtigen, soweit dies möglich ist. Dies hat zur Folge, daß teilweise langwierige Abstimmungsprozesse notwendig sind, um zu mehrheitsfähigen Beschlüssen beziehungsweise zu einem Konsens darüber zu kommen, welche Konzepte beziehungsweise diese repräsentierenden Entitäts- und Beziehungstypen für bestimmte Methodologien aufgenommen werden sollen. Auf der anderen Seite wird dadurch der Versuch unternommen, nach Möglichkeit die wichtigsten Varianten bestimmter Modellie-

Fortsetzung folgt auf der nächsten Seite.

Name des Attributs	Bedeutung
Name	***SubjectArea***
SubtypeOf	*MetaObject*
SupertypeOf	–
Description	*Definiert eine Sicht auf das „Integrierte EIA/CDIF-Meta-Modell“. Die zu einer bestimmten Sicht gehörenden Instanzen von „CollectableMetaObject“ werden über Instanzen der Meta-Meta-Beziehung „IsUsedIn“ der entsprechenden „SubjectArea“ zugeordnet.*
Usage	*Eine Instanz dieser Meta-Meta-Entität wird für jedes EIA/CDIF-Meta-Modell erzeugt. Die in einer „SubjectArea“ zusammengefaßten Definitionen können im Zuge eines EIA/CDIF-Transfers erweitert werden.*
Aliases	*View*
Constraints	–
Type	*Kernel*

Tabelle 3-11: Meta-Meta-Entität „SubjectArea“

Die Meta-Meta-Entität „SubjectArea“ verfügt über die folgenden ererbten Meta-Meta-Attribute:

1. „Aliases“ (von „MetaObject“),
2. „CDIFMetaIdentifier“ (von „MetaObject“),
3. „Constraints“ (von „MetaObject“),
4. „Description“ (von „MetaObject“),
5. „Name“ (von „MetaObject“) und
6. „Usage“ (von „MetaObject“).

Für die Meta-Meta-Entität „SubjectArea“ wird darüber hinaus noch das Meta-Meta-Attribut „VersionNumber“ definiert.

„SubjectArea“ nimmt direkt an der folgenden Meta-Meta-Beziehung teil:

---

rungsmethoden zu berücksichtigen (vgl. beispielsweise auch die Meta-Modelldefinitionen für den GegenstandsBereich „Datenmodellierung“ in [CDIF96b]).

1. „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“.[228]

3.1.4.2.1 Meta-Meta-Attribut „VersionNumber“

Das zwingend vorgeschriebene Meta-Meta-Attribut „VersionNumber“[229] ist vom EIA/CDIF-Datentyp „String“ und ist mit 8 Zeichen begrenzt. Es enthält die Versionsnummer des für einen bestimmten GegenstandsBereich festgelegten Meta-Modells.

Tabelle 3-12 beschreibt das Meta-Meta-Attribut „VersionNumber“.

Name des Attributs	Bedeutung
Name	***VersionNumber***
Description	*Versionsnummer der „SubjectArea“.*
Usage	*EIA/CDIF-Standardisierte GegenstandsBereiche führen die Versionsnummer im Format „nn.nn“ an, wobei „n“ für eine beliebige Zahl steht.*
Aliases	–
Constraints	–
DataType	*String*
Domain	–
Length	*8*
IsOptional	*False*

Tabelle 3-12: Meta-Meta-Attribut „VersionNumber“

228 Die Darstellung des vollqualifizierten Namens der Meta-Meta-Beziehungstypen erfolgt im Unterschied zum EIA/CDIF-Standard gemeinsam mit den entsprechenden Kardinalitäten. Die Namen der Meta-Meta-Beziehungen werden darin kursiv dargestellt. Die Namen jener Meta-Meta-Entitätstypen, deren Instanzen aufgrund der gegebenen Kardinalitäten an Instanzen des Meta-Meta-Beziehungstypes enthalten sein müssen, werden fett dargestellt. (Hierbei müssen Instanzen von Meta-Meta-Entitätstypen dann zwingenderweise in Instanzen eines Meta-Meta-Beziehungstyps enthalten sein, wenn die gegenüberliegende Kardinalität im Minimum einen Wert größer Null aufweist.)

229 Vgl. [CDIF94b], Seite 58.

3.1.4.3 Meta-Meta-Entität „CollectableMetaObject“

Die abstrakte Meta-Meta-Entität „CollectableMetaObject“[230] (deutsch: „SammelbaresMetaObjekt“) stellt die Wurzel für jene konkreten Meta-Meta-Entitäten dar, die im integrierten Meta-Modell gesammelt werden können. Es sind dies die Instanzen der (konkreten) Meta-Meta-Entitäten „MetaAttribute“, „AttributableMetaObject“, „MetaRelationship“ und „MetaEntity“.

Jede Instanz vom Typ „CollectableMetaObject“ kann – entsprechend der Meta-Meta-Beziehung „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ – in beliebig vielen (unterschiedlichen) „SubjectAreas“ referenziert[231] werden. Die mengenmäßige Vereinigung sämtlicher SammelbarerMetaObjekte über alle standardisierten EIA/CDIF-Meta-Modelle hinweg, also aller GegenstandsBereiche, bildet das sogenannte „Integrierte EIA/CDIF-Meta-Modell“[232].

Name des Attributs	**Bedeutung**
Name	***CollectableMetaObject***
SubtypeOf	*MetaObject*
SupertypeOf	*AttributableMetaObject* *MetaAttribute*
Description	*Definiert den Supertyp für jene MetaObjekte, die in GegenstandsBereichen (SubjectAreas) gesammelt werden können.*
Usage	*Diese Meta-Meta-Entität ist abstrakt und soll daher nicht instanziiert werden.*
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 3-13: Meta-Meta-Entität „CollectableMetaObject“

230 Vgl. [CDIF94b], Seite 43.

231 Eine Referenzierung setzt natürlich voraus, daß (vollständig) bekannt ist, welche Instanzen vom Typ „CollectableMetaObject“ bereits im Integrierten EIA/CDIF-Meta-Modell existieren, unabhängig davon, in welchem GegenstandsBereich diese ursprünglich definiert wurden.

232 Dieses hypothetische „Integrierte EIA/CDIF-Meta-Modell“ wird in dieser Arbeit unter Berücksichtigung aller bis Jänner 1998 offiziell von EIA/CDIF verabschiedeten Meta-Modelle erstmals physisch erarbeitet und vorgestellt.

Die Meta-Meta-Entität „CollectableMetaObject“ verfügt über die folgenden vererbten Meta-Meta-Attribute:

1. „Aliases“ (von „MetaObject“),
2. „CDIFMetaIdentifier“ (von „MetaObject“),
3. „Constraints“ (von „MetaObject“),
4. „Description“ (von „MetaObject“),
5. „Name“ (von „MetaObject“) und
6. „Usage“ (von „MetaObject“).

Darüber hinaus wird für diese Meta-Meta-Entität kein weiteres Meta-Meta-Attribut definiert.

„CollectableMetaObject“ nimmt direkt an der folgenden Meta-Meta-Beziehung teil:

1. „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“.

3.1.4.4 Meta-Meta-Entität „AttributableMetaObject“

Die Meta-Meta-Entität „AttributableMetaObject“[233] (deutsch: „Attribuierbares-MetaObjekt“) stellt die Wurzel für die attribuierbaren Meta-Meta-Entitäten „MetaEntity“ und „MetaRelationship“ dar. Direkte Instanzen dieses Typs werden graphisch als Rechtecke dargestellt.[234].

Mit Hilfe der rekursiven Meta-Meta-Beziehung „0:N AttributableMetaObject.*Has-Subtype*.AttributableMetaObject 0:N“ ist es möglich, Super-/Subtyp-

233 Vgl. [CDIF94b], Seite 41.

234 Im gesamten „Integrierten EIA/CDIF-Meta-Modell“ gibt es nur eine einzige direkte Instanz vom Typ „AttributableMetaObject“, und zwar mit der Bezeichnung „RootObject“.
Diese einzigartige Stellung von „RootObject“, als Supertyp für „RootEntity“ und „RootEntity.*IsRelatedTo*.RootEntity“, wird auch dadurch unterstrichen, daß das Surrogat in Form des Meta-Meta-Attributs „CDIFMetaIdentifier“ den Wert „1“ besitzt. Diese drei (teilweise indirekten) grundlegenden Instanzen von „AttributableMetaObject“ werden im Standard für das fundamentale EIA/CDIF-Meta-Modell (vgl. [CDIF94f]) definiert, wobei „RootObject“ eine Instanz von „AttributableMetaObject“, „RootEntity“ (Wert „2“ im Attribut von „CDIFMetaIdentifier“) eine Instanz von „Meta-Entity“ und „RootEntity.*IsRelatedTo*.-RootEntity“ (Wert „3“ im Attribut von „CDIFMetaIdentifier“) eine Instanz von „MetaRelationship“ ist. Sämtliche Meta-Entitäten und Meta-Beziehungen eines EIA/CDIF-Transfers müssen „RootObject“ als ihre Wurzel aufweisen.

hierarchien[235] für Instanzen von „AttributableMetaObject“ (also von „MetaEntity“ beziehungsweise „MetaRelationship“) anzugeben. Zusätzlich wird durch die „0:N“ zu „0:N“ Kardinalität in der Meta-Meta-Beziehung zum Ausdruck gebracht, daß auch Mehrfachvererbungen zulässig sind.[236]

Die Meta-Meta-Beziehung „0:N **MetaAttribute**.*IsLocalMetaAttributeOf.*AttributableMetaObject 1:1“ erlaubt die Zuordnung von beliebig vielen Attributdefinitionen (Instanzen von „MetaAttribute“) zu Instanzen von „AttributableMetaObject“, beziehungsweise auch zu dessen Subtypen „MetaEntity“ und „MetaRelationship“. Aus der Kardinalität ist ableitbar, daß Instanzen von „AttributableMetaObject“ nicht über zugeordnete Meta-Attribute verfügen müssen.

Name des Attributs	**Bedeutung**
Name	***AttributableMetaObject***
SubtypeOf	*CollectableMetaObject*
SupertypeOf	*MetaEntity* *MetaRelationship*
Description	*Definiert einen Supertyp für jene Meta-Meta-Entitäten (also „MetaEntity“ und „MetaRelationship“), die Meta-Attribute aufweisen dürfen.*
Usage	*Diese Meta-Meta-Entität erlaubt es Meta-Entitäten und Meta-Beziehungen, Meta-Attribute aufzuweisen sowie die Bildung von Supertyp/Subtyp-Hierarchien.*
Aliases	–

235 Werte für die Meta-Meta-Attribute „SubtypeOf“ und „SupertypeOf“ können aus der Meta-Meta-Beziehung „HasSubtype“ abgeleitet werden.

236 In der Diskussion der Mehrfachvererbung (vgl. [CDIF94a], Seiten 9 bis 10 und Seite 12) werden folgende drei Klarstellungen getroffen:

1) es gibt *keine* vordefinierte Reihenfolge, in der die direkten Supertypen aufzusuchen sind,
2) wenn ein Supertyp *X* der direkte beziehungsweise indirekte Supertyp von mehreren, unterschiedlichen in Mehrfachvererbung benutzten Supertypen ist, dann wird der Supertyp *X* selbst nur einmal in den Pfad aller Supertypen aufgenommen, und
3) für den Fall, daß über Mehrfachvererbung in unterschiedlichen, übergeordneten Supertypen unterschiedliche Meta-Attribute mit demselben Namen auftreten, liegt ein (vom Standard nichtlösbarer) Konflikt vor, der entsprechend dem EIA/CDIF-Standard vermieden werden soll. Bisher (Stand: Jänner 1998) wurde in den EIA/CDIF-Meta-Meta-Modellen, in denen die Mehrfachvererbung eingesetzt wird (vgl. beispielsweise die standardisierten Meta-Modelle in [CDIF96b] und [CDIF96c]), Bedacht auf diese Einschränkung genommen.

Name des Attributs	Bedeutung
Constraints	–
Type	*Kernel*

Tabelle 3-14: Meta-Meta-Entität „AttributableMetaObject“

Die Meta-Meta-Entität „AttributableMetaObject“ verfügt über die folgenden ererbten Meta-Meta-Attribute:

1. „Aliases“ (von „MetaObject“),
2. „CDIFMetaIdentifier“ (von „MetaObject“),
3. „Constraints“ (von „MetaObject“),
4. „Description“ (von „MetaObject“),
5. „Name“ (von „MetaObject“) und
6. „Usage“ (von „MetaObject“).

Darüber hinaus wird für diese Meta-Meta-Entität kein weiteres Meta-Meta-Attribut vorgesehen.[237]

Als (direkter) Subtyp von „CollectableMetaObject“ tritt „AttributableMetaObject“ über die Vererbungsbeziehung in der folgenden Meta-Meta-Beziehung auf:

1. „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“
   (definiert für den Supertyp „CollectableMetaObject“).

„AttributableMetaObject“ nimmt darüber hinaus direkt[238] an den folgenden Meta-Meta-Beziehungen teil:

1. „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“ und
2. „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMetaObject 1:1“.

---

237 Für den ISO/CDIF-Standard (vgl. [ISOCDIF98b]) wird dieser Meta-Meta-Entität ein Meta-Meta-Attribut „IsAbstract“ hinzugefügt, mit dessen Hilfe Instanzen vom Typ „AttributableMetaObject“ ausdrücklich als abstrakt definiert werden können.

238 Im EIA/CDIF-Standard wird dies als „lokal“ bezeichnet.

3.1.4.5 Meta-Meta-Entität „MetaAttribute“

Mit Hilfe der Meta-Meta-Entität „MetaAttribute“[239] (deutsch: „MetaAttribut“) werden Attribute definiert, die anschließend einem AttribuierbarenMetaObjekt über die Meta-Meta-Beziehung „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.-AttributableMetaObject 1:1“ zugeordnet werden.

Name des Attributs	Bedeutung
Name	***MetaAttribute***[240]
SubtypeOf	*CollectableMetaObject*
SupertypeOf	–
Description	*Definiert ein Meta-Attribut für Instanzen vom Typ „AttributableMetaObject“ und damit auch für dessen Subtypen „MetaEntity“ und „MetaRelationship“.*
Usage	–
Aliases	*Property, Characteristic*
Constraints	*Der Name eines MetaAttributs für ein AttribuierbaresMeta-Objekt darf nicht in dessen übergeordneten Attribuierbaren-MetaObjekten (für ein ererbtes MetaAttribut) verwendet worden sein.*
Type	*Kernel*

Tabelle 3-15: Meta-Meta-Entität „MetaAttribute“

Die Meta-Meta-Entität „MetaAttribute“ verfügt über die folgenden ererbten Meta-Meta-Attribute:

1. „Aliases“ (von „MetaObject“),
2. „CDIFMetaIdentifier“ (von „MetaObject“),
3. „Constraints“ (von „MetaObject“),
4. „Description“ (von „MetaObject“),

239 Vgl. [CDIF94b], Seite 44ff.

240 Der vollqualifizierte Name für ein MetaAttribut ergibt sich aus dem vollqualifizierten Namen des AttribuierbarenMetaObjekts, dem das MetaAttribut über die Meta-Meta-Beziehung „0:N **MetaAttribute**.*IsLocalMetaAttribute*.AttributableMetaObject 1:1“ zugeordnet ist, einem Punkt und dem Namen des MetaAttributs selbst.

5. „Name“ (von „MetaObject“) und
6. „Usage“ (von „MetaObject“).

Für die Meta-Meta-Entität „MetaAttribute“ werden die folgenden Meta-Meta-Attribute definiert:

1. „DataType“,
2. „Domain“,
3. „IsOptional“ und
4. „Length“.

Als (direkter) Subtyp von „CollectableMetaObject“ tritt „MetaAttribute“ über die Vererbungsbeziehung in der folgenden Meta-Meta-Beziehung auf:

1. „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“
   (definiert für den Supertyp „CollectableMetaObject“).

„MetaAtribute“ nimmt direkt an der folgenden Meta-Meta-Beziehung teil:

1. „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMetaObject 1:1“.

3.1.4.5.1 Meta-Meta-Attribut „DataType“

Das zwingend vorgeschriebene Meta-Meta-Attribut „DataType“[241] ist vom EIA/CDIF-Datentyp „Enumerated“ und gibt in Form von CDIF-Identifiern die zulässigen Werte für EIA/CDIF-Datentypen vor: „BitMap“, „Boolean“, „Date“, „Enumerated“, „Float“, „Identifier“, „Integer“, „IntegerList“, „Point“, „PointList“, „String“, „Text“ und „Time“.

Tabelle 3-16 beschreibt das Meta-Meta-Attribut „DataType“.

Name des Attributs	Bedeutung
Name	***DataType***
Description	*Definiert den zulässigen Datentyp.*
Usage	–
Aliases	*Usage, Type*

241 Vgl. [CDIF94b], Seite 45, und Abschnitt 3.1.2 auf Seite 54ff weiter oben.

Name des Attributs	Bedeutung
Constraints	–
DataType	*Enumerated*
Domain	BitMap, Boolean, Date, Enumerated, Float, Identifier, Integer, IntegerList, Point, PointList, String, Text, Time
Length	–
IsOptional	*False*

Tabelle 3-16: Meta-Meta-Attribut „DataType“

3.1.4.5.2 Meta-Meta-Attribut „Domain“

Das Meta-Meta-Attribut „Domain“[242] ist vom EIA/CDIF-Datentyp „Text“ und definiert die zulässige Wertemenge. Wenn „DataType“ vom EIA/CDIF-Datentyp „Enumerated“ ist, *muß* eine kommaseparierte Liste von Werten vom EIA/CDIF-Datentyp „Identifier“ dafür angegeben sein. In allen anderen Fällen ist ein Wert optional; wird er trotzdem angegeben, so soll es sich um eine kurze, unmißverständliche Beschreibung der zulässigen Wertemenge in englischer Sprache handeln.

Tabelle 3-17 beschreibt das Meta-Meta-Attribut „Domain“.

Name des Attributs	Bedeutung
Name	***Domain***
Description	*Definiert die Menge der zulässigen Werte.*
Usage	*Wenn „DataType“ vom EIA/CDIF-Datentyp „Enumerated“ ist, muß eine kommaseparierte Liste von zulässigen, eindeutigen Werten (vom EIA/CDIF-Datentyp „Identifier“) angegeben werden.* *In allen anderen Fällen kann eine ausdrückliche Festlegung der Wertemenge entfallen, sodaß sämtliche Werte entsprechend dem angegeben Datentyp in „DataType“ zulässig sind. Notwendige Einschränkungen werden in englischer Sprache prägnant und unmißverständlich*

242 Vgl. ebenda.

Name des Attributs	Bedeutung
	angegeben.
Aliases	–
Constraints	*Die Wertemenge muß angegeben werden, wenn „DataType“ vom EIA/CDIF-Datentyp „Enumerated“ ist.*
DataType	*Text*
Domain	–
Length	–
IsOptional	*True*

Tabelle 3-17: Meta-Meta-Attribut „Domain“

3.1.4.5.3 Meta-Meta-Attribut „IsOptional“

Das Meta-Meta-Attribut „IsOptional“ [243] ist vom EIA/CDIF-Datentyp „Boolean“ und gibt an, ob Werte angegeben werden müssen oder nicht.

Tabelle 3-18 beschreibt das Meta-Meta-Attribut „IsOptional“.

Name des Attributs	Bedeutung
Name	***IsOptional***
Description	*Gibt an, ob ein Wert angegeben werden muß.*
Usage	–
Aliases	*NotRequired, NotMandatory, Nullable*
Constraints	–
DataType	*Boolean*
Domain	–
Length	–
IsOptional	*True*

Tabelle 3-18: Meta-Meta-Attribut „IsOptional“

[243] Vgl. [CDIF94b], Seite 46.

3.1.4.5.4 Meta-Meta-Attribut „Length“

Das Meta-Meta-Attribut „Length“[244] ist vom EIA/CDIF-Datentyp „Integer“ und gibt die maximale Anzahl an Zeichen für EIA/CDIF-Daten vom Typ „String“ an.

Tabelle 3-19 beschreibt das Meta-Meta-Attribut „Length“.

Name des Attributs	Bedeutung
Name	***Length***
Description	*Legt fest, aus wievielen Zeichen maximal („Länge“) ein CDIF-“String“ bestehen darf.*
Usage	*Wird nur für den EIA/CDIF-Datentyp „String“ benutzt.*
Aliases	*Size*
Constraints	*Darf nur für den EIA/CDIF-Datentyp „String“ angegeben werden.*
DataType	*Integer*
Domain	1 <= n <= 1024
Length	–
IsOptional	*True*

Tabelle 3-19: Meta-Meta-Attribut „Length“

3.1.4.6 Meta-Meta-Entität „MetaEntity“

Die Meta-Meta-Entität „MetaEntity“[245] (deutsch: „MetaEntität“) erlaubt die Beschreibung von Meta-Entitäten in Meta-Modellen. Instanzen dieses Typs werden graphisch als Rechtecke dargestellt. Die Meta-Attribute, die derartige Instanzen von „MetaEntity“ aufweisen sollen, werden mit Hilfe von Instanzen von „MetaAttribute“ und der Meta-Meta-Beziehung „0:N **MetaAttribute**.*IsLocalMeta-AttributeOf.*AttributableMetaObject 1:1“ festgelegt, die „MetaEntity“ als Subtyp von „AttributableMetaObject“ erbt.

244 Vgl. [CDIF94b], Seite 46.
245 Vgl. [CDIF94b], Seite 47ff.

Name des Attributs	Bedeutung
Name	***MetaEntity***
SubtypeOf	*AttributableMetaObject*
SupertypeOf	–
Description	*Diese Meta-Meta-Entität erlaubt die Definition von Meta-Entitäten in Meta-Modellen.*
Usage	–
Aliases	–
Constraints	*Alle Namen von „MetaEntity"-Instanzen müssen über alle Meta-Modelle*[246] *hinweg eindeutig sein. Zudem muß jede Meta-Entität (in)direkt von „RootEntity" und damit letztlich von „RootObject"*[247] *abgeleitet werden.*
Type	*Kernel*

Tabelle 3-20: Meta-Meta-Entität „MetaEntity"

Die Meta-Meta-Entität „MetaEntity" verfügt über die folgenden vererbten Meta-Meta-Attribute:

1. „Aliases" (von „MetaObject"),
2. „CDIFMetaIdentifier" (von „MetaObject"),
3. „Constraints" (von „MetaObject"),
4. „Description" (von „MetaObject"),
5. „Name" (von „MetaObject") und
6. „Usage" (von „MetaObject").

246 Genauer: die Namen von Meta-Entitäten, sowie die vollqualifizierten Namen von Meta-Beziehungen (Instanzen der Meta-Meta-Entität „MetaRelationship") müssen im *„working meta-model"* (deutsch: Arbeits-Meta-Modell) eindeutig sein. Das „Arbeits-Meta-Modell" wird durch Definitionen von Meta-Entitäten, Meta-Beziehungen und entsprechenden Meta-Attributen, beziehungsweise durch die Referenzierung auf bereits definierte Meta-Modelle im Rahmen eines EIA/CDIF-Transfers gebildet.

247 Die Meta-Entität „RootObject" wird im EIA/CDIF-Standard, „Foundation Subject Area" (vgl. [CDIF94f]) als Wurzel für sämtliche EIA/CDIF-konformen Meta-Modelle definiert. Um EIA/CDIF-entsprechend (englisch: „compliant") zu sein, muß in einem EIA/CDIF-Transfer zumindest dieses einfache, aber grundlegende Meta-Modell referenziert werden.

Für die Meta-Meta-Entität „MetaEntity“ wird folgendes Meta-Meta-Attribut definiert:

1. „Type“.

Instanzen von „MetaEntity“ können aufgrund der im Meta-Meta-Modell dargestellten Vererbungsbeziehungen in Instanzen folgender Meta-Meta-Beziehungen enthalten sein:

1. „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“
   (definiert für den Supertyp „CollectableMetaObject“),
2. „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMetaObject 1:1“
   (definiert für den Supertyp „AttributableMetaObject“) und
3. „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“
   (definiert für den Supertyp „AttributableMetaObject“).

Instanzen von „MetaEntity“ *können* aufgrund der geltenden Kardinalitäten direkt an den folgenden Meta-Meta-Beziehungen teilnehmen:

1. „0:N **MetaRelationship**.*HasSource*.MetaEntity 1:1“ und
2. „0:N **MetaRelationship**.*HasDestination*.MetaEntity 1:1“.

3.1.4.6.1 Meta-Meta-Attribut „Type“

Das Meta-Meta-Attribut „Type“[248] ist vom EIA/CDIF-Datentyp „Enumerated“ und gibt in Form von CDIF-Identifiern die zulässigen Werte für EIA/CDIF-MetaEntity-Typen vor: „Kernel“, „Characteristic“ und „Associative“.[249]

---

248 Vgl. [CDIF94b], Seite 48.

249 Diese Klassifikation von Entitätstypen findet sich bereits in [Codd79] und später etwa in [Date85]. Vgl. in diesem Zusammenhang auch die Ausführungen in [Flat90b], Seiten 69 bis 92. Entitätstypen, die benutzerdefinierte Wertemengen beinhalten, werden in EIA/CDIF nicht unterschieden (vgl. hierzu auch die Diskussion in [Flat90b], Seite 77ff, über „designative“ versus „beschreibende“ Entitäten).
Die drei Klassifikationsmerkmale für Meta-Entitäten lassen sich demgemäß wie folgt beschreiben:
a) „Kernel“ – Meta-Entitäten dieses Typs verfügen über keine Fremdschlüssel im Primärschlüssel beziehungsweise in einem der Primärschlüsselkandidaten (vgl. [Flat90b], Seite 75),
b) „Characteristic“ – Meta-Entitäten dieses Typs verfügen über einen Fremdschlüssel im Primärschlüssel beziehungsweise in einem der Primärschlüsselkandidaten, der genau *eine* weitere Meta-Entität referenziert (vgl. [Flat90b], Seite 75) und diese daher näher bestimmt, sowie

Fortsetzung folgt auf der nächsten Seite.

Tabelle 3-21 beschreibt das Meta-Meta-Attribut „Type“.

Name des Attributs	Bedeutung
Name	***Type***
Description	*Erlaubt die optionale Klassifikation von Meta-Entitäten.*
Usage	*Dient zur Unterstützung semantischer Modellierungsvarianten, die Entitätstypen näher klassifizieren. „Kernel“ bezeichnet Meta-Entitäten, die für sich allein existieren können; „Characteristic“ bezeichnet Meta-Entitäten, die andere Meta-Entitäten näher beschreiben und daher in ihrer Existenz von dieser anderen Meta-Entität abhängig sind; „Associative“ bezeichnet Meta-Entitäten, die zwei oder mehrere Meta-Entitäten miteinander in Beziehung setzen und damit in ihrer Existenz von diesen abhängig sind.*[250]
Aliases	*Classification, RepresentedSemantic*
Constraints	–
DataType	*Enumerated*
Domain	*Kernel, Characteristic, Associative*
Length	–
IsOptional	*True*

Tabelle 3-21: Meta-Meta-Attribut „Type“

c) „Associative“ – Meta-Entitäten dieses Typs verfügen über zwei oder mehr Fremdschlüssel im Primärschlüssel beziehungsweise in einem der Primärschlüsselkandidaten, die zwei oder mehrere Meta-Entitäten referenzieren (vgl. [Flat90b], Seite 76 unten).
Die assoziative Klassifikation könnte in Meta-Modellen in Situationen eingesetzt werden, in denen mehrstellige (nichtbinäre) Meta-Beziehungen auftreten, die aufgrund der im Meta-Meta-Modell festgelegten Modellierungsmethodik nicht abbildbar sind, da sie nur binäre Meta-Beziehungen zuläßt. Stattdessen wird in einem solchen Fall diese Tatsache dadurch camoufliert, daß die Meta-Beziehung als assoziative Meta-Entität modelliert wird (eine Technik, die in manchen CASE-Systemen durchaus üblich ist, vgl. beispielsweise die Entity-Relationship-Variante der Firma Oracle, wie sie bereits in [Bark90b] dokumentiert und seither nicht verändert wurde). Natürlich wird dadurch die Semantik von Entity-Relationship-Modellen verfälscht und die Verständlichkeit von umfangreichen Entity-Relationship-Modellen beeinträchtigt.

250 Siehe auch Erläuterungen in Fußnote 249 auf Seite 79.

3.1.4.7 Meta-Meta-Entität „MetaRelationship“

Die Meta-Meta-Entität „MetaRelationship“[251] (deutsch: „MetaBeziehung“) erlaubt die Beschreibung von *binären*[252] Meta-Beziehungen in Meta-Modellen. Instanzen dieses Typs werden als Pfeile graphisch dargestellt und sind somit *gerichtet*, sodaß jederzeit feststellbar ist, welche der beiden am Beziehungstyp teilhabenden Entitätstypen[253] den Ausgangspunkt („Quelle“) und welche das Ziel darstellt. Die Meta-Attribute, die derartige Instanzen von „MetaRelationship“ aufweisen sollen, werden mit Hilfe von Instanzen von „MetaAttribute“ und der Meta-Meta-Beziehung „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMetaObject 1:1“ festgelegt, die „MetaRelationship“ als Subtyp von „AttributableMetaObject“ erbt.

Beziehungstypen in CDIF werden mit Kardinalitäten versehen, die darüber Auskunft geben, mit wievielen Instanzen eines Entitätstyps eine Beziehung *mindestens* eingegangen wird und mit wievielen Instanzen desselben Entitätstyps *maximal* Beziehungen existieren können.[254]

---

251 Vgl. [CDIF94b], Seite 53ff.

252 Auch wenn dadurch die Meta-Modellbildung nur mit Hilfe von binären Meta-Beziehungstypen erfolgen kann, ist es durch die Definition eines entsprechenden Meta-Modells möglich, Modelldaten mit nicht-binären Beziehungstypen zu tauschen. Vgl. hierzu auch das semantische Meta-Modell zum Austausch von Modelldaten für den GegenstandsBereich „Datenmodellierung“ [CDIF96b].

253 Welche beiden Entitätstypen an einem Beziehungstyp teilhaben, kann man nur aus der graphischen Darstellung oder dem vollqualifizierten Namen erkennen! In der Graphik gibt die Pfeilrichtung an, welcher Entitätstyp die Quelle (Ausgangspunkt des Pfeiles) und welcher das Ziel (Pfeilspitze) darstellt. Beim vollqualifizierten Namen bezeichnet der erste Namensbestandteil (vor dem ersten Punkt) die Quelle und der letzte (nach dem zweiten Punkt) das Ziel.

254 Werden die Kardinalitätsbeschriftungen *vertauscht,* können Sie als „strukturelle Bedingungen“ (englisch: „structural constraints“) im Sinne etwa von [ElmNav89], Seite 55ff, oder auch von [ElmNav89], Seite 30ff, aufgefaßt werden. Dieser einfache Tausch ist deshalb möglich, da EIA/CDIF-Meta-Modelle nur über binäre Beziehungstypen verfügen dürfen und andererseits für die Kardinalitäten die Minimum/Maximum-Schreibweise benutzt wird. In diesem Fall geben sie darüber Auskunft, wie oft mindestens und maximal eine Entität vom entsprechenden Typ in Beziehungen über diesen konkreten Beziehungstyp stehen muß. (Würde von der EIA/CDIF-Kardinalitätsbeschreibung der Maximum-Wert stehen bleiben, erhielten wir die klassische Kardinalitätsnotation in der Tradition von [Chen76]).
Meta-Modelle selbst können aber mehrwertige Beziehungstypen vorsehen, wie dies beispielsweise auch im standardisierten EIA/CDIF-Meta-Modell für den GegenstandsBereich „Datenmodellierung“ (vgl. [CDIF96b]) der Fall ist. Darüber hinaus werden unter anderem sogenannte innere (englisch: „inner“) und äußere (englisch: „outer“) Kardinalitäten in der Minimum/Maximum Schreibweise unterschieden, wie auch „komplexe“ Beziehungstypen erlaubt, die beispielsweise direkt mit weiteren Beziehungstypen assoziiert werden.

Name des Attributs	Bedeutung
Name	***MetaRelationship***
SubtypeOf	*AttributableMetaObject*
SupertypeOf	–
Description	*Diese Meta-Meta-Entität erlaubt die Definition von Meta-Beziehungen in Meta-Modellen.*
Usage	–
Aliases	–
Constraints	*Alle vollqualifizierten[255] Namen von „*MetaRelationship*“ Instanzen müssen über alle Meta-Modelle hinweg eindeutig sein. Zudem muß jede Meta-Beziehung indirekt von der Meta-Beziehung „RootEntity.IsRelatedTo.RootEntity“ und damit auch indirekt von „RootObject“[256] abgeleitet werden.*
Type	*Kernel*

Tabelle 3-22: Meta-Meta-Entität „MetaRelationship“

Die Meta-Meta-Entität „MetaRelationship“ verfügt über die folgenden vererbten Meta-Meta-Attribute:

1. „Aliases“ (von „MetaObject“),
2. „CDIFMetaIdentifier“ (von „MetaObject“),
3. „Constraints“ (von „MetaObject“),
4. „Description“ (von „MetaObject“),
5. „Name“ (von „MetaObject“) und
6. „Usage“ (von „MetaObject“).

255 Den vollqualifizierten Namen einer „MetaBeziehung“ erhält man, indem man unter Beachtung der Pfeilrichtung von der Quell-MetaEntität den Namen durch einen Punkt mit dem Namen der MetaBeziehung und durch einen Punkt mit dem Namen der Ziel-MetaEntität verbindet. Diese einfache Bildungsregel wird dadurch ermöglicht, daß Meta-Modelle nur binäre Meta-Beziehungstypen aufweisen können, die somit jeweils genau zwei Meta-Entitätstypen zueinander in Beziehung setzen.

256 Die Meta-Entität „RootObject“ wird im EIA/CDIF-Standard, „Foundation Subject Area“ (vgl. [CDIF94f]) als Wurzel für sämtliche EIA/CDIF-konformen Meta-Modelle definiert. Um EIA/CDIF-entsprechend (englisch: „compliant“) zu sein, muß in einem EIA/CDIF-Transfer zumindest dieses einfache, aber grundlegende Meta-Modell referenziert werden.

Für die Meta-Meta-Entität „MetaRelationship“ werden folgende Meta-Meta-Attribute definiert:[257]

1. „MinSourceCard“,
2. „MaxSourceCard“,
3. „MinDestCard“ und
4. „MaxDestCard“.

Instanzen von „MetaRelationship“ können aufgrund der im Meta-Meta-Modell dargestellten Vererbungsbeziehungen in Instanzen folgender Meta-Meta-Beziehungen enthalten sein:

1. „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“
   (definiert für den Supertyp „CollectableMetaObject“),
2. „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMetaObject 1:1“
   (definiert für den Supertyp „AttributableMetaObject“) und
3. „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“
   (definiert für den Supertyp „AttributableMetaObject“).

Instanzen von „MetaRelationship“ *müssen* wegen der geltenden Kardinalitäten direkt an den folgenden Meta-Meta-Beziehungen teilnehmen:

1. „0:N **MetaRelationship**.*HasSource*.MetaEntity 1:1“ und
2. „0:N **MetaRelationship**.*HasDestination*.MetaEntity 1:1“.

3.1.4.7.1 Meta-Meta-Attribut „MinSourceCard“

Das zwingend vorgeschriebene Meta-Meta-Attribut „MinSourceCard“[258] ist vom EIA/CDIF-Datentyp „String“ und gibt mit dem Wert „0“ beziehungsweise einer positiven Ganzzahl an, wie oft Quell-Meta-Entitäten *mindestens* von einer Ziel-Meta-Entität aus über Instanzen des gegebenen Meta-Beziehungstyps assoziiert werden. Ist der Wert größer als Null, dann müssen Instanzen der Ziel-Meta-

---

257 Im Unterschied zum EIA/CDIF-Standard wird in der Aufzählung und der daran anschließenden detaillierteren Dokumentation der Meta-Meta-Attribute nicht die alphabetische Reihenfolge gewählt. Stattdessen wird als Ordnungskriterium die Seite (Quelle und Ziel) benutzt, auf der Meta-Entitäten auftreten.

258 Vgl. [CDIF94b], Seite 56.

Entität zwingend an mindestens entsprechend vielen Beziehungen mit Instanzen der Quell-Meta-Entitäten partizipieren.

Tabelle 3-23 beschreibt das Meta-Meta-Attribut „MinSourceCard“.

Name des Attributs	Bedeutung
Name	***MinSourceCard***
Description	*Gibt an, wie oft Quell-Meta-Entitäten mindestens über die Meta-Beziehung von einer Ziel-Meta-Entität aus assoziiert werden.*
Usage	*Ein Wert von „0“ bedeutet optional, nicht-optional sonst.*
Aliases	–
Constraints	*Der Wert muß kleiner oder gleich dem Wert von „MaxSourceCard“ sein. Sofern die Meta-Beziehung in einer Generalisierungsbeziehung mit einer übergeordneten Meta-Beziehung steht, darf der Wert nicht den des Supertypen in „MinSourceCard“ unterschreiten.*
DataType	*String*
Domain	*0 oder positive Ganzzahl bis zum maximalen Wert, der mit einem EIA/CDIF-Datentyp „Integer“ abbildbar ist.*
Length	*10*
IsOptional	*FALSE*

Tabelle 3-23: Meta-Meta-Attribut „MinSourceCard“

3.1.4.7.2 Meta-Meta-Attribut „MaxSourceCard“

Das zwingend vorgeschriebene Meta-Meta-Attribut „MaxSourceCard“[259] ist vom EIA/CDIF-Datentyp „String“ und gibt mit einer positiven Ganzzahl beziehungsweise mit „N“[260] an, wie oft Quell-Meta-Entitäten *maximal* von einer Ziel-Meta-Entität aus über Instanzen des gegebenen Meta-Beziehungstyps assoziiert werden.

Tabelle 3-24 beschreibt das Meta-Meta-Attribut „MaxSourceCard“.

259 Vgl. [CDIF94b], Seite 55.

260 „N“ steht für „uneingeschränkt viele“.

Name des Attributs	Bedeutung
Name	***MaxSourceCard***
Description	*Gibt an, wie oft Quell-Meta-Entitäten maximal über die Meta-Beziehung von einer Ziel-Meta-Entität aus assoziiert werden.*
Usage	–
Aliases	–
Constraints	*Der Wert muß größer oder gleich dem Wert von „MinSourceCard“ sein. Sofern die Meta-Beziehung in einer Generalisierungsbeziehung mit einer übergeordneten Meta-Beziehung steht, darf der Wert nicht den des Supertypen in „MaxSourceCard“ überschreiten.*
DataType	*String*
Domain	*Positive Ganzzahl (mit Eins beginnend) bis zum maximalen Wert, der mit einem EIA/CDIF-Datentyp „Integer“ abbildbar ist oder „N“ für „uneingeschränkt viele“.*
Length	*10*
IsOptional	*FALSE*

Tabelle 3-24: Meta-Meta-Attribut „MaxSourceCard“

3.1.4.7.3 Meta-Meta-Attribut „MinDestCard“

Das zwingend vorgeschriebene Meta-Meta-Attribut „MinDestCard“[261] ist vom EIA/CDIF-Datentyp „String“ und gibt mit dem Wert „0“ beziehungsweise einer positiven Ganzzahl an, wie oft Ziel-Meta-Entitäten *mindestens* von einer Quell-Meta-Entität aus über Instanzen des gegebenen Meta-Beziehungstyps assoziiert werden. Ist der Wert größer als Null, dann müssen Instanzen der Quell-Meta-Entität zwingend an mindestens entsprechend vielen Beziehungen mit Instanzen der Ziel-Meta-Entitäten partizipieren.

Tabelle 3-25 beschreibt das Meta-Meta-Attribut „MinDestCard“.

261 Vgl. [CDIF94b], Seite 55.

Name des Attributs	Bedeutung
Name	***MinDestCard***
Description	*Gibt an, wie oft Ziel-Meta-Entitäten mindestens über die Meta-Beziehung von einer Quell-Meta-Entität aus assoziiert werden.*
Usage	*Ein Wert von „0“ bedeutet optional, nicht-optional sonst.*
Aliases	–
Constraints	*Der Wert muß kleiner oder gleich dem Wert von „MaxDestCard“ sein. Sofern die Meta-Beziehung in einer Generalisierungsbeziehung mit einer übergeordneten Meta-Beziehung steht, darf der Wert nicht den des Supertypen in „MinDestCard“ unterschreiten.*
DataType	*String*
Domain	*0 oder positive Ganzzahl bis zum maximalen Wert, der mit einem EIA/CDIF-Datentyp „Integer“ abbildbar ist.*
Length	*10*
IsOptional	*FALSE*

Tabelle 3-25: Meta-Meta-Attribut „MinDestCard“

3.1.4.7.4 Meta-Meta-Attribut „MaxDestCard“

Das zwingend vorgeschriebene Meta-Meta-Attribut „MaxDestCard“[262] ist vom EIA/CDIF-Datentyp „String“ und gibt mit einer positiven Ganzzahl beziehungsweise mit „N“[263] an, wie oft Ziel-Meta-Entitäten *maximal* von einer Quell-Meta-Entität aus über Instanzen des gegebenen Meta-Beziehungstyps assoziiert werden.

Tabelle 3-26 beschreibt das Meta-Meta-Attribut „MaxDestCard“.

262 Vgl. [CDIF94b], Seite 54.

263 „N“ steht für „uneingeschränkt viele“.

Name des Attributs	Bedeutung
Name	***MaxDestCard***
Description	*Gibt an, wie oft Ziel-Meta-Entitäten maximal über die Meta-Beziehung von einer Quell-Meta-Entität aus assoziiert werden.*
Usage	–
Aliases	–
Constraints	*Der Wert muß größer oder gleich dem Wert von „MinDestCard“ sein. Sofern die Meta-Beziehung in einer Generalisierungsbeziehung mit einer übergeordneten Meta-Beziehung steht, darf der Wert nicht den des Supertypen in „MaxDestCard“ überschreiten.*
DataType	*String*
Domain	*Positive Ganzzahl (mit Eins beginnend) bis zum maximalen Wert, der mit einem EIA/CDIF-Datentyp „Integer“ abbildbar ist oder „N“ für „uneingeschränkt viele“.*
Length	*10*
IsOptional	*FALSE*

Tabelle 3-26: Meta-Meta-Attribut „MaxDestCard“

3.1.5 Meta-Meta-Beziehungen

In diesem Abschnitt werden die Meta-Meta-Beziehungen des Meta-Meta-Modells detailliert vorgestellt. Wie im vorhergehenden Abschnitt über Meta-Meta-Entitäten erfolgt die Besprechung der Meta-Meta-Beziehungen in der Reihenfolge, die sich durch ein Top-Down-Abarbeiten der in Abbildung 3-1 auf Seite 53 zu findenden Beziehungstypen ergibt.

Die Meta-Meta-Beziehungen sind im EIA/CDIF-Standard mit denselben Attributen[264] dokumentiert, wie sie für Meta-Beziehungstypen in der Meta-Meta-Entität „MetaRelationship“ auf Seite 81ff in Form von Meta-Meta-Attributen angegeben werden.

Abweichend von den EIA/CDIF-Standards wird folgende Konvention im Zusammenhang mit vollqualifizierten Namen von Beziehungstypen eingeführt: der Name des Beziehungstyps wird kursiv dargestellt; wenn aufgrund der Kardinalität Instanzen von Entitätstypen[265] zwingend in Instanzen vom entsprechenden Beziehungstyp enthalten sein müssen. Des weiteren werden die Kardinalitäten selbst dem vollqualifizierten Namen beigefügt, und erlauben es damit, die wichtigsten strukturellen Eigenschaften zu erkennen.

Die Meta-Meta-Beziehungstypen „0:N **MetaRelationship**.*HasSource*.MetaEntity 0:N“ und „0:N **MetaRelationship**.*HasDestination*.MetaEntity 1:1“ reflektieren die Tatsache, daß für die Erstellung von EIA/CDIF-Meta-Modellen ausschließlich binäre Beziehungstypen verwendet werden dürfen.[266]

Einander ausschließende Beziehungstypen stehen zwar für die graphische Modellierung von Meta-Modellen zur Verfügung, können aber aufgrund der Definition des EIA/CDIF-Meta-Meta-Modells nicht ausdrücklich dokumentiert werden.

---

264 Auch hier gelten die Ausführungen, die zu den Meta-Meta-Entitäten in Fußnote 206 auf Seite 58 gegeben wurden.

265 Dies ist dann der Fall, wenn der Minimalwert der Kardinalität einen Wert größer als Null aufweist.

266 Wären n-ary-Beziehungstypen erlaubt, müßte dies auch im Meta-Meta-Modell durch eine entsprechende Definition ausgedrückt werden.

Tabelle 3-27 führt die Attribute an, die Meta-Meta-Beziehungen beschreiben.[267]

Name des Attributs	Bedeutung
Name	Name der Meta-Meta-Beziehung[268]
MinSourceCard	Mindestanzahl der Beziehungsinstanzen, in denen eine Instanz des Zielentitätstyps enthalten ist und damit in Verbindung mit Instanzen des Quellentitätstyps steht.
MaxSourceCard	Maximale Anzahl der Beziehungsinstanzen, in denen eine Instanz des Zielentitätstyps enthalten ist und damit in Verbindung mit Instanzen des Quellentitätstyps steht.
MinDestCard	Mindestanzahl der Beziehungsinstanzen, in denen eine Instanz des Quellentitätstyps enthalten ist und damit in Verbindung mit Instanzen des Zielentitätstyps steht.
MaxDestCard	Maximale Anzahl der Beziehungsinstanzen, in denen eine Instanz des Quellentitätstyps enthalten ist und damit in Verbindung mit Instanzen des Zielentitätstyps steht.
Description	Beschreibung des Zwecks
Usage	Verwendungshinweise
Aliases	Synonyme, sofern verfügbar
Constraints	Einschränkungen, Nebenbedingungen

Tabelle 3-27: Attribute für die Beschreibung von Meta-Meta-Beziehungstypen[269]

267 Im Unterschied zur Beschreibung von Meta-Meta-Enitäten, Meta-Entitäten und Meta-Beziehungen werden für die Definition von Meta-Meta-Beziehungen keine Attribute vorgesehen, die eine Über- oder Unterordnung ausdrücken. Dies erklärt sich einfach daraus, daß im Meta-Meta-Modell Meta-Meta-Beziehungstypen nicht durch Spezialisierung beziehungsweise durch Generalisierung gebildet werden, daher sind die Meta-Meta-Attribute „SubtypeOf“ und „SupertypeOf“ nicht notwendig. Nachdem aber das Meta-Meta-Modell selbst den darin definierten Regeln unterworfen ist, stünde diese Möglichkeit grundsätzlich offen.

268 Der vollqualifizierte Name eines Beziehungstyps ergibt sich durch die mit Punkten verbundene Folge der Namen des „Quellentitätstyps“ (Entitätstyp, von dem der den Beziehungstyp repräsentierende Pfeil ausgeht), „Beziehungstyp“ und „Zielentitätstyp“ (Entitätstyp, in den der den Beziehungstyp repräsentierende Pfeil eingeht).

269 Die Attribute werden in derselben Reihenfolge angeführt, wie im Standard (vgl. [CDIF94b]).

3.1.5.1 Meta-Meta-Beziehung „0:N CollectableMetaObject.-*IsUsedIn*.SubjectArea 1:N“

Der vollqualifizierte Name vom Meta-Meta-Beziehungstyp „IsUsedIn“[270] (deutsch: „WirdBenutztIm“) lautet: „0:N **CollectableMetaObject**.*IsUsedIn*.-SubjectArea 1:N“ (deutsch: „0:N **SammelbaresMetaObjekt**.*WirdBenutztIm*.-GegenstandsBereich 1:N“). Dieser Beziehungstyp erlaubt daher die Zuordnung von SammelbarenMetaObjekten aus dem „Integrierten EIA/CDIF-Meta-Modell“ zu GegenstandsBereichen.

EIA/CDIF-Meta-Modelle bilden, wie eingangs diskutiert, jene Konzepte ab, die für einen bestimmten Gegenstandsbereich (zumeist eine Modellierungsmethodologie) vom EIA/CDIF-Komitee für wichtig erachtet wurden. Der Beziehungstyp „IsUsedIn“ ermöglicht einerseits die Zusammenfassung der SammelbarenMetaObjekte eines bestimmten GegenstandsBereiches und andererseits die Wiederverwendung von bereits definierten Konzepten in weiteren GegenstandsBereichen.

Aufgrund der gegebenen Kardinalitäten *muß* (MinDestCard=„1“) jedes SammelbareMetaObjekt an dieser Beziehung teilhaben, das heißt zumindest in einem GegenstandsBereich benutzt werden. Nach oben hin gibt es keine Grenze, somit kann ein SammelbaresMetaObjekt in beliebig vielen (MaxDestCard=„N“) GegenstandsBereichen benutzt und damit wiederverwendet werden.

Ein GegenstandsBereich *kann* (MinSourceCard=„0“) beliebig viele (MaxSourceCard=„N“) SammelbareMetaObjekte beinhalten.

Die folgende Tabelle 3-28 beinhaltet sämtliche Definitionen für den Meta-Meta-Beziehungstyp „IsUsedIn“.

Name des Attributs	**Bedeutung**
Name	***IsUsedIn***
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*1*

[270] Vgl. [CDIF94b], Seite 60.

Name des Attributs	Bedeutung
MaxDestCard	*N*
Description	*Dieser Beziehungstyp erlaubt die Definition von Meta-Modellen, indem die SammelbarenMetaObjekte den entsprechenden GegenstandsBereichen zugeordnet werden.*
Usage	*Für jede Zuordnung eines SammelbarenMetaObjekts zu einem GegenstandsBereich wird eine Instanz dieses Beziehungstyps gebildet.*
Aliases	–
Constraints	*Wenn MetaAttribute einem GegenstandsBereich zugeordnet werden, müssen auch die entsprechenden Attribuierbaren-MetaObjekte im entsprechenden Gegenstandsbereich (über eine Instanz dieses Beziehungstyps) enthalten sein.*[271]

Tabelle 3-28: Meta-Meta-Beziehung „0:N **CollectableMetaObject**.*IsUsedIn*.-SubjectArea 1:N“

3.1.5.2 Meta-Meta-Beziehung „0:N MetaAttribute.*IsLocal-MetaAttributeOf*.AttributableMetaObject 1:1“

Der vollqualifizierte Name vom Meta-Meta-Beziehungstyp „IsLocalAttribute-Of“[272] (deutsch: „IstLokalesAttributVon“) lautet: „0:N **MetaAttribute**.*IsLocal-MetaAttributeOf*.AttributableMetaObject 1:1“ (deutsch: „0:N **MetaAttribut**.-*IstLokalesAttributVon*.AttribuierbarenMetaObjekt 1:1“). Dieser Beziehungstyp erlaubt daher die Zuordnung von MetaAttributen zu AttribuierbarenMetaObjekten und kann auch als Aggregierungsabstraktionsbeziehung im Sinne von [BaCeNa92][273] verstanden werden.

---

271 Aufgrund des Beziehungstyps „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.Attributable-MetaObject 1:1“, der weiter unten definiert wird, müssen MetaAttribute immer einem AttribuierbarenMetaObjekt zugeordnet sein und könnten ohne sie gar nicht existieren. Es wäre theoretisch denkbar, daß diese Einschränkung zuvor im Integrierten EIA/CDIF-Meta-Modell befolgt wird, aber daß MetaAttribute allein und ohne den zugehörigen AttribuierbarenMetaObjekten für einen GegenstandsBereich definiert werden. Aufgrund der angeführten Einschränkung wird dies nicht zugelassen.

272 Vgl. [CDIF94b], Seite 61.

273 Vgl. ebenda, Seite 17.

Aufgrund der gegebenen Kardinalitäten *muß* (MinDestCard=„1“) jedes SammelbareMetaObjekt an dieser Beziehung teilhaben, das heißt zumindest in einem Gegenstandsbereich benutzt werden. Es darf nicht der Fall eintreten, daß ein MetaAttribut öfter als einmal (MaxDestCard=„1“) im Beziehungstyp enthalten ist, also für mehr als ein AttribuierbaresMetaObjekt als Attribut zur Verfügung steht.

Ein AttribuierbaresMetaObjekt *kann* (MinSourceCard=„0“) aus beliebig vielen (MaxSourceCard=„N“) MetaAttributen bestehen.

Im Zuge der Bildung einer Generalisierungshierarchie mit Hilfe des weiter unten definierten Meta-Meta-Beziehungstyps „HasSubtype“ können MetaAttribute auch von übergeordneten AttribuierbarenMetaObjekten ererbt werden.[274]

Die folgende Tabelle 3-29 beinhaltet sämtliche Definitionen für den Meta-Meta-Beziehungstyp „IsLocalAttributeOf“.

Name des Attributs	**Bedeutung**
Name	***IsLocalMetaAttributeOf***
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*1*
MaxDestCard	*1*
Description	*Dieser Beziehungstyp erlaubt die Zuordnung von Meta-Attributen zu AttribuierbarenMetaObjekten und erlaubt damit die Aggregationsabstraktionsbeziehung.* *MetaAttribute, die über diesen Beziehungstyp AttribuierbarenMetaObjekten zugeordnet werden, bezeichnet man auch als „lokale“ MetaAttribute. Im Gegensatz dazu können über den als Generalisierungshierarchie interpretierbaren Meta-Meta-Beziehungstyp „HasSubtype“ MetaAttribute zudem auch von übergeordneten Typen ererbt werden.*
Usage	–

274 Mögliche Probleme des Erbens von MetaAttributen bei der Verwendung von Mehrfachvererbung werden im Abschnitt über „HasSubtype“ weiter unten diskutiert.

Name des Attributs	Bedeutung
Aliases	–
Constraints	–

Tabelle 3-29: Meta-Meta-Beziehung „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.-AttributableMetaObject 1:1“

3.1.5.3 Meta-Meta-Beziehung „0:N AttributableMetaObject.-*HasSubtype*.AttributableMetaObject 0:N“

Der vollqualifizierte Name vom Meta-Meta-Beziehungstyp „HasSubtype“[275] (deutsch: „BesitztUntergeordnetes“) lautet: „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“ (deutsch: „0:N AttribuierbaresMetaObjekt.-*BesitztUntergeordnetes*.AttribuierbaresMetaObjekt 0:N“). Dieser Beziehungstyp erlaubt daher die Bildung einer Generalisierungshierarchie, indem damit die Überordnung und die Unterordnung von AttribuierbarenMetaObjekten ermöglicht wird. Untergeordnete AttribuierbareMetaObjekte werden als Spezialisierungen aufgefaßt, die die Eigenschaften sämtlicher übergeordneter AttribuierbarenMetaObjekte über einen Vererbungsmechanismus ererben.

Von besonderem Interesse sind in diesem Zusammenhang die Strukturvererbungsregeln[276] von EIA/CDIF. Wie bereits bei der Definition von „MetaAttribute“ angeführt, müssen sämtliche Namen von MetaAttributen, sowohl von lokalen als auch von ererbten, für ein AttribuierbaresMetaObjekt eindeutig sein. Bei Mehrfachvererbung gilt dies für MetaAttribute sämtlicher definierter Pfade, wobei gilt, daß jedes MetaObjekt hierbei nur einmal aufgesucht wird. Das EIA/CDIF-Meta-Meta-Modell sieht keine Ordnung vor, in der Supertypen bei Mehrfachvererbung aufgesucht werden.[277]

Aufgrund der gegebenen Kardinalitäten *kann* (MinDestCard=„0“) jedes SammelbareMetaObjekt *ein oder mehrere* (MaxDestCard=„N“) Male an dieser Meta-

275 Vgl. [CDIF94b], Seite 59.

276 Vgl. hierzu auch [KapSch96], die einen systematischen Überblick über und Definitionen für objektorientierte Informationssysteme geben.

277 Damit ist *jede* Ordnung zulässig, auch beispielsweise eine, die in Implementierungen durch Sortieren nach den vollqualifizierten Namen der direkten Supertypen entsteht.

Meta-Beziehung teilhaben, das heißt, selbst ein oder mehrere Attribuierbares-MetaObjekte als Subtyp aufweisen. Umgekehrt *kann* (MinSourceCard=„0“) ein AttribuierbaresMetaObjekt *ein oder mehrere*[278] (MaxSourceCard=„N“) AttribuierbareMetaObjekte als Supertyp aufweisen.

Die folgende Tabelle 3-30 beinhaltet sämtliche Definitionen für den Meta-Meta-Beziehungstyp „HasSubtype“.

Name des Attributs	**Bedeutung**
Name	***HasSubtype***
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*N*
Description	*Dieser Beziehungstyp erlaubt die Definition eines Klassifikations-/Generalisierungsbaumes.*[279]
Usage	*Es werden sämtliche MetaAttribute sowie sämtliche Meta-Beziehungen von Supertypen ererbt. Die Namen der ererbten und lokalen MetaAttribute bilden eine Menge, das heißt, es dürfen wie weiter oben erwähnt, keine Namensduplikate als Folge der Vererbung auftreten.*
Aliases	–
Constraints	*Ein Supertyp darf weder direkt noch indirekt der Supertyp von sich selbst sein.*

Tabelle 3-30: Meta-Meta-Beziehung „0:N AttributableMetaObject.*HasSubtype*.-AttributableMetaObject 0:N“

278 Wenn ein AttribuierbaresMetaObjekt mehr als einen Supertyp aufweist, dann handelt es sich um einen Typ mit Mehrfachvererbung.

279 Im EIA/CDIF-Meta-Modell „Foundation“, das weiter unten vorgestellt wird, stellt das AttribuierbareMetaObjekt „RootObject“ nur den Supertyp für die MetaEntität „RootEntity“ und die MetaBeziehung „RootEntity.*IsRelatedTo*.RootEntity“ dar. Somit existieren im Integrierten EIA/CDIF-Meta-Modell genau zwei Instanzen dieser Meta-Meta-Beziehung, bei der „RootObject“ als Quell-MetaEntität auftritt.

3.1.5.4 Meta-Meta-Beziehung „0:N MetaRelationship.-*HasSource*.MetaEntity 1:1“

Der vollqualifizierte Name vom Meta-Meta-Beziehungstyp „HasSource“[280] (deutsch: „HatAlsQuelle“) lautet: „0:N **MetaRelationship**.*HasSource*.MetaEntity 0:N“ (deutsch: „0:N **MetaBeziehung**.*HatAlsQuelle*.MetaEntität 0:N“). Dieser Beziehungstyp erlaubt die Festlegung des Quell-Entitätstyps von MetaBeziehungen.

Aufgrund der gegebenen Kardinalitäten *muß* (MinDestCard=„1“) jede MetaBeziehung *genau einmal* (MaxDestCard=„1“) in einer Instanz dieser Meta-Meta-Beziehung enthalten sein. Umgekehrt *kann* (MinSourceDest=„1“) eine Meta-Entität beliebig oft (MaxSourceCard=„N“) an Instanzen dieser Beziehung partizipieren.

Im Zuge der Spezialisierung mit Hilfe des weiter oben definierten Meta-Meta-Beziehungstyps „HasSubtype“ können auch MetaBeziehungen spezialisiert werden. In einem solchen Fall darf die Minimum-Kardinalität im Wert nicht kleiner und die Maximum-Kardinalität nicht größer als die ererbte sein.

Die folgende Tabelle 3-31 beinhaltet sämtliche Definitionen für den Meta-Meta-Beziehungstyp „HasSource“.

Name des Attributs	**Bedeutung**
Name	***HasSource***
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*1*
MaxDestCard	*1*
Description	*Definiert für eine MetaBeziehung die entsprechende Quell-MetaEntität.*
Usage	–

280 Vgl. [CDIF94b], Seite 63.

Name des Attributs	Bedeutung
Aliases	–
Constraints	*Wenn eine MetaBeziehung über Spezialisierung einer MetaBeziehung gebildet wird, dann muß die MetaEntität oder ein Subtyp davon auch in der Spezialisierung der MetaBeziehung als Quell-Entitätstyp auftreten.* *Die Kardinalitäten von spezialisierten MetaBeziehungen müssen zumindest dieselben Einschränkungen wie die (des) Supertyp(en)s aufweisen.* [281]

Tabelle 3-31: Meta-Meta-Beziehung „0:N **MetaRelationship**.*HasSource*.MetaEntity 1:1“

3.1.5.5 Meta-Meta-Beziehung „0:N MetaRelationship.-*HasDestination*.MetaEntity 1:1“

Der vollqualifizierte Name vom Meta-Meta-Beziehungstyp „HasDestination“[282] (deutsch: „HatAlsZiel“) lautet: „0:N **MetaRelationship**.*HasDestination*.Meta-Entity 1:1“ (deutsch: „0:N **MetaBeziehung**.*HatAlsZiel*.MetaEntität 1:1“). Dieser Beziehungstyp erlaubt die Festlegung des Ziel-Entitätstyps von MetaBeziehungen.

Aufgrund der gegebenen Kardinalitäten *muß* (MinDestCard=„1“) jede MetaBeziehung *genau einmal* (MaxDestCard=„1“) in einer Instanz dieser Meta-Meta-Beziehung enthalten sein. Umgekehrt *kann* (MinSourceDest=„1“) eine MetaEntität beliebig oft (MaxSourceCard=„N“) an Instanzen dieser Beziehung partizipieren.

Im Zuge der Spezialisierung mit Hilfe des weiter oben definierten Meta-Meta-Beziehungstyps „HasSubtype“ können auch MetaBeziehungen spezialisiert werden.

---

281 Dies bedeutet, daß der Wert der Kardinalität im Minimum (Maximum) nicht kleiner (größer) als der kleinste entsprechende Wert der direkt übergeordneten Beziehungstypen sein darf.

282 Vgl. [CDIF94b], Seite 62.

Die folgende Tabelle 3-32 beinhaltet sämtliche Definitionen für den Meta-Meta-Beziehungstyp „HasSource“.

Name des Attributs	**Bedeutung**
Name	***HasDestination***
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*1*
MaxDestCard	*1*
Description	*Zeigt an, daß eine MetaBeziehung eine MetaEntität als Ziel aufweisen muß.*
Usage	–
Aliases	–
Constraints	*Wenn eine MetaBeziehung über Spezialisierung einer MetaBeziehung gebildet wird, dann muß die MetaEntität oder ein Subtyp davon auch in der Spezialisierung der MetaBeziehung als Ziel-Entitätstyp auftreten.* *Die Kardinalitäten von spezialisierten MetaBeziehungen müssen zumindest dieselben Einschränkungen wie die (des) Supertyp(en)s aufweisen.*[283]

Tabelle 3-32: Meta-Meta-Beziehung „0:N **MetaRelationship**.*HasDestination*.Meta-Entity 1:1“

283 Vgl. hierzu auch Fußnote 281 auf Seite 95.

3.2 Verteilung des EIA/CDIF-Meta-Meta-Modells mit Hilfe von CORBA

Das EIA/CDIF-Komitee begann 1996[284] mit den Arbeiten der Erstellung von Definitionen, die es ermöglichen sollen, EIA/CDIF-Meta-Modelle und darauf aufbauende Modelldaten in einer verteilten Umgebung in einer standardisierten Form verfügbar zu machen. Dies erfolgt, indem der CORBA-Standard[285] der Object Management Group (abgekürzt: „OMG“) eingesetzt wird: für das EIA/CDIF-Meta-Meta-Modell und das fundamentale EIA/CDIF-Meta-Modell „Foundation“ sowie für die Erstellung weiterer Meta-Modelle beziehungsweise für die Erweiterung bestehender EIA/CDIF-Meta-Modelle werden entsprechende Namensraumregeln beziehungsweise Gültigkeitsbereichsregeln, Datentyp- und Ausnahmedefinitionen sowie die entsprechenden Schnittstellen dafür in OMG IDL[286] definiert. Dieser EIA/CDIF-Standard wurde in [CDIF97b] mit dem Titel „CDIF Transfer Format – OMG IDL Bindings“ veröffentlicht und wird in diesem Abschnitt prägnant beschrieben.

3.2.1 EIA/CDIF-Datentypdefinitionen in OMG IDL

Für die unterschiedlichen EIA/CDIF-Datentypen werden eigene OMG IDL-Datentypen[287] festgelegt, die alle mit Hilfe einer *union*-Struktur in den Datentyp „AnyDT“ zusammengefaßt werden. Die Aufzählungspunkte 1-13 in der folgenden Abbildung 3-2 legen die OMG IDL-Definitionen für die Repräsentation der EIA/CDIF-Datentypen[288] fest, Aufzählungspunkt Nummer 14 die Definition des Datentyps „AnyDT.

---

284 Ein Überblick für den zweiten Entwurf dieses Standards wird in [Flat97a] gegeben.

285 Vgl. hierzu [CORBA98].

286 Mit Hilfe der IDL (englisches Akronym für: „Interface Definition Language“, deutsch: Schnittstellendefinitionssprache) können Schnittstellen zu Objekten definiert werden, die je nach Zielprogrammiersprachen mit unterschiedlichen Implementierungen umgesetzt werden und dessen ungeachtet miteinander lokal oder über Netzwerke verteilt kommunizieren können.

287 Vgl. hierzu auch [CORBA98] Kapitel 3, „OMG IDL Syntax and Semantics“; unter anderem wird für den Zeichensatz der OMG IDL (Seite 3-3) ISO Latin 1 (8859.1) festgelegt.

288 Die Datentypdefinitionen für *Enumerated, Identifier* und für *String* werden im Unterschied zu den Definitionen im Standard nicht mit einem *typedef* mit *Istring*, sondern mit *string* festgelegt. Obwohl in [CDIF97b] auf Seite 5 im Abschnitt 3.2, „Character Sets“, die Rede von einem OMG IDL-Datentyp „Istring“ ist, wurde dieser Datentyp nicht in die CORBA Version 2.2 vom Februar 1998 (vgl. [CORBA98]) aufgenommen.

```
[1]  struct PixelDT {
           long redIntensity;
           long greenIntensity;
           long blueIntensity;
      };
      typedef sequence<PixelDT> BitMapRowDT;
      typedef sequence<BitMapRowDT> BitMapDT;
[2]  typedef boolean BooleanDT;
[3]  enum DateClassValueDT {
           DateAbsolute,
           DateRelativePositive,
           DateRelativeNegative
      };
      struct DateDT {
           short year; // positive
           short month; // within 1..12
           short day; // within 1..31
           DateClassValueDT dateClassValue;
      };
[4]  typedef string EnumeratedDT;
[5]  typedef double FloatDT;
[6]  typedef string IdentifierDT;
[7]  typedef long IntegerDT;
[8]  typedef sequence<IntegerDT> IntegerListDT;
[9]  struct PointDT {
           long xvalue;
           long yvalue;
           long zvalue;
      };
[10] typedef sequence<PointDT> PointListDT;
[11] typedef string StringDT;
[12] typedef sequence<octet> TextDT;[289]
[13] enum TimeClassDT {
           TimeAbsoluteUTC,
           TimeAbsoluteLocal,
           TimeRelativePositive,
           TimeRelativeNegative
      };
      struct TimeDT {
           short hours; // within 0...23
           short minutes; // within 0...59
           float seconds; // within 0...59
           TimeClassDT timeClassValue;
```

289 Es ist bemerkenswert, daß für die Repräsentation des EIA/CDIF-Datentyps „Text“ eine IDL *octet* -Sequenz definiert wird, deren Interpretation von den Anwendungen abhängt.

```
     };
[14] union AnyDT switch( short ) {
          case 0:[290]  short             nullValue;
          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;
          case 12:   TextDT             theTextValue;
          case 13:   TimeDT             theTimeValue;
          default:[291] short           ignore;
     };
```

Abbildung 3-2: [CDIF97b] OMG IDL-Definition der Datentypen[292]

3.2.2 EIA/CDIF-Ausnahmendefinitionen in OMG IDL

Für die festgelegten Schnittstellen werden in [CDIF97b], Seite 12, die in Tabelle 3-33 festgelegten Ausnahmebedingungen definiert. Sie geben damit darüber Auskunft, welche Fehlerbedingungen für die Zugriffe über die Schnittstellendefinitionen vorgesehen sind.

290 Damit wird bei Funktionsergebnissen oder bei Parametern die Tatsache repräsentiert, daß „kein Wert“ vorhanden ist, beziehungsweise in einer Zuweisung an ein Attribut dieses einen möglicherweise vorhandenen Wert verlieren soll und als Ergebnis keinen Wert mehr aufweist. Ein allfälliger Wert, der vom Datentyp *short* in der Variable „nullValue“ gespeichert ist, wird entsprechend dem Standard [CDIF97b], Seite 11, ignoriert.

291 Entsprechend dem Standard [CDIF97b], Seite 11, wird der Wert eines derartigen, ungültigen Datentyps einfach ignoriert.

292 Entnommen aus [CDIF97b], Seiten 8 bis 12.

[1]	`exception` **`IllegalDataType`**`{};`
	Die Ausnahmebedingung „ungültiger Datentyp“ wird ausgelöst, wenn der Wert für ein Meta-Attribut nicht vom entsprechenden Datentyp ist.
[2]	`exception` **`ValueMustBePresent`**`{};`
	Die Ausnahmebedingung „Wert muß gegeben sein“ wird ausgelöst, wenn für ein Meta-Attribut, das immer Werte aufweisen muß, kein Wert zugewiesen ist, also in der „AnyDT“-Union den Fall „0“ („nullValue“) repräsentiert.
[3]	`exception` **`IllegalTraversal`**`{};`
	Die Ausnahmebedingung „ungültige Navigation“ wird ausgelöst, wenn das Objekt nicht in einer Meta-Meta-Beziehung beziehungsweise Meta-Beziehung enthalten ist, die für die Navigation herangezogen wird.
[4]	`exception` **`IllegalAccess`**`{};`
	Die Ausnahmebedingung „ungültiger Zugriff“ wird ausgelöst, wenn das angegebene Meta-Attribut nicht für das Zielobjekt (ein Meta-Objekt) definiert ist.

Tabelle 3-33: [CDIF97b] OMG IDL – Definition der Ausnahmebedingungen[293]

3.2.3 OMG IDL-Modul „CDIF“ und Überführung von EIA/CDIF-Bezeichner in OMG IDL-Bezeichner

In [CDIF97b] wird das Modul „CDIF“ definiert, das den Namensraum für das EIA/CDIF-Meta-Meta-Modell und den standardisierten EIA/CDIF-Meta-Modellen, gemeinsam mit den Datentyp- und Ausnahmedefinitionen umfaßt. Jedes standardisierte EIA/CDIF-Meta-Modell definiert einen im Modul „CDIF“ eingebetteten Namensraum, jedes nicht-standardisierte Meta-Modell sowie die Erweiterungen zu standardisierten EIA/CDIF-Meta-Modellen werden *außerhalb* des Moduls „CDIF“ verfaßt und bilden jeweils eigene Gültigkeitsbereiche. Abbildung 3-3 stellt diesen Zusammenhang übersichtsmäßig in Anlehnung an [CDIF97b], Seite 7, dar.

Zusammengesetzte EIA/CDIF-Bezeichner werden in gültige OMG IDL-Bezeichner übergeführt, indem einfach die konkatenierenden Punkte bei der

293 Vgl. [CDIF97b], Seiten 12 bis 13. Grundsätzlich gilt, daß Zeichen, die für OMG IDL Bezeichner nicht erlaubt sind, durch Unterstriche ersetzt werden.

EIA/CDIF-Vollqualifikation von Meta-Beziehungstypen durch Unterstriche ersetzt werden.[294]

```
[1]  module CDIF {
           <Definitionen der Datentypen>
           <Definitionen der Ausnahmebedingungen>
           <Definition des EIA/CDIF-Meta-Meta-Modells>
           <Definitionen der standardisierten EIA/CDIF-Meta-Modelle>[295]
      };
      module Meta_Modell_1 {};
      module Meta_Modell_2 {};
     ...
      module Meta_Modell_n {};
     ...
```

Abbildung 3-3: OMG IDL – Geltungsbereiche der Moduldefinitionen

„Meta_modell_n“ in Abbildung 3-3 steht für OMG IDL-Module mit Schnittstellendefinitionen, die für nichtstandardisierte Meta-Modelle oder für Erweiterungen von standardisierten Meta-Modellen erstellt werden.

3.2.4 OMG IDL-Definitionen für das EIA/CDIF-Meta-Meta-Modell[296]

Abbildung 3-4 stellt die Schnittstellendefinitionen in der OMG IDL für das EIA/CDIF-Meta-Meta-Modell dar. Bemerkenswerterweise können die Ausprägungen der Meta-Meta-Attribute nur abgefragt[297], nicht aber verändert werden. Somit ist es über diese standardisierten Schnittstellen nur möglich, die Struktur von (verteilten) Meta-Modellen abzufragen, nicht aber sie in irgendeiner Art und Weise zu ändern.[298]

---

294 Entsprechend [CDIF97b], Seiten 26 bis 27, gilt zusätzlich, daß die Namen von MetaAttributen durch einen weiteren Unterstrich mit den (vollqualifizierten) Namen der AttribuierbarenMetaObjekte verbunden werden. Beispielsweise würde die vollqualifizierte Bezeichnung des MetaAttributs „DateCreated“ des AttribuierbarenMetaObjekts „RootObject“ (vgl. hierzu [CDIF94f]) demgemäß „RootObject_DateCreated“ lauten.

295 Im Geltungsbereich des Moduls „CDIF“ dürfen nur von EIA/CDIF-Standardisierte Meta-Modelle definiert sein.

296 Vgl. in diesem Zusammenhang [CDIF97b], Seiten 13 bis 14, sowie Seite 21ff.

297 Dies folgt aus der Tatsache, daß lediglich *get*-Schnittstellen definiert sind.

298 [CDIF97b] stellt die (verteilten) Modelle in den Mittelpunkt und erlaubt, sie zu ändern, wie dies weiter unten noch zur Diskussion gestellt wird. Die Meta-Modelle selbst, werden hingegen absichtlich unveränderlich gestellt (vgl. [CDIF97b], Seite 41, Überschrift „Question 11“).

Die Reflexivität des Meta-Meta-Modells wird in Abbildung 3-4[299] durch die Typdefinitionen *„typedef MetaEntity MetaMetaEntity“* und *„MetaRelationship MetaMetaRelationship“* festgelegt. Somit legt dieser Standard fest, daß das OMG IDL Meta-Meta-Modell des EIA/CDIF-Meta-Meta-Modells durch sich selbst beschreibbar ist.[300]

```
[1]  module MetaMetaModel_02_00 {
          interface MetaObject;
          interface MetaEntity;
          interface MetaRelationship;
          typedef sequence<MetaObject> MetaObject_Set;
          typedef MetaEntity MetaMetaEntity;
          typedef MetaRelationship MetaMetaRelationship;
[2]       interface MetaObject {
            AnyDT getAliases();
            AnyDT getCDIFMetaIdentifier();
            AnyDT getConstraints();
            AnyDT getDescription();
            AnyDT getName();
            AnyDT getUsage();
[3]         MetaMetaEntity get_metaMetaEntity();
[4]         boolean is_identical( in MetaObject otherObject );
[5]         MetaObject_Set traverseSrcToDest(
                 in MetaMetaRelationship aMetaMetaRelationship )
                 raises( IllegalTraversal );
[6]         MetaObject_Set traverseDestToSrc(
                 in MetaMetaRelationship aMetaMetaRelationship )
                 raises( IllegalTraversal );
          };
[7]       interface SubjectArea : MetaObject {
             AnyDT getVersionNumber();
          };
[8]       interface CollectableMetaObject : MetaObject {
          };
[9]       interface MetaAttribute : CollectableMetaObject {
            AnyDT getDataType();
            AnyDT getDomain();
            AnyDT getIsOptional();
```

299 Die Numerierung der einzelnen Absätze (Anweisungen) ist nicht Teil der OMG IDL, sondern erlaubt es lediglich in den danach folgenden Ausführungen, Teile der OMG IDL eindeutig zu referenzieren.

300 Andernfalls müßte für die Abbildung des Meta-Meta-Modells ein weiteres Modell definiert werden, das dann folgerichtig als Meta-Meta-Meta-Modell bezeichnet werden müßte, usw.

```
            AnyDT getLength();
          };
[10]      interface AttributableMetaObject:CollectableMetaObject{ };
[11]      interface MetaRelationship : AttributableMetaObject {
            AnyDT getMinSourceCard();
            AnyDT getMaxSourceCard();
            AnyDT getMinDestCard();
            AnyDT getMaxDestCard();
          };
[12]      interface MetaEntity : AttributableMetaObject {
            AnyDT getType();
          };
      }; /* module MetaMetaModel_02_00 */
```

Abbildung 3-4: OMG IDL Definitionen für das EIA/CDIF-Meta-Meta-Modell

Zusätzlich zu den für die Abbildung des EIA/CDIF-Meta-Meta-Modells definierten Schnittstellen, finden sich in Abbildung 3-4 die folgenden, bemerkenswerten Funktionen:

1. *„MetaMetaEntity get_metaMetaEntity()“* in Nummer [3]
   Diese Funktion liefert als Ergebnis den Typ des Objekts.

2. *„is_identical()“* in Nummer [4]
   Diese Funktion liefert den Boole'schen Wert *True*, wenn der Wert im Meta-Meta-Attribut *CDIFMetaIdentifier* mit dem Wert im Meta-Meta-Attribut *CDIFMetaIdentifier* des Arguments übereinstimmt, *False* sonst.

3. *„traverseSrcToDest()“* in Nummer [5]
   Diese Funktion liefert als Ergebnis eine Menge[301] von Instanzen vom Typ Meta-Meta-Entität, mit denen das MetaObjekt[302] selbst als Quelle über Instanzen eines als Argument angeführten Meta-Meta-Beziehungstyps in Relation steht. Die Ausnahmebedingung *IllegalTraversal* wird dann aufgeworfen, wenn das MetaObjekt selbst an der angegebenen Instanz des Meta-Meta-Beziehungstyps nicht Anteil haben kann.[303] Nachdem sämtli-

---

301 Man beachte, daß die OMG IDL Definition aus implementationstechnischen Überlegungen eine *Sequence* definiert, jedoch konzeptionell eine Menge repräsentiert. Vgl. hierzu auch die Ausführungen in [CDIF97b], Seite 42, unter der Überschrift „Question 13“.

302 Es handelt sich hierbei um eine Instanz vom Typ Meta-Meta-Entität.

303 Eine Instanz vom Meta-Meta-Entitätstyp „AttributableMetaObject“ kann beispielsweise nicht über eine Instanz des Meta-Meta-Beziehungstyps „IsLocalMetaAttributeOf“ als *Quelle* mit Instanzen vom Meta-Meta-Entitätstyp „MetaAttribute“ assoziiert werden.

che konkreten Quell-Meta-Meta-Entitäten aufgrund der strukturellen Bedingungen mindestens einmal in einer Meta-Meta-Beziehung enthalten sein müssen, darf als Ergebnis nie eine leere Menge zurückgegeben werden! [304]

4. *„traverseDestToSrc()“* in Nummer [6]

   Diese Funktion liefert als Ergebnis eine Menge[305] von Instanzen vom Typ Meta-Meta-Entität, mit denen das MetaObjekt[306] selbst als Ziel über Instanzen eines als Argument angeführten Meta-Meta-Beziehungstyps in Relation steht. Die Ausnahmebedingung *IllegalTraversal* wird dann aufgeworfen, wenn das MetaObjekt selbst an der angegebenen Instanz des Meta-Meta-Beziehungstyps nicht Anteil haben kann.[307] Sofern ein MetaObjekt nicht an einer gültigen Meta-Meta-Beziehung[308] teilhat, wird als Ergebnis eine leere Menge geliefert.

304 Vgl. hierzu beispielsweise die Intension des EIA/CDIF-Meta-Meta-Modells wie sie in Abbildung 3-1 auf Seite 53 dargestellt ist.

305 Man beachte, daß die OMG IDL Definition aus implementationstechnischen Überlegungen eine *Sequence* definiert, jedoch konzeptionell eine Menge repräsentiert. Vgl. hierzu auch die Ausführungen in [CDIF97b], Seite 42, unter der Überschrift „Question 13“.

306 Es handelt sich hierbei um eine Instanz vom Typ Meta-Meta-Entität.

307 Eine Instanz vom Meta-Meta-Entitätstyp „MetaAttribute“ kann beispielsweise nicht über eine Instanz des Meta-Meta-Beziehungstyps „IsLocalMetaAttributeOf“ als *Ziel* mit Instanzen vom Meta-Meta-Entitätstyp „AttributableMetaObject“ assoziiert werden.

308 Wenn eine Instanz vom Typ „MetaEntity“ in keiner Beziehung zu sich selbst oder zu anderen Instanzen steht, kann sie demgemäß auch nicht in irgendeiner Instanz des Meta-Meta-Beziehungstyps „HasSource“ als Ziel enthalten sein.

3.3 Abbildungen des EIA/CDIF-Meta-Meta-Modells

Sofern die EIA/CDIF-Meta-Modelle in maschinenverarbeitbarer Form zur Verfügung gestellt werden sollen, müssen entsprechende Spezifaktionen erstellt und anhand von Implementierungsversuchen überprüft werden. Es liegt nahe, daß dazu die Definitionen des EIA/CDIF-Meta-Meta-Modells benutzt werden, das ja die Struktureigenschaften von EIA/CDIF-Meta-Modellen festlegt.

Im Vergleich zu [KapSch96][309] wird der Unterschied zwischen dem Begriff *Typ* und *Klasse* in dieser Arbeit etwas allgemeiner getroffen, indem durch einen *Typ* die Spezifikation der Struktur von Konzepten des EIA/CDIF-Meta-Meta-Modells erfolgt, eine *Klasse* hingegen stellt die Implementierung des *Typs*[310] in irgendeiner objektorientierten Programmiersprache dar.[311] Entsprechend werden in weiterer Folge für relationale Datenbankverwaltungssysteme, die dem SQL89- beziehungsweise dem SQL92-Standard[312] folgen, *Tabellen* den *Typ* implementieren.

Abbildung 3-5 stellt das EIA/CDIF-Meta-Meta-Modell mit den Meta-Meta-Attributen dar, die für die entsprechenden Meta-Meta-Entitäten definiert sind und über die Vererbung auch den untergeordneten Entitätstypen zur Verfügung stehen. Sowohl die vorgeschlagene Implementierung für relationale Daten-

309 Ebenda wird in den Ausführungen auf Seite 160 ein Objekt*typ* zur Bezeichnung der Spezifikation von Struktur und Verhalten benutzt, der Begriff Objekt*klasse* hingegen als Behälter für eine Menge an Objekten aufgefaßt.
Die Object Database Management Group benutzt in ihren Definitionen für die Object Definition und Query Language (vgl. [ODMG97]) den Begriff *Typ* sowohl für die Spezifikation als auch in bezug auf die Bildung von Extensionen (englisch: „extents“). Der Begriff „Klasse“ (englisch: „class“) kommt in dieser summarischen Darstellung überhaupt nicht vor.

310 Somit könnten für ausführbare Spezifikationssprachen wie beispielsweise Eiffel beide Begriffe zusammenfallen.

311 Somit entfällt hier die Forderung von [KapSch96], daß *Klassen* in jedem Fall Behälter für Objekte sein müssen. Beispielsweise wäre dies für die Programmiersprache Object Rexx nicht der Fall, wenngleich durch das Einfügen entsprechender Klassenmethoden eine derartige Eigenschaft zur Verfügung gestellt werden könnte.

312 Vgl. hierzu beispielsweise [DatDar97] oder [PanTau98].

bankverwaltungssysteme[313] als auch die für objektorientierte Programmiersprachen[314] werden diese Strukturen 1:1 abbilden.

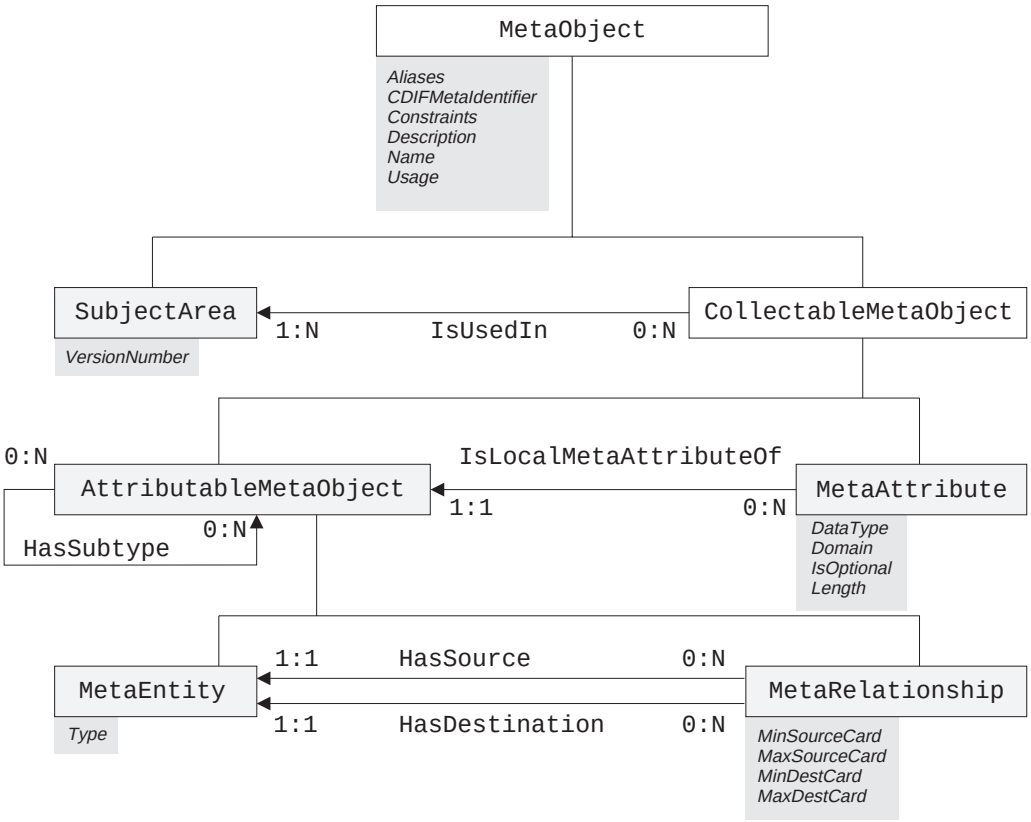


Abbildung 3-5: Das EIA/CDIF-Meta-Meta-Modell mit Meta-Meta-Attributen

313 Ein Implementierungsbeispiel in Oracle erfolgt im Anhang im Abschnitt 6.2.1, „Implementierung des EIA/CDIF-Meta-Meta-Modells in ORACLE“, auf Seite 424ff.

314 Ein Implementierungsbeispiel in der objektorientierten Programmiersprache Object Rexx erfolgt im Anhang im Abschnitt 6.2.2, „Implementierung des EIA/CDIF-Meta-Meta-Modells in Object Rexx“, Seite 449ff.

3.3.1 Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf relationale Datenbankverwaltungssysteme

Die Spezifikation der Strukturen der einzelnen Meta-Meta-Entitätstypen und der Meta-Meta-Beziehungstypen erfolgt im SQL92-Standard[315]. Die Implementierung in ORACLE wird im Anhang im Abschnitt 6.2.1, „Implementierung des EIA/CDIF-Meta-Meta-Modells in ORACLE“ auf Seite 424ff, gemeinsam mit allen VIEW-, TRIGGER- und Prozedurdefinitionen beschrieben.

3.3.1.1 Relational Model of Tasmania (RM/T)

Das in diesem Abschnitt zugrundegelegte Verständnis der Abbildung von Sub- und Supertypen auf relationale Tabellen geht ursprünglich auf Codds „Relational Model/Tasmania“ (abgekürzt: RM/T) in [Codd79] zurück.[316]

Codd unterscheidet im RM/T zwischen unterschiedlichen Relationen, wobei die beiden wichtigsten die *E-Relationen*[317] und die *P-Relationen*[318] sind, die beide gemeinsam zur Bildung von relationalen Tabellen benutzt werden. Für alle Relationen gilt, daß die einzelnen Tupel über eine vom Benutzer unter normalen Umständen nicht sichtbaren Eigenschaft *unabhängig* von weiteren benutzerdefinierten Eigenschaften eindeutig identifiziert werden. Dies wird als

---

315 Vgl. hierzu beispielsweise [DatDar97] oder [PanTau98].

316 Vgl. hierzu auch die Ausführungen zu Codd’s RM/T von [Date89] in Kapitel 25.3 auf Seite 610ff, ausführlich in [Date85] in Kapitel 6 auf Seite 241ff, sowie ein prägnanter Überblick in [Flat90b] im Kapitel 3.5.4 auf Seite 68ff.

317 ***E**-Relationen* (vgl. [Codd79], Seite 410ff) repräsentieren **E**ntitätstypen, die in „Kernel“, „Associative“ und „Characteristic“ eingeteilt werden, je nachdem ob Entitäten vom entsprechenden Typ eigenständig existieren können, einen Beziehungstyp repräsentieren oder Kernel-Entitäten näher beschreiben und davon existenzabhängig sind (auch als „weak entities“ – „schwache Entitäten“ – bezeichnet).
Für assoziative Entitätstypen wird eine ***AG**-Relation* (Abkürzung für englisch: „**A**ssociation **G**raph Relation“) definiert, mit deren Hilfe bestimmt wird, welche *E-Relationen* miteinander in Bezug gesetzt werden. Entsprechend erlaubt eine ***CG**-Relation* (Abkürzung für englisch: „**C**haracteristic **G**raph Relation“) die Zuordnung von charakterisierenden *E-Relationen* zu den entsprechenden *Kernel E-Relationen*, die dadurch näher beschrieben weren.

318 P-Relationen (vgl. [Codd79], Seite 412ff) repräsentieren „Properties“, also Eigenschaften, die Entitätstypen aufweisen und in relationalen Tabellen als Spalten dargestellt sind. Eine ***PG**-Relation* (Abkürzung für englisch: „**P**roperty **G**raph Relation“) erlaubt die Zuordnung der einzelnen *P-Relationen* über einen Fremdschlüssel zu *E-Relationen*.

„Surrogat“[319] bezeichnet. Für jede Entität (englisch: „**e**ntity“) *e* wird ein Eintrag in der ***E**-Relation* durchgeführt, sowie für jede Eigenschaft (englisch: „**p**roperty“) der Entität *e* ein entsprechender Eintrag in der ***P**-Relation*.

Im Zusammenhang mit der Möglichkeit der Subtypbildung, die in [Codd79] unter dem Gesichtswinkel der Generalisierung untersucht wird, sind die folgenden beiden RM/T-Regeln wichtig:

1) „Property inheritance rule: Given any subtype e, all of the properties of its parent type(s) are applicable to e.“[320]

2) „Rule 7 (subtype integrity): Whenever a surrogate (say s) belongs to the E-relation for an entity of type e, s must also belong to the E-relation for each entity type of which e is a subtype.“[321]

Regel 2) hat zur Folge, daß in sämtlichen Entitätstypen das entsprechende E-Surrogat eingefügt werden muß. In Tabellen relationaler Datenbanken erfolgt dies durch den Eintrag eines entsprechenden Tupels in den die Supertypen repräsentierenden Tabellen.

RM/T legt darüber hinaus für die Definition von Generalisierungshierarchien eine *UGI“[322]-Relation* fest, die festhält, welcher Entitätstyp welchem anderen aufgrund welcher Kategorie übergeordnet ist. Damit kann ein RM/T-Datenbankverwaltungssystem zumindest die Subtypintegritätsbedingung gewährleisten, und zum anderen mit Hilfe der *PG-Relation* die obige Regel 1) erfüllen.

---

319 Surrogatwerte müssen vom Datenbankverwaltungssystem erzeugt werden und sollen über die gesamte Datenbank eindeutig sein. Zudem gilt, daß Surrogatwerte in ihrem Wert nicht mehr verändert werden dürfen, noch im Falle, daß ein Tupel gelöscht wird und damit ein Surrogatwert theoretisch wieder frei wird, wiederverwendet werden.
Durch die Definition eines eigenständigen Surrogats für jede RM/T-Relation wird unter anderem die Entitätsintegritätsbedingung sehr einfach und effizient erfüllt.

320 [Codd79], Seite 420, deutsche Übersetzung: „Eigenschaftsvererbungsregel: Für einen beliebigen Subtypen e stehen alle Eigenschaften von seine(m) Elterntyp(en) e zur Verfügung.“

321 [Codd79], Seite 421, deutsche Übersetzung: „Subtypintegrität: Wann immer ein Surrogat (zum Beispiel „s“) zu einer E-Relation vom Typ e gehört, muß s auch zu allen E-Relationen gehören, von denen e ein Subtyp ist.“

322 „UGI“ stellt die Abkürzung für „**U**nconditional **G**en **I**nclusion“ dar und beinhaltet die Namen der Entitätstypen in den Eigenschaften „SUB“ und „SUP“, sowie den Namen einer Kategorie „CAT“, aufgrund der die Generalisierung durchgeführt wird. Vgl. in diesem Zusammenhang [Codd79], Seite 419ff, sowie [Date83], Seite 263ff.

3.3.1.2 Tabellendefinitionen

Die folgenden Tabellendefinitionen sind in der Syntax von SQL92[323] definiert und wurden im Rahmen dieser Arbeit in der relationalen Datenbank von ORACLE implementiert, die in Anhang 6.2.1, „Implementierung des EIA/CDIF-Meta-Meta-Modells in ORACLE“ auf Seite 424ff, dokumentiert sind. Die Unterschiede zur Implementierung in ORACLE liegen einerseits in der Bezeichnung der Datentypen, andererseits werden in der Dokumentation der Implementierung auch die Definitionen von Triggern und von gespeicherten Oracle PL*SQL-Prozeduren angegeben, die die Integrität der Abbildung der EIA/CDIF-Meta-Meta-Modelle auch in bezug auf die Vererbungsregeln aufrecht erhalten soll.[324]

In den folgenden Tabellendefinitionen beginnen Spalten, die nicht direkt auf Attribute des EIA/CDIF-Meta-Meta-Modells zurückführbar sind, mit einem Kleinbuchstaben und enden mit einem Großbuchstaben. Handelt es sich um zusammengesetzte Wörter, werden innerhalb des Namens die einzelnen Wörter mit einem Großbuchstaben begonnen. Die Bezeichner für Spalten, die EIA/CDIF-Meta-Meta-Attribute repräsentieren, beginnen entsprechend den EIA/CDIF-Konventionen mit einem Großbuchstaben. Damit können in den folgenden Tabellenspezikationen in SQL und darauf aufbauenden Abbildungen – die auf EIA/CDIF-Meta-Meta-Attribute direkt zurückführbaren Spalten – sofort erkannt werden.

Es gelten folgende Regeln:

1. Jede Tabelle besteht zumindest aus einer Surrogatspalte namens „surR“ mit deren Hilfe die Entitätsintegritätsbedingung umgesetzt, das heißt als Primärschlüssel benutzt wird,

---

323 In diesem Zusammenhang muß darauf verwiesen werden, daß für viele SQL92-Datentypen die Dimensionen von Datentypen den entsprechenden Implementierungen von relationalen Datenbankverwaltungssystemen überlassen ist. In den folgenden Tabellendefinitionen wird zumindest für den EIA/CDIF-Datentyp „Text“, der aus einer unbeschränkten Anzahl von Zeichen bestehen kann, ein Wert von maximal 4.000 Zeichen angegeben. Dies erklärt sich daraus, daß aus der Untersuchung des Autors von sämtlichen standardisierten EIA/CDIF-Meta-Modellen zum Zeitpunkt Frühjahr 1998, kein Texteintrag gefunden wurde, der aus mehr Zeichen bestanden hätte.

324 Beispielsweise wird beim Löschen von Tupeln darauf geachtet, daß auch die entsprechenden Tupeln in übergeordneten Tabellen erfolgreich entfernt werden, nachdem aufgrund der automatisch ausgelösten Löschvorgänge Mutating-Table Probleme zur Laufzeit auftreten (können).

2. sämtliche Tabellen, die „MetaObject“ untergeordnete Entitätstypen repräsentieren, werden so definiert, daß sie über eine „DELETE CASCADE“-Foreign-Key-Beziehung über die gemeinsame Surrogatspalte „surR“ die Tabelle für „MetaObject“ referenzieren,[325]

3. Basistabellen erhalten im Unterschied zu Sichtweisen (englisch: „VIEW“) als Konvention einen Unterstrich an den Tabellennamen angefügt,

4. jedes zu speichernde Konzept wird in Form einer Datenzeile gespeichert, wobei zumindest in der entsprechenden Tabelle und allen ihren übergeordneten Tabellen zumindest eine Zeile mit dem Surrogatwert eingetragen wird,[326]

5. die Werte von Attributen einer Datenzeile werden entsprechend ihrer Definitionen auf die verschiedenen Tabellen aufgeteilt, für die sie (lokal) im EIA/CDIF-Standard definiert sind,

6. wenn eine Zeile *z* aus irgendeiner Tabelle gelöscht wird, müssen sämtliche Zeilen mit demselben Surrogatwert von *z* in allen anderen Tabellen gelöscht werden,

7. VIEW-Definitionen verfügen über keinen Unterstrich als letztes Zeichen des Tabellennamens und werden dazu eingesetzt, um Tabellen, die Subtypen repräsentieren, mit sämtlichen Attributen zu versehen, die sich aus einer konzeptionellen Vererbung entsprechend der Property-Inheritance-Regel von RM/T ergeben.

---

325 Damit wird zunächst sichergestellt, daß das Löschen von Zeilen aus der Tabelle, die „MetaObject“ repräsentiert, dazu führt, daß in *sämtlichen* untergeordneten Tabellen die entsprechenden Zeilen gelöscht werden. Damit die Tabellen, die das EIA/CDIF-Meta-Meta-Modell repräsentieren, zueinander konsistent bleiben, muß daher in der Spezifikation das Löschen von Zeilen immer über die Tabelle erfolgen, die „MetaObject“ repräsentiert. Die im Anhang beschriebene Implementierung in Oracle erlaubt dahingegen das Löschen von Zeilen aus beliebigen Tabellen, da mit Hilfe von Triggern und gespeicherten Prozeduren die Konsistenz gewährleistet bleibt.

326 Dies bedeutet beispielsweise für die Tabelle, die den Wurzeltyp „MetaObject“ repräsentiert, daß darin für *alle* gespeicherten Konzepte Einträge zu finden sind. Mit anderen Worten, alle Meta-Objekte von Meta-Modellen finden sich als Datensatz in der Tabelle wieder, die für „MetaObject“ definiert wird.

Die Definition von Tabellen für die Meta-Meta-Entitätstypen und für die Meta-Meta-Beziehungstypen des Integrierten EIA/CDIF-Meta-Meta-Modells[327] erfolgt in einem Top-Down-Vorgehen und innerhalb einer Ebene von links nach rechts.

3.3.1.2.1 Tabellendefinitionen für Meta-Meta-Entitätstypen

Die Namen der Tabellen von Meta-Meta-Entitätstypen werden in Form von Abkürzungen angegeben, indem als Akronym dafür die Großbuchstaben im Namen der entsprechenden Entitätstypen ohne Leerstellen dienen.[328]

Abbildung 3-6 auf Seite 119 stellt die definierten Basistabellen mit jenen Tupeln graphisch dar, die die Meta-Objekte von [CDIF94f] repräsentieren, also das grundlegende EIA/CDIF-Meta-Modell „Foundation“, das direkt oder indirekt von sämtlichen EIA/CDIF-konformen Meta-Modellen referenziert werden muß.

3.3.1.2.1.1 Abbildung des Meta-Meta-Entitätstyps „MetaObject“

In der SQL-Tabellendefinition 3-1 werden für den EIA/CDIF-Meta-Meta-Entitätstyp „MetaObject“[329] zunächst folgende Spalten für die Meta-Meta-Attribute in alphabetischer Reihenfolge angelegt: „Aliases“, „CDIFMetaIdentifier“, „Constraints“, „Description“, „Name“, „Usage“.

Zusätzlich findet sich eine Surrogatspalte „surR“, eine Spalte „mo_typE“, die das Akronym des Meta-Meta-Entitätstyps aufnimmt, für die die Zeile in der Tabelle angelegt wurde, sowie eine Spalte „longNamE“, die für die Aufnahme der vollqualifizierten Namen für Entitäten vom Typ „AttributableMetaObject“[330] vorgesehen ist.

```
CREATE TABLE MO_ (
   surR               NUMERIC( 6 )    ,
   mo_typE            VARCHAR( 5 )    ,
   ALIASES            VARCHAR( 1536 ) ,
   CDIFMetaIdentifier VARCHAR(   40 ) ,
   Constraints        VARCHAR( 2000 ) ,
   Description        VARCHAR( 4000 ) ,
   Name               VARCHAR(   40 ) ,
```

327 Vgl. Abbildung 3-5 auf Seite 107.

328 Dementsprechend ist das Akronym von „MetaEntity“ die Zeichenfolge „ME“.

329 Vgl. dazu auch die Definitionen im Meta-Meta-Modell in Kapitel 3.1.4.1 auf Seite 59ff.

330 Für die instanziierbaren Meta-Objekte vom Typ „SubjectArea“ und „MetaAttribute“ findet sich in dieser Spalte kein Wert.

```
    longNamE            VARCHAR(  120 ) ,
    Usage               VARCHAR( 2000 ) ,
       PRIMARY KEY ( surR )
    ) ;
```

SQL-Tabellendefinition 3-1: Basistabelle „MO_“

Die SQL-Viewdefinition 3-1 ermöglicht den Zugriff auf dieselben Daten wie in Tabelle „MO_“ über den Namen „MO“.

```
CREATE VIEW MO AS
       SELECT * FROM MO_ ;
```

SQL-Viewdefinition 3-1: View „MO“

3.3.1.2.1.2 Abbildung des Meta-Meta-Entitätstyps „SubjectArea“

In der SQL-Tabellendefinition 3-2 wird für den EIA/CDIF-Meta-Meta-Entitätstyp „SubjectArea“[331] zunächst für das Meta-Meta-Attribut „VersionNumber“ eine Spalte angelegt.

Zusätzlich findet sich eine Surrogatspalte „surR“, eine Spalte „shortHanD“, die die EIA-CDIF üblichen Abkürzungen[332] für die unterschiedlichen, standardisierten EIA/CDIF-Meta-Modelle, sowie eine Spalte „cdifNumbeR“, die für die Aufnahme der offiziellen EIA/CDIF Katalognummer[333] des entsprechenden Standards vorgesehen ist. Die Fremdschlüsseldefinition mit der Aktion „DELETE CASCADES“ legt fest, daß das Löschen von Tupeln in der Tabelle „MO_“ dazu führt, daß die entsprechenden Tupeln in der Tabelle „SA_“ automatisch vom Datenbankverwaltungssystem aus gelöscht werden.

```
CREATE TABLE SA_ (
   surR                NUMERIC(  6 )   ,
   VersionNumber       VARCHAR( 16 )   ,
   shortHanD           VARCHAR(  6 )   ,
   cdifNumbeR          VARCHAR( 20 )   ,
      PRIMARY KEY ( surR )             ,
```

331 Vgl. dazu auch die Definitionen im Meta-Meta-Modell in Kapitel 3.1.4.2 auf Seite 66ff.

332 Beispielsweise stellt „FND“ die Abkürzung für „CDIF Integrated Meta-model: Foundation Subject Area“ dar.

333 „EIA/IS-111“ ist beispielsweise die EIA/CDIF-Katalognummer für den Standard: „CDIF Integrated Meta-model: Foundation Subject Area“.

```
        FOREIGN KEY ( surR ) REFERENCES MO_ ( surR )
                             ON DELETE CASCADE
    ) ;
```

SQL-Tabellendefinition 3-2: Basistabelle „SA_“

Die SQL-Viewdefinition 3-2 definiert eine Sicht, in der sämtliche Spalten der Basistabelle[334] „MO_“ und der Basistabelle „SA_“ enthalten sind. Die Zuordnung erfolgt über eine natürliche Verknüpfung über die gemeinsame Spalte „surR“.

```
CREATE VIEW SA AS
       SELECT MO_.*, SA_.shortHanD,
                     SA_.cdifNumbeR,
                     SA_.VersionNumber
       FROM   MO_, SA_
       WHERE  MO_.surR = SA_.surR ;
```

SQL-Viewdefinition 3-2: View „SA“

3.3.1.2.1.3 Abbildung des Meta-Meta-Entitätstyps „CollectableMetaObject“

Nachdem der Meta-Meta-Entitätstyp „CollectableMetaObject“[335] über keine eigenen Meta-Meta-Attribute verfügt, wird in der SQL-Tabellendefinition 3-3 lediglich die Surrogatspalte „surR“ dafür definiert. Die Fremdschlüsseldefinition mit der Aktion „DELETE CASCADES“ legt fest, daß das Löschen von Tupeln in der Tabelle „MO_“ dazu führt, daß die entsprechenden Tupeln in der Tabelle „CMO_“ automatisch vom Datenbankverwaltungssystem aus gelöscht werden.

```
CREATE TABLE CMO_ (
   surR                NUMERIC( 6 )    ,
      PRIMARY KEY ( surR )            ,
      FOREIGN KEY ( surR ) REFERENCES MO_ ( surR )
                           ON DELETE CASCADE
   );
```

SQL-Tabellendefinition 3-3: Basistabelle „CMO_“

---

334 Nachdem die Sicht „MO“ für die Repräsentation des Meta-Meta-Entitätstyps „MetaObject“ über die gleichen Spalten verfügt wie die Basistabelle „MO“, ist es in diesem Fall gleichgültig, ob die Sicht „MO_“ oder die Basistabelle „MO“ für die Defintion der Sicht „SA“ benutzt wird. Die Viewdefinitionen weiter unten nutzen im Gegensatz dazu die bereits definierten Sichten, um die SQL-Anweisung so übersichtlich wie möglich zu gestalten.

335 Vgl. dazu auch die Definitionen im Meta-Meta-Modell in Kapitel 3.1.4.3 auf Seite 69ff.

Da der Meta-Meta-Entitätstyp „CollectableMetaObject“ über keine eigenen Meta-Meta-Attribute verfügt, genügt es für die SQL-Viewdefinition 3-3, daß nur die Spalten von „MO_“ ausgewählt werden und somit auch für den Zugriff über „CMO“ zur Verfügung stehen. Die Zuordnung erfolgt über eine natürliche Verknüpfung über die gemeinsame Spalte „surR“.

```
CREATE VIEW CMO AS
       SELECT MO_.*
       FROM   MO_, CMO_
       WHERE  MO_.surR = CMO_.surR ;
```

SQL-Viewdefinition 3-3: View „CMO“

3.3.1.2.1.4 Abbildung des Meta-Meta-Entitätstyps „AttributableMetaObject“

Nachdem der Meta-Meta-Entitätstyp „AttributableMetaObject“[336] über keine eigenen Meta-Meta-Attribute verfügt, wird in der SQL-Tabellendefinition 3-4 lediglich die Surrogatspalte „surR“ definiert. Die Fremdschlüsseldefinition mit der Aktion „DELETE CASCADES“ legt fest, daß das Löschen von Tupeln in der Tabelle „MO_“ dazu führt, daß die entsprechenden Tupeln in der Tabelle „AMO_“ automatisch vom Datenbankverwaltungssystem aus gelöscht werden.

```
CREATE TABLE AMO_ (
   surR                NUMERIC( 6 )    ,
      PRIMARY KEY ( surR )             ,
      FOREIGN KEY ( surR ) REFERENCES MO_ ( surR )
                           ON DELETE CASCADE
   );
```

SQL-Tabellendefinition 3-4: Basistabelle „AMO_“

Da der Meta-Meta-Entitätstyp „AttributableMetaObject“ über keine eigenen Meta-Meta-Attribute verfügt, genügt es für die SQL-Viewdefinition 3-4, daß nur die Spalten der Sicht von „CMO“ ausgewählt werden und somit auch für den Zugriff über „AMO“ zur Verfügung stehen. Die Zuordnung erfolgt über eine natürliche Verknüpfung über die gemeinsame Spalte „surR“.

```
CREATE VIEW AMO AS
       SELECT CMO.*
```

336 Vgl. dazu auch die Definitionen im Meta-Meta-Modell in Kapitel 3.1.4.4 auf Seite 70ff.

```
        FROM   CMO, AMO_
        WHERE  CMO.surR = AMO_.surR ;
```

SQL-Viewdefinition 3-4: View „AMO“

3.3.1.2.1.5 Abbildung des Meta-Meta-Entitätstyps „MetaAttribute“

In der SQL-Tabellendefinition 3-5 werden für den EIA/CDIF-Meta-Meta-Entitätstyp „MetaAttribute“[337] zunächst folgende Spalten für die Meta-Meta-Attribute in alphabetischer Reihenfolge angelegt: „DataType“, „Domain“, „IsOptional“, „Length“.

Zusätzlich findet sich noch die Surrogatspalte „surR“. Die Fremdschlüsseldefinition mit der Aktion „DELETE CASCADES“ legt fest, daß das Löschen von Tupeln in der Tabelle „MO_“ dazu führt, daß die entsprechenden Tupeln in der Tabelle „MA_“ automatisch vom Datenbankverwaltungssystem aus gelöscht werden.

```
CREATE TABLE MA_ (
   surR                NUMERIC(    6 ) ,
   DataType            VARCHAR(   40 ) ,
   Domain              VARCHAR( 2000 ) ,
   IsOptional          VARCHAR(   10 ) ,
   Length              VARCHAR(   10 ) ,
      PRIMARY KEY ( surR )             ,
      FOREIGN KEY ( surR ) REFERENCES MO_ ( surR )
                           ON DELETE CASCADE
   );
```

SQL-Tabellendefinition 3-5: Basistabelle „MA_“

Die SQL-Viewdefinition 3-5 definiert eine Sicht, in der sämtliche Meta-Meta-Attribute der Sicht „CMO“ und zusätzlich jene Spalten der Basistabelle „MA_“ enthalten sind, die die Meta-Meta-Attribute des Entitätstyps „MetaAttribute“ repräsentieren. Die Zuordnung erfolgt über eine natürliche Verknüpfung über die gemeinsame Spalte „surR“.

```
CREATE VIEW MA AS
       SELECT CMO.*, MA_.DataType, MA_.Domain,
                     MA_.IsOptional, MA_.Length
       FROM   CMO, MA_
```

337 Vgl. dazu auch die Definitionen im Meta-Meta-Modell in Kapitel 3.1.4.5 auf Seite 73ff.

```
        WHERE  CMO.surR = MA_.surR ;
```

SQL-Viewdefinition 3-5: View „MA“

3.3.1.2.1.6 Abbildung des Meta-Meta-Entitätstyps „MetaEntity“

In der SQL-Tabellendefinition 3-6 wird für den EIA/CDIF-Meta-Meta-Entitätstyp „MetaEntity“[338] zunächst für das Meta-Meta-Attribut „Type“ eine Spalte angelegt, sowie die Surrogatspalte „surR“. Die Fremdschlüsseldefinition mit der Aktion „DELETE CASCADES“ legt fest, daß das Löschen von Tupeln in der Tabelle „MO_“ dazu führt, daß die entsprechenden Tupeln in der Tabelle „ME_“ automatisch vom Datenbankverwaltungssystem aus gelöscht werden.

```
CREATE TABLE ME_ (
   surR                NUMERIC(  6 )   ,
   Type                VARCHAR( 40 )   ,
      PRIMARY KEY ( surR )            ,
      FOREIGN KEY ( surR ) REFERENCES MO_ ( surR )
                           ON DELETE CASCADE
   );
```

SQL-Tabellendefinition 3-6: Basistabelle „ME_“

Die SQL-Viewdefinition 3-6 definiert eine Sicht, in der sämtliche Spalten der Sicht „AMO“ und die Spalte „Type“ aus der Basistabelle „ME_“ enthalten sind. Die Zuordnung erfolgt über eine natürliche Verknüpfung über die gemeinsame Spalte „surR“.

```
CREATE VIEW ME AS
       SELECT AMO.*, ME_.Type
       FROM   AMO, ME_
       WHERE  AMO.surR = ME_.surR ;
```

SQL-Viewdefinition 3-6: View „ME“

3.3.1.2.1.7 Abbildung des Meta-Meta-Entitätstyps „MetaRelationship“

In der SQL-Tabellendefinition 3-7 werden für den EIA/CDIF-Meta-Meta-Entitätstyp „MetaRelationship“[339] zunächst folgende Spalten für die Meta-Meta-Attribute angelegt: „MinDestCard“, „MaxDestCard“, „MinSourceCard“,

338 Vgl. dazu auch die Definitionen im Meta-Meta-Modell in Kapitel 3.1.4.6 auf Seite 77ff.

339 Vgl. dazu auch die Definitionen im Meta-Meta-Modell in Kapitel 3.1.4.7 auf Seite 81ff.

„MaxSourceCard“, sowie die Surrogatspalte „surR“. Die Fremdschlüsseldefinition mit der Aktion „DELETE CASCADES“ legt fest, daß das Löschen von Tupeln in der Tabelle „MO_“ dazu führt, daß die entsprechenden Tupeln in der Tabelle „MR_“ automatisch vom Datenbankverwaltungssystem aus gelöscht werden.

```
CREATE TABLE MR_ (
   surR                NUMERIC(    6 ) ,
   MinDestCard         VARCHAR(   10 ) ,
   MaxDestCard         VARCHAR(   10 ) ,
   MinSourceCard       VARCHAR(   10 ) ,
   MaxSourceCard       VARCHAR(   10 ) ,
      PRIMARY KEY ( surR )             ,
      FOREIGN KEY ( surR ) REFERENCES MO_ ( surR )
                           ON DELETE CASCADE
);
```

SQL-Tabellendefinition 3-7: Basistabelle „MR_“

Die SQL-Viewdefinition 3-7 definiert eine Sicht, in der sämtliche Spalten der Sicht „AMO“ und die die Meta-Meta-Attribute von „MetaRelationship“ repräsentierenden Spalten aus der Basistabelle „MR_“ enthalten sind. Die Zuordnung erfolgt über eine natürliche Verknüpfung über die gemeinsame Spalte „surR“.

```
CREATE VIEW MR AS
       SELECT AMO.*,
              MR_.MinSourceCard, MR_.MaxSourceCard,
              MR_.MinDestCard, MR_.MaxDestCard
       FROM   AMO, MR_
WHERE  AMO.surR = MR_.surR ;
```

SQL-Viewdefinition 3-7: View „MR“

3.3.1.2.1.8 Darstellung am Beispiel [CDIF94f]

Die folgende Abbildung 3-6 stellt jene Tupeln dar, die die verschiedenen Meta-Objekte von [CDIF94f] ausschnittweise repräsentieren.[340]

340 Die Tabellenstrukturen stellen hierbei die Intensionen, die Tupeln die entsprechenden Extensionen dar.

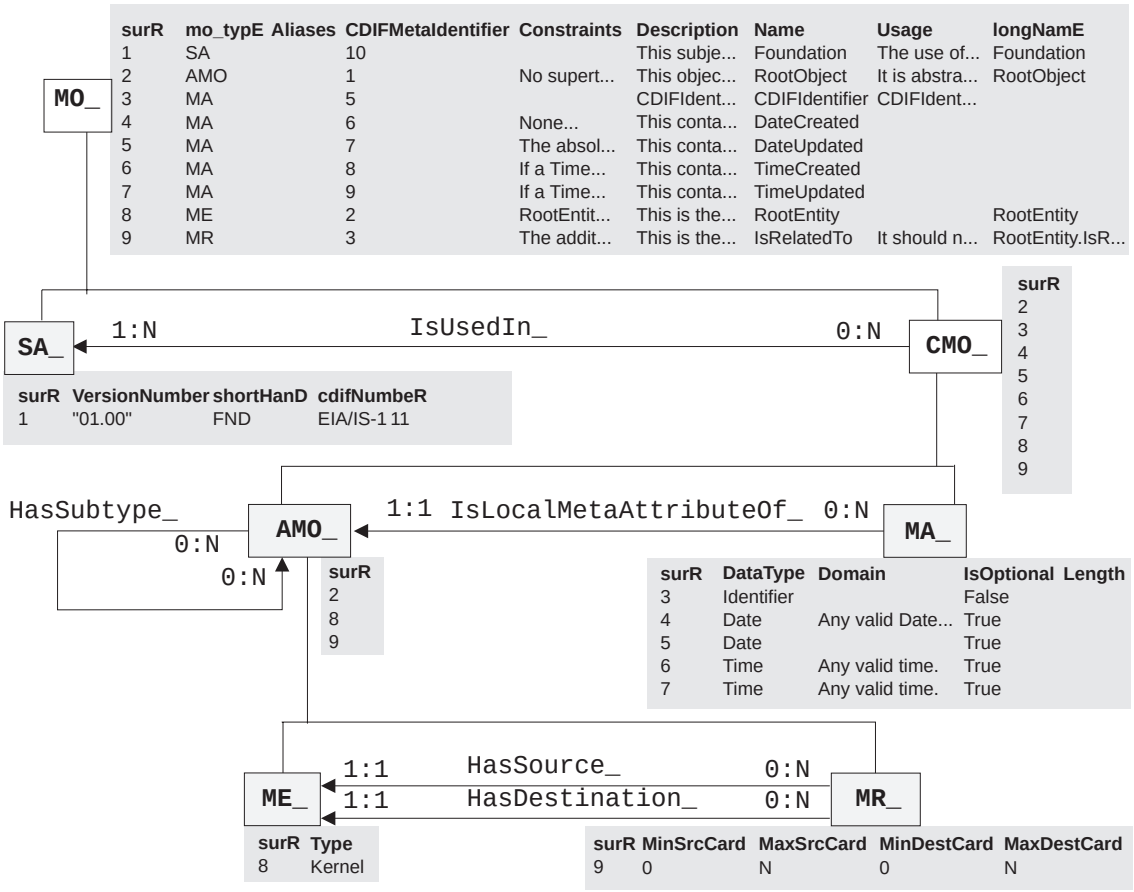


Abbildung 3-6: Tabellen, die Meta-Meta-Entitätstypen repräsentieren und die Auszüge der Meta-Objektdefinitionen für das fundamentale EIA/CDIF-Meta-Modell enthalten[341, 342]

3.3.1.2.2 Tabellendefinitionen für Meta-Meta-Beziehungstypen

Im Unterschied zur Namensbildung der Tabellen, die Meta-Meta-Entitätstypen repräsentieren, werden für die die Meta-Meta-Beziehungstypen repräsentierenden Tabellennamen keine Abkürzungen eingeführt. Damit soll in SQL-Anweisungen, die auf diese Tabellendefinitionen bezug nehmen, jederzeit ersichtlich sein, welche Tabellen Entitätstypen (Namen bestehen aus Akronymen) und welche Beziehungstypen (Namen sind nicht abgekürzt) repräsentieren. Aus Gründen der besseren Handhabbarkeit tragen die Tabellen die einfachen und nicht die vollqualifizierten Namen der Meta-Meta-Beziehungstypen.

---

341 Vgl. dazu [CDIF94f] und Abschnitt 4.1 auf Seite 155ff, nachdem die Extensionen die Definitionen des fundamentalen Meta-Modells „Foundation“ repräsentieren.

342 Hinweis: die Surrogatwerte in dieser Abbildung weisen *willkürlich* von eins aufsteigende Werte auf.

Nachdem die Meta-Meta-Beziehungstypen über keine Meta-Meta-Attribute verfügen, entfällt demgemäß die Definition von Sichten.

Sämtliche Tabellendefinitionen für die Repräsentation der Meta-Meta-Beziehungstypen verfügen über drei Spalten: eine Surrogatspalte („surR“), eine Spalte („Source“) zur Referenzierung der Quellentitätsypen und eine Spalte („Destination“) zur Referenzierung der Zielentitätstypen. Es werden „FOREIGN KEY“-Tabellenbedingungen mit einer „DELETE CASCADE“-Aktion sowohl für die Quell- als auch für die Zieltabellen angegeben.

Abbildung 3-7 auf Seite 123 stellt die definierten Basistabellen mit jenen Tupeln graphisch dar, die die Meta-Objekte von [CDIF94f] repräsentieren, also das grundlegende EIA/CDIF-Meta-Modell „Foundation“, das direkt oder indirekt von sämtlichen EIA/CDIF-Meta-Modellen referenziert werden muß.

3.3.1.2.2.1 Abbildung des Meta-Meta-Beziehungstyps „IsUsedIn“

Die SQL-Tabellendefinition 3-8 zeigt die Abbildung des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ in Form der Basistabelle „IsUsedIn_“.

```
CREATE TABLE IsUsedIn_ (
   surR                    NUMERIC( 6 )      ,
   Source                  NUMERIC( 6 )      ,
   Destination             NUMERIC( 6 )      ,
      FOREIGN KEY ( Source ) REFERENCES CMO_ ( surR )
                               ON DELETE CASCADE ,
      FOREIGN KEY ( Destination ) REFERENCES SA_ ( surR )
                                    ON DELETE CASCADE ,
      PRIMARY KEY ( surR )
   );
```

SQL-Tabellendefinition 3-8: Basistabelle „IsUsedIn_“

3.3.1.2.2.2 Abbildung des Meta-Meta-Beziehungstyps „IsLocalMetaAttributeOf“

Die SQL-Tabellendefinition 3-9 zeigt die Abbildung des Meta-Meta-Beziehungstyps „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMeta-Object 1:1“ in Form der Basistabelle „IsLocalMetaAttributeOf_“.

```
CREATE TABLE IsLocalMetaAttributeOf_ (
```

```
    surR                    NUMERIC( 6 )      ,
    Source                  NUMERIC( 6 )      ,
    Destination             NUMERIC( 6 )      ,
       FOREIGN KEY ( Source ) REFERENCES MA_ ( surR )
                               ON DELETE CASCADE ,
       FOREIGN KEY ( Destination ) REFERENCES AMO_ ( surR )
                                    ON DELETE CASCADE ,
       PRIMARY KEY ( surR )
    );
```

SQL-Tabellendefinition 3-9: Basistabelle „IsLocalMetaAttributeOf_“

3.3.1.2.2.3 Abbildung des Meta-Meta-Beziehungstyps „HasSubtype“

Die SQL-Tabellendefinition 3-10 zeigt die Abbildung des Meta-Meta-Beziehungstyps „0:N AttributableMetaObject.*HasSubtype*.AttributableMeta-Object 0:N“ in Form der Basistabelle „HasSubtype_“.

```
CREATE TABLE HasSubtype_ (
   surR                    NUMERIC( 6 )      ,
   Source                  NUMERIC( 6 )      ,
   Destination             NUMERIC( 6 )      ,
      FOREIGN KEY ( Source ) REFERENCES AMO_ ( surR )
                               ON DELETE CASCADE ,
      FOREIGN KEY ( Destination ) REFERENCES AMO_ ( surR )
                                    ON DELETE CASCADE ,
      PRIMARY KEY ( surR )
   );
```

SQL-Tabellendefinition 3-10: Basistabelle „HasSubtype_“

3.3.1.2.2.4 Abbildung des Meta-Meta-Beziehungstyps „HasSource“

Die SQL-Tabellendefinition 3-11 zeigt die Abbildung des Meta-Meta-Beziehungstyps „0:N **MetaRelationship**.*HasSource*.MetaEntity 1:1“ in Form der Basistabelle „HasSource_“.

```
CREATE TABLE HasSource_ (
   surR                    NUMERIC( 6 )      ,
   Source                  NUMERIC( 6 )      ,
   Destination             NUMERIC( 6 )      ,
      FOREIGN KEY ( Source ) REFERENCES MR_ ( surR )
                               ON DELETE CASCADE ,
      FOREIGN KEY ( Destination ) REFERENCES ME_ ( surR )
                                    ON DELETE CASCADE ,
```

```
      PRIMARY KEY ( surR )
   );
```

SQL-Tabellendefinition 3-11: Basistabelle „HasSource_“

3.3.1.2.2.5 Abbildung des Meta-Meta-Beziehungstyps „HasDestination“

Die SQL-Tabellendefinition 3-12 zeigt die Abbildung des Meta-Meta-Beziehungstyps „0:N **MetaRelationship**.*HasDestination*.MetaEntity 1:1“ in Form der Basistabelle „HasDestination_“.

```
CREATE TABLE HasDestination_ (
   surR                   NUMERIC( 6 )      ,
   Source                 NUMERIC( 6 )      ,
   Destination            NUMERIC( 6 )      ,
      FOREIGN KEY ( Source ) REFERENCES MR_ ( surR )
                               ON DELETE CASCADE ,
      FOREIGN KEY ( Destination ) REFERENCES ME_ ( surR )
                                    ON DELETE CASCADE ,
      PRIMARY KEY ( surR )
   );
```

SQL-Tabellendefinition 3-12: Basistabelle „HasDestination_“

3.3.1.2.2.6 Darstellung am Beispiel [CDIF94f]

Die folgende Abbildung 3-7 stellt gemeinsam mit Abbildung 3-6 auf Seite 119 Tupeln dar, die die verschiedenen Meta-Objekte von [CDIF94f] über Instanzen der Meta-Meta-Beziehungstypen miteinander in Beziehung setzen.

CMO_.IsUsedIn_.SA_

surR	sourcE	destinatioN
10	2	1
11	8	1
12	9	1
13	3	1
14	4	1
15	5	1
16	6	1
17	7	1

AMO_.HasSubtype_.AMO_

surR	sourcE	destinatioN
18	2	8
19	2	9

MR_.HasSource_.ME_

surR	sourcE	destinatioN
20	9	8

MR_.HasDestination_.ME_

surR	sourcE	destinatioN
21	9	8

MA_.IsLocalMetaAttributeOf_.AMO_

surR	sourcE	destinatioN
22	3	2
23	5	2
24	6	2
25	4	2
26	7	2

Abbildung 3-7: Tabellen, die Meta-Meta-Beziehungstypen repräsentieren und die Meta-Entitäten[343] von [CDIF94f] entsprechend dem EIA/CDIF-Meta-Meta-Modell miteinander in Beziehung setzen[344]

343 Vgl. hierzu Abbildung 3-6 auf Seite 119.

344 Hinweis: die Surrogatwerte in dieser Abbildung weisen *willkürlich* von 10 aufsteigende Werte auf.

3.3.2 Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf Object Rexx

In diesem Abschnitt wird eine Klassenhierarchie spezifiziert, die in der interpretierten, objektorientierten Sprache Object Rexx[345] abgefaßt ist und die Meta-Meta-Entitätstypen des EIA/CDIF-Meta-Meta-Modells repräsentiert.

Meta-Meta-Entitätstypen werden in Form von Klassen repräsentiert, die Meta-Meta-Attribute werden in Form von Attributmethoden für die entsprechende Klasse deklariert, womit das Abfragen und Setzen von Attributwerten standardmäßig für Instanzen der entsprechenden Klasse möglich wird. Im Gegensatz dazu werden die Meta-Meta-Beziehungstypen bei diesem Vorschlag im Sinne einer minimalen Spezifikation durch Object Rexx-Instanzen vom Typ *Relation*[346] direkt repräsentiert. Dies ist deshalb möglich, weil die Meta-Meta-

345 Object Rexx wurde im Rahmen dieser Arbeit aus verschiedensten Überlegungen ausgewählt, unter anderem, weil es sich dabei um eine innovative, objektorientierte und interpretierte sowie aufgrund ihrer Syntax relativ leicht zu erlernende Programmiersprache handelt, die darüber hinaus auf den verschiedensten Betriebssystemplattformen (AIX, Linux, OS/2, Windows NT, Windows 95) zur Verfügung steht. Eine kurze Charakterisierung der Programmiersprache findet sich im Anhang im Abschnitt 6.2.2.1, „Kurzeinführung in die Programmiersprache Object Rexx“ auf Seite 451ff unten. Informationen über Rexx und Object Rexx können auch über das Internet recherchiert werden, vgl. in diesem Zusammenhang auch [W3Rexx] respektive [W3ORexx].
In den Ausführungen im Hauptteil dieser Arbeit wird Object Rexx so eingesetzt, als würde es sich dabei um objektorientierten Pseudo-Code handeln, bei dem der Nachrichtenoperator durch eine Tilde („**~**“, auch als „Twiddle“ bezeichnet), kaskadierende Nachrichten durch eine Doppeltilde („**~~**“) repräsentiert werden. Nichtsdestoweniger können die dargestellten Programmanweisungen direkt unter einem Object Rexx-Interpreter ausgeführt werden, sodaß es sich hierbei eigentlich um ausführbare Spezifikationen handelt.
Jedes Object Rexx-Programm wird vom Interpreter vor der Ausführung zuerst zerlegt und auf Syntaxfehler hin überprüft, wobei Deklarationen für die Definition von Klassen, Methoden und Routinen, die mit einem doppelten Doppelpunkt („**::**“) eingeleitet werden, anschließend für das Programm zur Benutzung zur Verfügung stehen. Klassenobjekte werden hierbei direkt über die Kopplung mit Hilfe der Laufzeitumgebungsverzeichnisse adressierbar gemacht (Bezeichner, die mit einem Punkt beginnen, beziehen sich auf Objekte, die in der Laufzeitumgebung in einem Directory – also einem Verzeichnis mit eindeutigen Namen als Schlüssel – gespeichert sind; vgl. hierzu [Flat96c], in dem die Gültigkeitsregeln und die Verzeichnisse der Laufzeitumgebung systematisch über die offiziell zur Verfügung stehende Dokumentation von Object Rexx hinaus dargestellt sind).
Im Unterschied dazu werden im Anhang Object Rexx-Programme dokumentiert, die in der Implementierung selbst auf Klassen, Routinen und Methoden zurückgreifen, die in frei über das Internet beziehbaren Klassen- und Modulbibliotheken des Autors definiert sind. Vgl. hierzu z.B. [Flat96c], [Flat96d], [Flat96e], [Flat97b] und [Flat97c].

346 Die Object Rexx-Klasse *Relation* erlaubt das beliebig oftmalige Inbeziehungsetzen von zwei Objekten miteinander, wobei ein Objekt als Schlüssel (Index), das andere als damit assoziiert festgelegt wird. Ein Schlüssel-Objekt kann unterschiedlich viele Objekte referenzieren, ein assoziiertes Objekt mit beliebig vielen unterschiedlichen Schlüssel-Objekten in Beziehung stehen. Die Object Rexx-Implementierung erlaubt einerseits das

Fortsetzung folgt auf der nächsten Seite.

Beziehungstypen selbst über keine Meta-Meta-Attribute verfügen und somit lediglich die Möglichkeit vorgesehen werden muß, die Beziehungen zwischen den Meta-Meta-Entitätstypen nach Meta-Meta-Beziehungstypen getrennt zu dokumentieren und in jede Richtung hin navigierbar zu machen.

Wie auch weiter oben im Abschnitt 3.3.1, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf relationale Datenbankverwaltungssysteme“ auf Seite 108ff, diskutiert, werden für die Spezifikationen in Object Rexx die Abkürzungen für die Meta-Meta-Entitätstypen und Meta-Meta-Beziehungstypen benutzt.

Programmcode 3-1 stellt die Spezifikation des EIA/CDIF-Meta-Meta-Modells in Object Rexx dar:[347]

– Sämtliche Instanzen der Object Rexx-Klasse *„Relation“*, die jeweils einen Meta-Meta-Beziehungstyp repräsentieren, werden im Object Rexx Laufzeitverzeichnis *.Local* gespeichert und stehen daher für sämtliche Programme und Module des jeweiligen Prozesses zur Verfügung.[348]

– Für die Relationen gilt als Konvention, daß die Quell-Meta-Entitäten das Schlüssel-Objekt (Index) bilden und die Ziel-Meta-Entitäten damit assoziieren.[349]

Navigieren von Schlüssel-Objekten zu den zugeordneten Objekten, sowie von assoziierten Objekten zu sämtlichen auf sie verweisenden Schlüssel-Objekte.
Somit können mit Instanzen der Object Rexx Klasse *Relation* sämtliche für das EIA/CDIF-Meta-Meta-Modell benötigten Kardinalitäten repräsentiert werden, nämlich <0,N> zu <1,1>, <0,N> zu <1,N> und <0,N> zu <0,N>.

347 Ein Object Rexx-Programm muß mit einem Kommentar beginnen, der durch die Zeichenkette „**/***“ eingeleitet und durch „***/**“ abgeschlossen wird. Kommentare dürfen ineinander verschachtelt sein und sich über mehrere Zeilen erstrecken.

348 Beispielsweise kann nach dem Aufruf des Programmes im folgenden Programmcode 3-1 die Relation, die den Meta-Meta-Beziehungstyp „CollectableMetaObject.IsUsedIn.-SubjectArea“ repräsentiert, mit folgenden Varianten aus dem Umgebungsverzeichnis abgerufen und der Variable *„tmpIsUsedIn“* zugewiesen werden:

```
tmpIsUsedIn = .IsUsedIn             /* Variante 1 */
tmpIsUsedIn = .Local ~ IsUsedIn     /* Variante 2 */
```

349 Durch die Nutzung von Object Rexx-Relationen kann mit Hilfe der dafür definierten Methoden sehr einfach festgestellt werden, welche MetaEntitäten als Quelle mit welchen Ziel-MetaEntitäten assoziiert sind und umgekehrt, sowie getestet werden, ob eine bestimmte MetaEntität als Quelle oder als Ziel in einem bestimmten Meta-Meta-Beziehungstyp enthalten ist.

– Die definierten Klassen repräsentieren die entsprechenden Meta-Meta-Entitätstypen und stehen aufgrund des *PUBLIC*-Schlüsselwortes innerhalb des jeweiligen Prozesses sämtlichen Programmen und Modulen zur Verfügung.[350] Ähnlich wie in Smalltalk[351] oder vergleichbar mit den Ausführungen zu RM/T weiter oben, erfolgt die Instanziierung einer Object Rexx-Klasse so, daß entsprechend der Generalisierungshierarchie sämtliche Klassen am Weg zur Object Rexx-Wurzel *Object* instanziiert werden.[352, 353]

```
/* EIA/CDIF-Meta-Meta-Model: Spezifikation in Object Rexx */

/* Meta-Meta-Relationship:
   "CollectableMetaObject.IsUsedIn.SubjectArea" */
.Local ~ IsUsedIn                = .relation ~ new

/* Meta-Meta-Relationship:
   "MetaAttribute.IsLocalMetaAttributeOf.AttributableMetaObject" */
.Local ~ IsLocalMetaAttributeOf  = .relation ~ new

/* Meta-Meta-Relationship:
   "AttributableMetaObject.HasSubtype:AttributabelMetaObject"    */
.Local ~ HasSubtype              = .relation ~ new

/* Meta-Meta-Relationship:
   "MetaRelationship.HasSource.MetaEntity" */
.Local ~ HasSource               = .relation ~ new

/* Meta-Meta-Relationship:
   "MetaRelationship.HasDestination.MetaEntity" */
.Local ~ HasDestination          = .relation ~ new

/* Deklarationen                                  */
/* Deklaration für: "MetaObject"                        */
:: CLASS MO     PUBLIC
:: METHOD ALIASES               ATTRIBUTE
:: METHOD CDIFMetaIdentifier    ATTRIBUTE
```

350 Wird das Schlüsselwort *PUBLIC* bei der Deklaration von Klassen und Routinen weggelassen, beschränkt sich ihr Gültigkeitsbereich auf das Modul beziehungsweise Programm, in dem die Deklaration erfolgt.

351 Vgl. hierzu [GolRob83].

352 Im Anhang 6.2, „Implementierungsbeispiele“ auf Seite 423ff, wird diese direkte Entsprechung auch in der Implementierung in ORACLE und in Object Rexx ausgenutzt.

353 Im folgenden Programmcode werden die Definitionen für die Meta-Meta-Beziehungstypen und die Deklarationen der Object Rexx-Klassen gemeinsam mit den dafür definierten Attributmethoden durch einen grauen Hintergrund hervorgehoben.

```
:: METHOD Constraints           ATTRIBUTE
:: METHOD Description           ATTRIBUTE
:: METHOD Name                  ATTRIBUTE
:: METHOD Usage                 ATTRIBUTE

/* Deklaration für: "SubjectArea"               */
:: CLASS SA             SUBCLASS MO     PUBLIC
:: METHOD VersionNumber         ATTRIBUTE

/* Deklaration für: "CollectableMetaObject"     */
:: CLASS CMO            SUBCLASS MO     PUBLIC

/* Deklaration für: "MetaAttribute"             */
:: CLASS MA             SUBCLASS CMO    PUBLIC
:: METHOD DataType              ATTRIBUTE
:: METHOD Domain                ATTRIBUTE
:: METHOD IsOptional            ATTRIBUTE
:: METHOD Length                ATTRIBUTE

/* Deklaration für: "AttributableMetaObject"    */
:: CLASS AMO            SUBCLASS CMO    PUBLIC

/* Deklaration für: "MetaEntity"                */
:: CLASS ME             SUBCLASS AMO    PUBLIC
:: METHOD Type                  ATTRIBUTE

/* Deklaration für: "MetaRelationship"          */
:: CLASS MR             SUBCLASS AMO    PUBLIC
:: METHOD MinSourceCard         ATTRIBUTE
:: METHOD MaxSourceCard         ATTRIBUTE
:: METHOD MinDestCard           ATTRIBUTE
:: METHOD MaxDestCard           ATTRIBUTE
```

Programmcode 3-1: Spezifikation des EIA/CDIF-Meta-Meta-Modells in Object Rexx

Die gewählte Klassenhierarchie definiert die Wurzel des EIA/CDIF-Meta-Meta-Modells als normale Objektklasse, indem durch das Laufzeitsystem die Klasse „MetaObject“ als direkter Subtyp der Object Rexx Klasse „Object“ festgelegt wird. Somit können zwar Instanzen der Klassen gebildet werden, die EIA/CDIF-

Meta-Modelle repräsentieren können, allerdings sind diese Objekte bei der hier angeführten Spezifikation selbst nicht mehr instanziierbar.[354]

Eine beispielhafte Extension für die MetaObjekte könnte wie in Abbildung 3-6 auf Seite 119 auch für die Object Rexx Spezifikation dargestellt werden, wobei die Spalte „surR“ durch das entsprechende implizite Object Rexx Surrogat[355] ersetzt wird. Im Unterschied dazu kann zwar die Extension der Meta-Meta-Beziehungstypen wie in Abbildung 3-7 auf Seite 123 konzeptionell dargestellt werden, allerdings würden für die einzelnen Beziehungen zwischen MetaEntitäten hier keine eigenständigen Object Rexx Objekte notwendig sein, stattdessen werden die entsprechenden Methoden der Object Rexx Klasse „Relation“ dafür genutzt.[356]

354 Wenn im Unterschied dazu die Deklaration für die Klasse „MetaObject“ die Object Rexx-Klasse „Class“ als Supertyp anführt, dann erhält man dadurch Metaklassen, deren Instanzen selbst instanziierbar wären.

355 Es handelt sich hierbei um ein Surrogat, das sicherstellt, daß jedes Object Rexx-Objekt in Sammelobjekten eindeutig referenzierbar ist.

356 Die Implementierung der einzelnen Beziehungen zwischen Indexobjekten zu assoziierten Objekten entzieht sich dem direkten Zugriff der Programmierer, sodaß die entsprechenden Surrogate beziehungsweise Hash-Werte der Objekt-Nachricht „==“ nicht feststellbar sind. Die Navigation beziehungsweise der Zugriff auf Meta-Entitäten erfolgt unter anderem über die Nachrichten dieser Klasse wie zum Beispiel mit „AT“, „ALLAT“, „INDEX“ und „ALLINDEX“. Die Enumeration über sämtliche Beziehungen, die in einem Object Rexx-Objekt vom Typ *„Relation“* verwaltet werden, erfolgt mit Hilfe eines Objektes vom Typ *„Supplier“*, das als Resultat der Nachricht „SUPPLIER“ an eine Instanz vom Typ *„Relation“* zurückgegeben wird.

3.4 Zusammenfassende Diskussion des EIA/CDIF-Meta-Meta-Modells

Das EIA/CDIF-Meta-Meta-Modell wurde über einen Zeitraum von sieben Jahren entwickelt, von 1987 bis 1994. Es legt die Konzepte fest, mit deren Hilfe Meta-Modelle definiert werden können, deren Instanziierungen wiederum für die Dokumentation und den Austausch von Modelldaten, entsprechend den Konzepten der Meta-Modelle, benutzt werden können. Das EIA/CDIF-Meta-Meta-Modell repräsentiert ein minimales erweitertes Entity-Relationship-Attribut-Modell.

3.4.1 Allgemeine Diskussion

In diesem Abschnitt werden Überlegungen zu verschiedenen, von einer konkreten Sichtweise auf das Integrierte EIA/CDIF-Meta-Modell unabhängigen, Konzepten des EIA/CDIF-Meta-Meta-Modells angestellt und in weiterer Folge prägnant diskutiert.

3.4.1.1 Konzeptionelle Modellierung

Im EIA/CDIF-Meta-Meta-Modell finden sich die grundlegenden Konstrukte der konzeptionellen Modellierung wieder, und zwar in der Form von erweiterten Entity-Relationship-Modellen.[357] Diese Variante der Entity-Relationship-Attribut-Modellierung wird mittlerweile auch in den entsprechenden Lehrbüchern[358] weltweit dargestellt und damit vermittelt. Der Einfluß dieser Methode der konzeptionellen (Daten-)Modellierung ist so groß geworden, daß auch internationale Standardisierungsorganisationen wie ISO/IEC sie zu standardisieren versuchen.[359]

---

357 In diesem Zusammenhang sind dies die Konzepte „Entitätstypen“ und „Beziehungstypen“, die über Attribute verfügen dürfen, sowie von Generalisierungsbeziehungen, mit deren Hilfe hierarchische Über- beziehungsweise Unterordnung ausgedrückt werden kann.

358 Vgl. zum Beispiel [ElmNav89], [BaCeNa92] oder auch [FerSin98].

359 Vgl. hierzu den Entwurf (Status: „CD“, Abkürzung für englisch: „Committee Draft“, deutsch: Komitee-Entwurf) in [ISO97a].

Das EIA/CDIF-Meta-Meta-Modell legt fundamental die Ausdrucks- beziehungsweise Strukturierungsmöglichkeiten für Meta-Modelle fest. Gleichzeitig gibt es damit das Regelwerk vor, mit dessen Hilfe Meta-Modelle konstruiert werden dürfen.

3.4.1.2 Reflexivität des EIA/CDIF-Meta-Meta-Modells

Das EIA/CDIF-Meta-Meta-Modell kann durch Instanziierung der Meta-Meta-Entitätstypen und Meta-Meta-Beziehungstypen das EIA/CDIF-Meta-Meta-Modell selbst abbilden und ist damit ein reflexives System, ähnlich der menschlichen Sprache,[360] mir deren Hilfe man die Sprache selbst beschreibt.

3.4.1.3 Aggregation von MetaAttributen zu AttribuierbarenMetaObjekten

Auf Ebene des EIA/CDIF-Meta-Meta-Modells können die Meta-Meta-Entitätstypen durch Meta-Meta-Attribute beschrieben werden. Für die Definition von EIA/CDIF-Meta-Modellen werden auf Meta-Meta-Modellebene die Meta-Attribute als Instanzen des Meta-Meta-Entitätstyps „MetaAttribute“ definiert und anschließend über den Meta-Meta-Beziehungstyp „IsLocalMetaAttributeOf“ Instanzen des Meta-Meta-Entitätstyps „AttributableMetaObject“ zugeordnet.

In den EIA/CDIF-Standards erfolgt die Darstellung der Definition von EIA/CDIF-Meta-Modellen jedoch so, daß MetaAttribute zu den AttribuierbarenMetaObjekten hin in Form von lokalen Attributen aggregiert werden. Auf Meta-Modellebene existieren daher MetaAttribute nie für sich allein.[361, 362]

Das EIA/CDIF-Meta-Meta-Modell selbst erlaubt es nicht, ausdrücklich Meta-Meta-Beziehungstypen als aggregierend zu definieren. Somit kann nur auf-

---

360 Vgl. in diesem Zusammenhang auch die Diskussion in [Henn80], Seite 779, 2. Spalte oben, wo diese Reflexivität der Sprache als Voraussetzung dafür angesehen wird, daß es einsprachige Lexika überhaupt geben kann.

361 Vgl. in diesem Zusammenhang beispielsweise die Ausführungen über die Aggregationsabstraktion in [BaCeNa92], Seite 17.

362 Aufgrund der Kardinalitäten zwischen den Meta-Meta-Entitätstypen „MetaAttribute“ und „AttributableMetaObject“ über den Meta-Meta-Beziehungstyp „IsLocalMetaAttributeOf“ ist klar, daß Instanzen vom Typ „MetaAttribute“ voll im Beziehungstyp partizipieren (Untergrenze: „1“) und aufgrund der Obergrenze von „1“ in den Meta-Meta-Entitätstyp „AttributableMetaObject“ aggregiert werden müssen. Vgl. hierzu auch Abbildung 3-1 auf Seite 53.

grund der Darstellung der EIA/CDIF-Meta-Modelle selbst festgestellt werden, ob Meta-Entitätstypen, die mit der Kardinalität (Minimum: „1“, Maximum: „1“)[363] über (binäre) Meta-Meta-Beziehungstypen festgelegt sind, aggregiert werden oder nicht. Beispielsweise könnten die Instanzen des Meta-Meta-Entitätstyps „MetaRelationship“ zu den Instanzen von „MetaEntity“ in Form von Attributen aggregiert werden, was aber aufgrund des Studiums der standardisierten EIA/CDIF-Meta-Modelle nicht geschieht.[364, 365, 366]

3.4.1.4 Vererbungsregeln

Über den Meta-Meta-Beziehungstyp „0:N AttributableMetaObject.*HasSubtype*.-AttributableMetaObject 0:N“ kann für AttribuierbareMetaObjekte eine Generalisierungs- beziehungsweise Spezialisierungshierarchie[367] gebildet werden. Hierbei gelten folgende Regeln:

---

363 Es läßt sich feststellen, daß die Aggregierung vergleichbar wie bei den Überleitungsregeln von Entity-Relationship-Diagrammen zu relationalen Tabellen (vgl. hierzu zum Beispiel den weiter oben angeführten [BaCeNa92], aber auch [Flat96b]) erfolgen kann, indem sämtliche Attribute jenes Entitätstyps, der als Maximum den Wert „1“ aufweist in den anderen Entitätstyp beziehungsweise in die diesen Entitätstyp repräsentierende Tabelle aufgenommen wird.

364 Es ist durchaus nicht unüblich, daß Beziehungen in Entitäten dadurch dargestellt werden, daß sie zu Attributen aggregiert werden. Beispielsweise repräsentieren Fremdschlüsselspalten in relationalen Tabellen Beziehungen zu bestimmten Tupeln in derselben oder in anderen Tabellen. Gleichermaßen kennt die Datenmodellierungssprache EXPRESS von STEP keine expliziten Beziehungstypen, stattdessen werden Beziehungen in Form von Attributen in Entitätstypdefinitionen angegeben. (Die entsprechenden ISO/IETC-Standards sind im Abschnitt 1.3.2.3, „EIA/CDIF und STEP/EXPRESS (ISO/IEC TC 184/SC 4)“ auf Seite 21ff, angeführt.)

365 MetaBeziehungen bleiben daher in Meta-Modellen als „first class objects“ (abgekürzt: „FCO“) bestehen und weisen zum einen dementsprechende Benennungsregeln auf, die die Eineindeutigkeit gewährleisten sollen, zum anderen das eineindeutige Surrogat „CDIFMetaIdentifier“ (wäre in diesem Bild der FCO der entsprechende „object identifier“). In weiterer Folge können daher MetaBeziehungen mit Hilfe von MetaAttributen attribuiert, sowie mit Hilfe des Meta-Meta-Beziehungstyps „0:N AttributableMetaObject.*Has-Subtype*.AttributableMetaObject 0:N“ entsprechend spezialisiert werden.

366 Dieselben Überlegungen können auch für das EIA/CDIF-Meta-Meta-Modell angestellt werden, da es die weiter oben angeführte Eigenschaft der Reflexivität besitzt und somit auf sich selbst anwendbar ist.

367 In dieser Arbeit wird eine „Spezialisierung“ dadurch definiert, daß im Meta-Meta-Beziehungstyp „AttributableMetaObject.HasSubtype.AttributableMetaObject“ das Quell-AttribuierbareMetaObjekt jenes ist, das durch das Ziel-AttribuierbareMetaObjekt spezialisiert wird. Eine „Generalisierung“ wird dadurch festgelegt, indem im Meta-Meta-Beziehungstyp „AttributableMetaObject.HasSubtype.AttributableMetaObject“ das Ziel-AttribuierbareMetaObjekt durch das Quell-AttribuierbareMetaObjekt generalisiert wird. Insoferne stellt die Generalisierung den inversen Fall der Spezialisierung dar und umgekehrt.

– das fundamentale AttribuierbareMetaObjekt „RootObject“ tritt in genau zwei Instanzen vom Meta-Meta-Beziehungstyp „0:N AttributableMeta-Object.*HasSubtype*.AttributableMetaObject 0:N“ auf, nämlich als Quell-AttribuierbaresMetaObjekt für die MetaEntitäten „RootEntity“ und einmal für „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“,[368]

– mit Ausnahme der direkten Subtypen von „RootObject“, das als (einzige) Instanz des Meta-Meta-Entitätstyps „AttributableMetaObject“ auftritt, darf in der Spezialisierung von „RootEntity“ und von „0:N RootEntity.IsRelatedTo.RootEntity 0:N“ der Meta-Typ[369] nicht mehr verändert werden.[370]

– Ein Subtyp erbt sämtliche MetaAttribute aller übergeordneten Supertypen. Der Name eines ererbten MetaAttributs muß eindeutig sein, das heißt, er darf weder bereits für ein lokales MetaAttribut vergeben worden sein, noch über die Generalisierungshierarchie ererbt werden.[371]

Der EIA/CDIF-Standard, wie er in [CDIF94b] definiert ist, läßt auch die Mehrfachvererbung zu. Eine Mehrfachvererbung ist dann gegeben, wenn über den Meta-Meta-Beziehungstyp „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“ ein untergeordnetes AttribuierbaresMetaObjekt mehr als ein direkt übergeordnetes AttributierbaresMetaObjekt besitzt. In einem solchen Fall ist eine bestimmte Reihenfolge in den direkten Supertypen undefiniert.[372]

---

368 Vgl. hierzu die Ausführungen zum standardisierten EIA/CDIF-Meta-Modell „Foundation“ im Abschnitt 4.1.1, Seite 155ff weiter unten.

369 Vgl. die Ausführungen zu dem Begriff „Meta-Typ“ weiter oben in Fußnote 122 auf Seite 31.

370 Theoretisch wäre es denkbar, daß Subtypen von „RootEntity“ oder „RootEntity.*IsRelatedTo*.RootEntity“ beliebige Meta-Typen aufweisen könnten.

371 Vgl. unter anderem auch die Erläuterungen im Meta-Meta-Attribut „Constraints“ der Meta-Meta-Entität „MetaAttribute“ in [CDIF94b], Seite 44.

372 Für Implementierungen zu Dokumentationszwecken von EIA/CDIF-Meta-Modellen empfiehlt es sich aber, für die Aufbereitung von Ausdrucken, eine bestimmte Reihenfolge vorzusehen. Damit kann sichergestellt werden, daß die Dokumentation die Reihenfolge für das Aufsuchen von Supertypen konsistent gleich wählt. Beispielsweise könnten die Supertypen für diesen Zweck alphabetisch sortiert werden. Nachdem keine bestimmte Reihenfolge vom EIA/CDIF-Standard vorgegeben ist, kann dementsprechend jede beliebige Reihenfolge benutzt werden, auch eine, die von Menschen als alphabetisch sortiert angesehen werden kann.
Die vom Autor erstellten Implementierungen in Object Rexx für das Aufbereiten von EIA/CDIF-Meta-Modelldefinitionen für das World-Wide-Web in Form von HTML-Versionen, benutzt diese Möglichkeit. Die HTML-Versionen der standardisierten EIA/CDIF-Meta-Modelle wurden EIA/CDIF im Herbst 1997 für deren Mitglieder zur Verfügung gestellt, die über das WWW darauf zugreifen können. Leider können aus lizenzrechtlichen Gründen diese Versionen vom Autor nicht der interessierten Öffentlichkeit zur

Fortsetzung folgt auf der nächsten Seite.

3.4.1.5 Typisierung von MetaBeziehungen

Sowohl die Definition von Meta-Meta-Beziehungstypen als auch die Festlegung des Meta-Meta-Entitätstyps „MetaRelationship“ sehen keine Möglichkeit vor, die Beziehungstypen selbst zu typisieren und sie damit nach weiteren Kriterien voneinander unterscheidbar zu machen. Beispielsweise ist es nicht möglich, jene Beziehungstypen ausdrücklich zu kennzeichnen, die identitätsstiftend für assoziative beziehungsweise charakterisierende Entitätstypen sind.[373] In weiterer Folge ist es auch nicht möglich, jene Beziehungstypen zu kennzeichnen, die aggregierend im Sinne von beispielsweise [BaCeNa92][374] sind.[375]

Somit wird sowohl für das EIA/CDIF-Meta-Meta-Modell selbst als auch für EIA/CDIF-Meta-Modelle vorgeschlagen, Beziehungstypen mit Hilfe eines optionalen Meta-(Meta-)Attributs[376] vom EIA/CDIF-Datentyp „Enumerated“ mit folgender Wertemenge zu definieren: „Associate“[377] (deutsch: „in Beziehung setzen“), „Create“[378] (deutsch: „begründend“, „identitätsstiftend“) und „Aggregate“ (deutsch: „aggregierend“). Für EIA/CDIF-Meta-Modelle, die für CDIF-Austäusche herangezogen werden, könnte eine derartige Typisierung einfach dadurch erreicht werden, indem der Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.-

---

Verfügung gestellt werden. Dies deshalb, weil die Texte aus elektronischen Versionen der EIA/CDIF-Standards direkt mit Hilfe von selbsterstellten Object Rexx-Programmen extrahiert wurden und somit die Rechte an den dargestellten Daten bei EIA/CDIF liegen.

373 Sowohl assoziative als auch charakterisierende Entitätstypen sind für sich allein nicht existenzfähig und benötigen daher eine Beziehung zu identitätsstiftenden weiteren Entitätstypen. Somit muß in mindestens einem Primärschlüsselkandidaten mindestens ein Fremdschlüssel enthalten sein. Vgl. hierzu zum Beispiel die Ausführungen in [Flat90b], Abschnitt 3.5.4.1.2, Seite 74ff.

In EIA/CDIF-Meta-Modellen sind existenzabhängige Meta-Entitätstypen immer an der Pfeilspitze zu finden und weisen im Minimum der Quellkardinalität einen Wert auf, der größer Null ist.

374 Vgl. ebenda die Ausführungen von Seite 19 bis 23 über die Aggregationsabstraktion.

375 Im EIA/CDIF-Meta-Meta-Modell aggregiert beispielsweise der Meta-Meta-Beziehungstyp „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMetaObject 1:1“ Meta-Attribute zu den entsprechenden AttribuierbarenMetaObjekte.

In EIA/CDIF-Meta-Modellen wären jene Meta-Entitätstypen, die als Aggregat andere Meta-Entitätstypen aufweisen, immer an der Pfeilspitze derartiger Meta-Beziehungstypen zu finden. Im EIA/CDIF-Meta-Meta-Modell ist das Aggregat daher der Meta-Meta-Entitätstyp „AttributableMetaObject“.

376 Ein sinnvoller Name für dieses Meta-Attribut wäre „Type“.

377 Diese Ausprägung bezeichnet einen „normalen“ Beziehungstyp, das heißt einen Beziehungstyp, der nur eine Beziehung zwischen zwei Entitätstypen herstellt, wobei über diese Tatsache hinaus keine weitere Bedeutung damit verbunden ist.

378 Im Sinne von „identitätsstiftend“, sodaß eventuell die Bezeichnung „Identifying“ angebrachter erscheinen mag.

RootEntity 0:N“ des fundamentalen Meta-Modells[379] mit Hilfe des vorgesehenen EIA/CDIF-Erweiterungsmechanismus das angesprochene Meta-Attribut erhält.

Eine weitere Möglichkeit, diese Semantik ausdrücklich zu dokumentieren, wird in dieser Arbeit in Form des Meta-Modells „M2Level“ im Abschnitt 4.4, „Definition eines EIA/CDIF-konformen Meta-Modells „M2Level““ auf Seite 356ff, gegeben. Es hat unter anderem den Vorteil, daß das EIA/CDIF-Meta-Meta-Modell dafür nicht geändert werden muß.

3.4.1.6 Einander ausschließende MetaBeziehungen

EIA/CDIF[380] sieht vor, daß für die Erstellung von EIA/CDIF-Meta-Modellen das graphische Darstellungsmittel eines Bogens (englisch: „arc“) zur Verfügung steht. Mit Hilfe dieses Bogens ist es möglich, einander ausschließende Beziehungstypen[381], die von einem Entitätstyp ausgehen oder zu einem hinführen, zu kennzeichnen.[382] Damit wird ausgedrückt, daß eine Entität eines Entitätstyps nur an einer einzigen Instanz der mit dem Bogen gekennzeichneten Beziehungstypen teilhaben darf.[383] Die Minimumwerte der entsprechenden Kardinalitäten bestimmen, ob eine der alternativen Beziehungstypen instanziiert werden muß (Wert größer Null) oder kann (Wert gleich Null).

Für diese bedingten Beziehungstypen stellt sich allerdings das Problem, daß sie ausschließlich in graphischen Darstellungen zur Verfügung stehen. Im EIA/CDIF-Meta-Meta-Modell steht ansonsten keine Möglichkeit zur Verfügung, einander ausschließende Beziehungstypen ausdrücklich zu dokumentieren.[384]

---

379 Vgl. hierzu auch den Abschnitt 4.1.1.1.3 auf Seite 164 unten.

380 Vgl. hierzu auch den Abschnitt „Mutual Exclusivity of Meta-relationships“ in [CDIF94a], Seite 17ff.

381 „Einander ausschließende Beziehungstypen“ werden manchmal auch als „bedingte Beziehungstypen“ bezeichnet.

382 Derartige Einschränkungsmöglichkeiten sind durchaus üblich und finden sich auch in kommerziellen CASE-Systemen wieder, siehe zum Beispiel die Beschreibung der Entity-Relationship-Modellierung in [Bark90b] für ORACLEs-CASE*Method, wie sie auch noch im 1998 aktuellen Designer/2000 Produkt der Firma zur Verfügung steht.

383 Sofern der Maximumwert der Kardinalität des betroffenen Quell-Entitätstyps einen Wert höher als eins aufweist, bedeutet dies, daß eine Entität dieses Typs öfters als einmal in einer der vom Bogen berührten Beziehungstypen enthalten sein kann. Vgl. dazu [CDIF94a] auf Seite 18 oben.

384 Der technische Leiter des EIA/CDIF-Komitees, Dr. Johannes Ernst, schlägt in seinem Arbeitspapier [CDIF97d] vor, dieses Problem dadurch zu lösen, daß in das EIA/CDIF-Meta-Meta-Modell eine Meta-Meta-Entität „MetaRole“ aufgenommen wird, die über eine rekursive Meta-Meta-Beziehung „MetaRole.IsMutuallyExclusiveWith.MetaRole“ die ein-

Fortsetzung folgt auf der nächsten Seite.

Somit empfiehlt es sich, dieses Konstrukt für die Definition von EIA/CDIF-Meta-Modellen nach Möglichkeit nicht einzusetzen.[385]

Eine Möglichkeit, einander ausschließende Beziehungstypen ausdrücklich zu dokumentieren, wird in dieser Arbeit in Form des Meta-Modells „M2Level“ im Abschnitt 4.4, „Definition eines EIA/CDIF-konformen Meta-Modells „M2Level““ auf Seite 356ff, gegeben. Es hat unter anderem den Vorteil, daß das EIA/CDIF-Meta-Meta-Modell dafür nicht geändert werden muß.

3.4.1.7 Überlappende Intensions-/Extensionspaarungen

Implementierungen des EIA/CDIF-Meta-Meta-Modells selbst, die entsprechend diesen Regeln erfolgen, können zunächst theoretisch sämtliche Definitionen für Meta-Modelle als Instanzen der entsprechenden Meta-Meta-Entitätstypen und Meta-Meta-Beziehungstypen repräsentieren:

- Im Falle einer Implementierung der Schnittstellen entsprechend dem EIA/CDIF-Standard, wie er in [CDIF97b] festgelegt und im Abschnitt „Verteilung des EIA/CDIF-Meta-Meta-Modells mit Hilfe von CORBA“ auf Seite 98ff beschrieben ist, wird ein Nur-Lese-Zugriff auf die Attribute der Objekte eingeräumt, die die Konzepte des EIA/CDIF-Meta-Meta-Modells selbst sowie das entsprechende EIA/CDIF-Meta-Modell repräsentieren.
  Instanzen des EIA/CDIF-Meta-Modells stellen dementsprechend die Modelldaten dar, deren Attribute über den im Standard vorgesehenen Schreib-/Lesezugriff auf die MetaAttribute in ihren Werten verändert werden können.
- Im Falle einer Implementierung mit Hilfe einer relationalen Datenbank[386], wie dies in dieser Arbeit durch die Spezifikationen in SQL 92 im Abschnitt,

---

ander ausschließenden MetaBeziehungen zu dokumentieren erlaubt. Entitäten vom Typ *MetaRole* werden über die neu zu definierende Meta-Meta-Beziehung „MetaType.Plays.-MetaRole“ den MetaEntitäten zugeordnet („MetaEntity“ wird in diesem Vorschlag in „MetaType“ umbenannt).
In diesem Zusammenhang sind auch die Festlegungen für das EIA/CDIF-Meta-Modell für den GegenstandsBereich „Datenmodellierung“ interessant, die im Abschnitt 4.1.3.2, „DMOD – EIA/CDIF-Meta-Modell für die Datenmodellierung“ auf Seite 241ff, übersichtlich dargestellt sind, insbesondere die Definitionen für die Konzepte „Rolle“, „RollenSpieler“ und „RollenBeschränkung“.

385 Von den bis Anfang Jänner 1998 standardisierten EIA/CDIF-Meta-Modellen setzt nur der GegenstandsBereich „Datenmodellierung“ einander ausschließende MetaBeziehungen ein.

3.3.1, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf relationale Datenbankverwaltungssysteme“ auf Seite 108ff, vorgesehen wird, können die Meta-Modelle in relationalen Tabellen gespeichert werden. Unter dem Gesichtspunkt der Definition von Tabellen, die die damit gespeicherten Strukturen der Meta-Modellkonzepte selbst in Form von SQL-Tabellendefinitionen realisieren, können die das EIA/CDIF-Meta-Meta-Modell repräsentierenden Tabellen als Repository für die entsprechend darin gespeicherten EIA/CDIF-Meta-Modelle angesehen werden.
Sofern durch entsprechende Programmunterstützung in weiterer Folge derartige Tabellen generiert werden, können auch die Modelldaten selbst in SQL-Tabellen abgelegt werden.[387]

– Im Falle einer Implementierung in Object Rexx[388], wie dies in dieser Arbeit durch die Spezifikation der Klassenhierarchie einerseits und der Definition der Object Rexx Objekte vom Typ *Relation* andererseits in Abschnitt 3.3.2, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf Object Rexx“ auf Seite 124ff, erfolgt, können die Konzepte von Meta-Modellen als Objekte repräsentiert werden. Dies erfolgt mit Hilfe der Instanziierung der entsprechenden Klassen, sowie mit dem Inbeziehungsetzen der Objekte miteinander mit Hilfe der Relationsobjekte, die die Meta-Meta-Beziehungen repräsentieren.
Für den Austausch von Modelldaten mag es wünschenswert erscheinen, daß die in Objekten repräsentierten Konzepte der entsprechenden Meta-Modelle selbst instanziierbar sind, das heißt, Metaklassencharakter aufweisen müssen. Dies kann dadurch gewährleistet werden, indem die Klassendefinitionen für die Wurzel des EIA/CDIF-Meta-Meta-Modells,

386 Vgl. hierzu Abschnitt 6.2.1, „Implementierung des EIA/CDIF-Meta-Meta-Modells in ORACLE“ auf Seite 424ff.

387 Tabellen, die die Strukturinformationen über instanziierbare Modelldaten beinhalten, können entsprechend selbst wiederum als Repository angesehen werden, indem diese Informationen dazu benutzt werden, um Tabellen zu definieren, die das Modell selbst darstellen. Ausprägungen beziehungsweise Instanzen der Modellkonzepte könnten in diesen entsprechenden Tabellen gespeichert sein.

388 Vgl. hierzu Abschnitt 6.2.2, „Implementierung des EIA/CDIF-Meta-Meta-Modells in Object Rexx“ auf Seite 449ff.

„MetaObject“, als Subklasse der Object Rexx Metaklasse „Class“ definiert wird.[389]

Damit lassen sich die bereits eingangs in Abschnitt 2.2.1 auf Seite 31 diskutierten, überlappenden Intensions-/Extensions-Paare in diesen unterschiedlichen Schnittstellen-, Datenbank- und objektorientierten Programmdefinitionen wiederfinden:

- Das EIA/CDIF-Meta-Meta-Modell („M3“) stellt die Intension für EIA/CDIF-Meta-Modelle dar. EIA/CDIF-Meta-Modelle („M2“) sind daher die Extensionen des EIA/CDIF-Meta-Meta-Modells und können daraufhin überprüft werden, ob sie den darin vorgegebenen Strukturen entsprechen.
- Ein beliebiges EIA/CDIF-konformes, das heißt dem EIA/CDIF-Meta-Meta-Modell folgenden, Meta-Modell („M2“) stellt selbst die Intension für EIA/CDIF-Modelldaten dar. Diese EIA/CDIF-Modelldaten („M1“) sind daher Extensionen des Meta-Modells und können daraufhin überprüft werden, ob sie den darin vorgegebenen Strukturen entsprechen.[390]

---

389 Daraus folgt auch, daß die Objekte, die die einzelnen Konzepte der Modelldaten repräsentieren, selbst Metaklassenobjekte sind.
Ist dies nicht gewünscht, so genügt es, in der Repräsentation des fundamentalen EIA/CDIF-Meta-Modells „Foundation“, den Entitätstyp „RootObject“ als Subtyp von der Object Rexx-Klasse *„Object“* zu definieren, sowie mit der Object Rexx-Metaklassen-Anweisung „METACLASS“ in der Deklaration der Klasse auf die Metaklasse „AttributableMetaObject“ bezug zu nehmen. Dadurch wird vom Object Rexx-Laufzeitsystem aus die für die „RootObject“ repräsentierende Klasse eine Instanz der Metaklasse „AttributableMetaObject“ gebildet. Entsprechend müßten dann für die Klasse „RootEntity“ die Metaklasse „MetaEntity“ und für die Klasse „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ die Metaklasse „MetaRelationship“ festgelegt werden. Damit legt das Object Rexx-Laufzeitsystem ein Klassenobjekt vom Typ „MetaEntity“ für die Klasse „RootEntity“ und für jede einzelne ihrer Subklassen an, respektive ein Klassenobjekt vom Typ „MetaRelationship“ für die Klasse „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ und für jede einzelne ihrer Subklassen.
Dieselben Überlegungen können entsprechend auch für die Objekte angestellt werden, die die Klassen, die die Modelldaten beziehungsweise die damit definierten Konzepte repräsentieren. Mit anderen Worten könnte es sinnvoll sein, Klassen, die das EIA/CDIF-Meta-Meta-Modell, EIA/CDIF-Meta-Modelle und Modelldaten repräsentieren, als Metaklassen zu definieren.

390 Daten, die Modelle repräsentieren, können selbst wiederum als Intension für Ausprägungen aufgefaßt werden, die die Extensionen der Modelle darstellen. Dadurch könnte eine weitere Intensions-/Extensions-Paarung festgelegt werden, die allerdings auf dieser Ebene ihr Ende findet. Mit anderen Worten, die Extensionen selbst können nicht mehr als Intensionen interpretiert werden und stellen somit die „atomaren“, das heißt nicht mehr weiter zerlegbaren Einheiten in diesem Verfeinerungsvorgang dar.
EIA/CDIF setzt sich wie eingangs diskutiert mit dem Austausch von Modelldaten auseinander und bezeichnet die atomaren Daten als „M0“.

3.4.1.8 Das Meta-Meta-Attribut „CDIFMetaIdentifier“

Im EIA/CDIF-Meta-Meta-Modell wird für das Wurzelobjekt, dem Meta-Meta-Entitätstyp „MetaObject“, ein Meta-Meta-Attribut definiert, das als Surrogat dient. Es weist folgende Eigenschaften auf: [391]

- der Wert muß eineindeutig über alle MetaObjekte hinweg sein,[392]
- der Wert für ein bestimmtes MetaObjekt darf nie mehr geändert werden[393] und
- für den Fall, daß ein MetaObjekt auf Dauer aus dem Standard entfernt wird, darf der Wert nicht mehr wiederverwendet werden.

Verfügen zwei verschiedene Referenzen auf MetaObjekte in einer Implementierung über denselben Wert im Meta-Meta-Attribut „CDIFMetaIdenfier“, dann verweisen beide zumindest[394] auf dieselbe Repräsentation desselben CDIF MetaObjekts.[395]

---

391 Vgl. beispielsweise die Ausführungen zu Surrogaten z.B. in [Codd79], [Date85], [Date86], [ElmNav89], [BaCeNa92]. Auch in objektorientierten Implementierungen wird üblicherweise ein (oftmals implizites) Surrogatattribut für den Zweck der eindeutigen Identifizierung vorgesehen, das als „Object Identification“ (auch: „Object ID“) bekannt ist. Beispielsweise kann In Object Rexx der Surrogatwert eines beliebigen Objekts mit der Nachricht „==“ abgefragt werden.

392 Damit wird die Identifikationseigenschaft konstituiert.

393 Somit kann ein MetaObjekt immer nur durch einen einzigen auf Dauer festgelegten Wert identifiziert werden.

394 In einem verteilten System ist der Fall denkbar, daß die Implementierungen selbst voneinander unabhängig sind, wenngleich es sich um Implementierungen desselben MetaObjektes handeln kann. Vgl. hierzu beispielsweise auch die Ausführungen zum EIA/CDIF OMG/IDL-Standard, der weiter oben im Abschnitt 3.2, „Verteilung des EIA/CDIF-Meta-Meta-Modells mit Hilfe von CORBA“ auf Seite 98ff, vorgestellt wird, wodurch auch die EIA/CDIF-Meta-Modelle selbst grundsätzlich verteilt sein können. Die Schnittstelle „`is_identical()`“ (vgl. Punkt [4] in Abbildung 3-4 auf Seite 104) liefert den Wahrheitswert *True* dann, wenn der Wert im Meta-Meta-Attribut „CDIFMetaIdentifier“ mit dem des Argumentes übereinstimmt, *False* sonst.

395 Wie weiter unten im Abschnitt 4.1.3.2.3.2 auf Seite 251 erläutert wird, konnte im Rahmen dieser Arbeit aufgrund einer systematischen, rechnerunterstützten Auswertung sämtlicher zum Zeitpunkt 1. Jänner 1998 standardisierten EIA/CDIF-Meta-Modelle festgestellt werden, daß die Surrogateigenschaften nicht über alle EIA/CDIF-Meta-Modelle hinweg eindeutig gelten.
Diese Fehler wurden umgehend dem EIA/CDIF-Komitee mitgeteilt und werden in zukünftigen Überarbeitungen der standardisierten EIA/CDIF-Meta-Modelle korrigiert.

3.4.1.9 Meta-Meta-Attribute „SubtypeOf“ und „SupertypeOf“

Die Meta-Meta-Attribute „SubtypeOf“ und „SupertypeOf“ werden selbst nicht ausdrücklich im EIA/CDIF-Meta-Meta-Modell als Meta-Meta-Attribute definiert![396] Von diesen beiden Meta-Meta-Attributen wird in den EIA/CDIF-Standards nur das Meta-Meta-Attribut „SubtypeOf“ benötigt, um die Meta-Meta-Beziehungen vom Typ „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“ zu instanziieren.

In weiterer Folge können dann die Werte für „SupertypeOf“ aus der Auswertung der Meta-Meta-Beziehungen vom Typ „0:N AttributableMetaObject.-*HasSubtype*.AttributableMetaObject 0:N“ jederzeit abgeleitet werden. Damit entfällt die Notwendigkeit, dieses Meta-Meta-Attribut zu führen, wenngleich es für Dokumentationszwecke manchmal interessant sein mag.[397, 398]

3.4.2 Sichtweisenbezogene Diskussion

In diesem Abschnitt werden Überlegungen zu unterschiedlichen, dargestellten Konzepten angestellt und in weiterer Folge diskutiert, die direkt oder indirekt mit der Partitionierung der MetaObjekt-Definitionen zusammenhängen. Aus diesem Grunde wird intensiver als im vorigen Abschnitt zur Veranschaulichung auf EIA/CDIF-Meta-Modelle zurückgegriffen, die ja in einem instanziierten EIA/-CDIF-Meta-Meta-Modell repräsentiert sind.

---

396 Vgl. hierzu die Ausführungen im EIA/CDIF-Standard „Framework for Modeling and Extensibility“ [CDIF94b], beispielsweise die Definitionen zu „MetaObject“ (Seite 49) oder „AttributableMetaObject“ (Seite 41).

397 Durch diese Ableitungsmöglichkeit können zudem auch die ererbten MetaAttribute und MetaBeziehungen für jeden Subtyp gefunden werden.

398 Hinzu kommt, daß diese Werte für das Meta-Meta-Attribute „SupertypeOf“ in den gegenwärtig standardisierten, EIA/CDIF-konformen Meta-Modellen nur im Hinblick auf das entsprechende Meta-Modell selbst auf Vollständigkeit überprüfbar wäre. Letztendlich erscheint das Führen eines solchen Meta-Meta-Attributs nur für das vom EIA/CDIF-Komitee hypothetisch vorgesehene Integrierte EIA/CDIF-Meta-Modell sinnvoll zu sein, da nur es über sämtliche Konzepte aller standardisierten EIA/CDIF-Meta-Modelle verfügen kann. Für diese Arbeit wurde dieses hypothetische Meta-Modell konkret erzeugt und wird im Abschnitt 4.2, “Das „Integrierte EIA/CDIF-Meta-Modell““ auf Seite 267ff, vorgestellt, dokumentiert und diskutiert.

3.4.2.1 Das „Integrierte EIA/CDIF-Meta-Modell“ und seine Sichten

Wenn das EIA/CDIF-Meta-Meta-Modell mit sämtlichen MetaAttributen und AttribuierbarenMetaObjekten aus standardisierten EIA/CDIF-Meta-Modellen instanziiert wurde, so spricht man vom „Integrierten EIA/CDIF-Meta-Modell“[399]. In diesem Zusammenhang ist es wichtig, daß MetaObjekte nicht mehrfach im Integrierten Meta-Modell selbst enthalten sein dürfen.[400]

Die mit Anfang 1998 standardisierten und dieser Arbeit zugrundegelegten EIA/CDIF-Meta-Modelle sind:

- „Foundation Subject Area“,[401]
- „Common Subject Area",[402]
- „Data Modeling Subject Area",[403]
- „Data Flow Subject Area"[404] und
- „Presentation Location and Connectivity Subject Area".[405]

Sollen diese angeführten und standardisierten EIA/CDIF-Meta-Modelle aus dem Integrierten EIA/CDIF-Meta-Modell abgeleitet werden, benötigt man dafür einen Sichtmechanismus, der die entsprechenden MetaObjekte zu referenzieren

---

399 Beispielsweise kann man das Integrierte EIA/CDIF-Meta-Modell dadurch erzeugen, indem etwa entsprechend den Spezifikationen im Abschnitt 3.3.1, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf relationale Datenbankverwaltungssysteme“ auf Seite 108ff, relationale Tabellen für die Repräsentation des EIA/CDIF-Meta-Meta-Modells in einer relationalen Datenbank eingerichtet werden. Daran anschließend werden diese Tabellen mit Tupeln gefüllt, die die MetaObjekte der entsprechenden, standardisierten EIA/CDIF-Meta-Modelle repräsentieren.
Analog dazu wäre das Vorgehen, wenn stattdessen die Object Rexx-Spezifikationen im Abschnitt 3.3.2, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf Object Rexx“ auf Seite 124ff, benutzt würden. In diesem Fall könnte die Klassenhierarchie direkt mit den MetaObjekten der EIA/CDIF-Meta-Modelle instanziiert und die Beziehungen zwischen diesen MetaObjekten mit Hilfe der in der lokalen Umgebung verfügbaren, nach den Meta-Meta-Beziehungstypen benannten Relationen abgebildet werden.

400 In [CDIF94b] kann ein MetaObjekt einerseits über (vollqualifizierte) Namen eindeutig benannt werden, sowie unabhängig davon mit Hilfe des Meta-Meta-Attributes „CDIF-MetaIdentifier“, das als Surrogat für MetaObjekte benutzt wird. Vgl. dazu auch die Ausführungen in den Abschnitten 3.4.2.2 (Seite 142) und 3.4.2.3 (Seite 143).

401 Vgl. [CDIF94f].

402 Vgl. [CDIF95].

403 Vgl. [CDIF96b].

404 Vgl. [CDIF96c].

405 Vgl. [CDIF96d].

imstande ist. Das EIA/CDIF-Komitee hat dafür den Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ vorgesehen, der bereits für die ursprüngliche Definition der EIA/CDIF-Meta-Modelle eingesetzt wurde.[406]

Sofern mehrere EIA/CDIF-Meta-Modelle dieselben MetaObjekte definieren, indem der Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.-SubjectArea 1:N“ instanziiert wird, geht die Information verloren, für welchen GegenstandsBereich ein MetaObjekt ursprünglich definiert wurde.[407]

Unabhängig vom Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.-SubjectArea 1:N“ wird in sämtlichen verabschiedeten EIA/CDIF-Meta-Modellen noch die Information dokumentiert, welche MetaObjekte implizit über Vererbungsbeziehungen als Supertypen referenziert werden, sofern sie nicht bereits im Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ enthalten sind. Hierbei handelt es sich zumeist[408] um die MetaObjekte „RootObject“, „RootEntity“ und „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“, da die letzten beiden die Supertypen für MetaEntitäten und MetaBeziehungen bilden und somit indirekt über die Vererbung dem entsprechenden EIA/CDIF-Meta-Modell zur Verfügung stehen müssen.[409]

---

406 Jedes EIA/CDIF-Meta-Modell, gleichgültig ob standardisiert oder im Stadium des Entwurfes, definiert auch eine Instanz der Meta-Meta-Entität „SubjectArea“, sodaß in weiterer Folge die SammelbarenMetaObjekte den entsprechenden GegenstandsBereichen zugeordnet werden können.

407 Eine entsprechende Erweiterung am Meta-Meta-Modell wurde Anfang 1998 im Rahmen der ISO/IEC JTC1/SC7 WG11 Arbeiten an einer ISO Version der EIA/CDIF-Standards vorgeschlagen, in der ein neuer Meta-Meta-Beziehungstyp vorgesehen wird: „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“.

408 Die folgenden MetaObjekte könnten auch im Hauptteil angeführt werden, das heißt in dem Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ explizit enthalten sein.

409 Zusätzlich müssen auch sämtliche MetaBeziehungen referenziert werden, in denen die definierten beziehungsweise referenzierten MetaEntitäten mit einer Minimumkardinalität von eins oder größer partizipieren und somit immer in derartigen MetaBeziehungen enthalten sein müssen.
[ISOCDIF98b] definiert einen zusätzlichen Meta-Meta-Beziehungstyp, „IsDefinedIn“ für das Meta-Meta-Modell (vollqualifiziert: „0:N **CollectableMetaObject**.*IsDefinedIn*.SubjectArea 1:1“), mit dessen Hilfe der GegenstandsBereich angegeben werden muß, in dem ein MetaObjekt definiert wird. Der Meta-Meta-Beziehungstyp „IsUsedIn“ (vollqualifiziert: „0:N CollectableMetaObject.*IsUsedIn*.SubjectArea 0:N“) wird in ISO/CDIF dafür eingesetzt, ausdrücklich jene Meta-Objekte anzugeben, die darüber hinaus im GegenstandsBereich benutzt werden. Als Folge davon ist es in ISO/CDIF nicht vorgesehen, zusätzliche Meta-Objekte über die Generalisierungshierarchie in einem GegenstandsBereich implizit einzubinden.

3.4.2.2 Benennungsregeln für EIA/CDIF-Meta-Modelle

Die Namen von MetaObjekten müssen entsprechend den Standardisierungsregeln von EIA/CDIF so gewählt werden, daß die damit bezeichneten MetaObjekte innerhalb eines gültigen EIA/CDIF-Meta-Modells eindeutig bestimmbar sind. Es werden hierbei zwischen einfachen (englisch: „simple“) und zusammengesetzten (englisch: „compound“, „qualified“) Namen unterschieden. Für die Benennung von MetaObjekten für EIA/CDIF-Meta-Modelle gelten folgende Regeln:

– MetaEntitäten weisen einen einfachen, eineindeutigen Namen auf.

– MetaBeziehungen weisen einen einfachen, möglicherweise uneindeutigen Namen auf. Ein eineindeutiger Name wird durch das Zusammensetzen der einfachen Namen der Quell-MetaEntität, der Meta-Beziehung selbst, sowie der Ziel-MetaEntität erzeugt, wobei als Zusammensetzungszeichen der Punkt benutzt wird. Der zusammengesetzte (vollqualifizierte) Name von MetaBeziehungen muß eindeutig sein.

– MetaAttribute besitzen einen einfachen Namen, der innerhalb des AttribuierbarenMetaObjektes eineindeutig sein muß.[410] Damit ein MetaAttribut eindeutig über das ganze Meta-Modell hinweg bestimmbar wird, sieht der EIA/CDIF-Standard eine Vollqualifizierung dadurch vor, daß der eineindeutige Name des AttribuierbarenMetaObjekts, durch einen Punkt getrennt, dem Namen des MetaAttributs vorangestellt wird.

Wie in Abschnitt 2.2.2.6 auf Seite 38 dargestellt, gibt es die Vorstellung eines „Integrierten EIA/CDIF-Meta-Modells“, das aus der Vereinigung sämtlicher standardisierter EIA/CDIF-Meta-Modelle entsteht. Für sämtliche, darin definierten MetaObjekte muß daher gelten, daß die einfachen Namen der MetaEntitäten, sowie die vollqualifizierten Namen der MetaBeziehungstypen, sowie der MetaAttribute eineindeutig sind.

---

410 Aufgrund der Einschränkungen, die in den Definitionen zu „MetaAttribute“ beim Meta-Meta-Attribut „Constraints“ in [CDIF94a] auf Seite 44 gegeben sind, dürfen auch in den Supertypen des entsprechenden AttribuierbarenMetaObjekts keine MetaAttribute angegeben sein, die denselben Namen tragen.

Damit können sämtliche Meta-Objekte der standardisierten EIA/CDIF-Meta-Modelle durch die Angabe der eineindeutigen Namen identifiziert (und damit auch eindeutig referenziert) werden.

3.4.2.3 Namen und Surrogate

CDIF Meta-Modelle werden seit 1987 von Arbeitsgruppen des EIA/CDIF-Komitees in einem aufwendigen, oft Jahre dauernden Verfahren definiert, ohne daß dafür ein zentrales Repository zur Verfügung gestanden hätte. Stattdessen wurden die EIA/CDIF-Meta-Modelle in Form von strukturierten Textdokumenten erfaßt und darin im Zuge des Entwicklungsprozesses laufend abgeändert. Aufgrund der natürlichen Fluktuation der Editoren über so einen langen Zeitraum hinweg, kommt es fast zwangsläufig zu besonders aufwendigen und damit auch fehlerträchtigen Überarbeitungen von Entwürfen von Meta-Modellen. In jedem Fall läßt sich denken, daß der Koordinierungsaufwand bei einer derartig dezentralen Vorgehensweise so groß werden kann, daß er von den Arbeitsgruppenmitglieder nicht mehr in allen Fällen wahrgenommen wird. [411]

Wenn nunmehr die Arbeitsgruppen voneinander isoliert EIA/CDIF-Meta-Modelle erarbeiten, müssen vor der Verabschiedung als EIA/CDIF-Standard die entsprechenden Meta-Modellentwürfe daraufhin untersucht werden, daß sie einerseits bei der Benennung der MetaObjekte und andererseits bei der Festlegung

411 Unter anderem hat diese Erkenntnis auch dazu geführt, daß sich der Autor mit Spezifikationen bis hin zu Implementierungen des EIA/CDIF-Meta-Meta-Modells für/in relationale Datenbankverwaltungssysteme sowie in einer objektorientierten Programmiersprache auseinandersetzte. Damit soll ein möglicher Grundstein für die informationstechnisch unterstützte Verwaltung von EIA/CDIF-Meta-Modelldefinitionen erstellt werden. Im Anhang im Abschnitt 6.2, „Implementierungsbeispiele“ auf Seite 423ff, werden grundlegende Implementierungen mit Hilfe des relationalen Datenbankverwaltungssystems Oracle dokumentiert, sowie Object Rexx-Programme, mit deren Hilfe eine beliebige relationale Datenbank mit EIA/CDIF-Meta-Modelldaten beschickt aber auch aus ihr EIA/CDIF-Meta-Modelldaten extrahieren läßt.
Ein im Rahmen dieser Arbeit erstellter Satz an Object Rexx-Programmen erlaubt es darüber hinaus, die Meta-Modelldaten auf ihre Korrektheit entsprechend den EIA/CDIF-Definitionen zu überprüfen. Ein weiterer entwickelter Satz an Object Rexx-Programmen erstellt im Stil der EIA/CDIF-Standards HTML-Dokumente aus den EIA/CDIF-Meta-Modellen, wobei sämtliche Definitionen und Verweise mit dynamischen HTML-Verweisen ausgestattet sind und daher das Erarbeiten der entsprechenden Festlegungen in einer bisher nicht bekannten Form erlauben (vgl. zum Einsatz von HTML-Verweisen auch [Flat98b]). Diese Arbeiten werden aufgrund des hohen Platzbedarfs nicht im Rahmen dieser Arbeit dokumentiert, wenngleich die Ergebnisse zumindest den Mitgliedern von EIA/CDIF in Form der angesprochenen HTML-Aufbereitungen zur Verfügung gestellt wurden.

von Surrogatwerten für das Meta-Meta-Attribut „CDIFMetaIdentifier“ keine Fehler erfolgen.[412]

Dazu muß konzeptionell das hypothetische „Integrierte EIA/CDIF-Meta-Modell“ herangezogen werden, das alle SammelbarenMetaObjekte sämtlicher bisher standardisierter EIA/CDIF-Meta-Modelle beinhaltet. Die Überprüfung muß sicherstellen, daß

- für MetaEntitäten die Namen eindeutig sind, [413]
- für MetaBeziehungen und MetaAttribute die vollqualifizierten Namen eindeutig sind, sowie
- für sämtliche MetaObjekte die Werte im Meta-Meta-Attribut „CDIFMetaIdentifier“ eindeutig sind.

In Abwesenheit eines physisch existierenden Integrierten EIA/CDIF-Meta-Modells ist zu erwarten, daß aufgrund der mittlerweile zahlreich existierenden MetaObjekte und der daraus folgenden mangelnden Übersicht Fehler auftreten.[414]

3.4.2.4 Referenzieren von definierten MetaObjekten

EIA/CDIF-konforme Meta-Modelle zur Definition neuer oder für die Erweiterung bestehender EIA/CDIF-Meta-Modelle können auf die bereits bestehenden, standardisierten EIA/CDIF-Meta-Modelle bezug nehmen. Dies erfolgt, indem mit Hilfe einer Instanz vom Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:1“ das bereits definierte SammelbareMetaObjekt auch dem neuen GegenstandsBereich zugeordnet wird.[415]

---

412 Ein Beispiel dafür ist das ungewollte Verwenden derselben Surrogatwerte für das Meta-Meta-Attribut „CDIFMetaIdentifier“ für *unterschiedliche* MetaObjekte.

413 Dies gilt auch für den Namen „RootObject“, der der einzigen Instanz vom Typ „AttributableMetaObject“ gegeben wird.

414 Ein derartiger, grundlegender Fehler existiert in einem der standardisierten EIA/CDIF-Meta-Modellen durch duplikate Werte für das Meta-Meta-Attribut CDIFMetaIdentifier, vgl. hierzu Fußnote 395 auf Seite 138 und den dort angeführten Verweis auf die im Rahmen dieser Arbeit erfolgten Untersuchungen von EIA/CDIF-Meta-Modellen.

415 Das SammelbareMetaObjekt wird in den derzeit bestehenden standardisierten EIA/CDIF-Meta-Modellen in einem solchen Fall auch vollständig neu definiert, wobei sämtliche Attributwerte dupliziert werden. Ohne Kenntnis der zeitlichen Reihenfolge des Entstehens der derzeitigen Standards ist es nicht entscheidbar, welches MetaObjekt in Wirklichkeit die ursprüngliche Deifnition und welches das Duplikat darstellt.

Fortsetzung folgt auf der nächsten Seite.

Für den Austausch von Modelldaten mit Hilfe eines EIA/CDIF-Transfers können zudem sämtliche Objekte eines GegenstandsBereiches mit einer entsprechenden Klausel referenziert werden.[416] Voraussetzung für das erfolgreiche Importieren von Modelldaten, die standardisierte EIA/CDIF-Meta-Modelle referenzieren, ist die vollständige Kenntnis sämtlicher darin definierter MetaObjekte.[417]

Für das Referenzieren von (bereits definierten) MetaObjekten, das heißt, dem Aufnehmen in eine Instanz des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“, gelten folgende Regeln:[418]

- soll eine MetaBeziehung referenziert werden, so müssen die damit zueinander in Verbindung gesetzten Quell- und Ziel-MetaEntitäten selbst ausdrücklich referenziert werden,
- wenn eine referenzierte MetaEntität in irgendeiner MetaBeziehung zwingend[419] enthalten ist, so hat dies nicht zur Folge, daß diese MetaBeziehung selbst referenziert werden muß,
- wenn ein MetaAttribut referenziert wird, dann folgt daraus, daß das damit beschriebene AttribuierbareMetaObjekt, die MetaEntität oder die MetaBeziehung, auch zu referenzieren ist,
- wenn ein AttribuierbaresMetaObjekt referenziert wird, dann folgt daraus nicht, daß die entsprechenden MetaAttribute benutzt werden müssen, auch keine zwingend[420] (!) vorgeschriebenen MetaAttribute,

---

Wird das EIA/CDIF-Meta-Meta-Modell durch einen Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsDefinedIn*.SubjectArea 1:1“ erweitert, kann eine eindeutige Unterscheidung getroffen werden, indem die Referenzierung von MetaObjekten ausschließlich mit dem Meta-Meta-Beziehungstyp „0:N CollectableMetaObject.*IsDefinedIn*.SubjectArea 0:N“ erfolgt (man beachte die damit notwendig gewordene Änderung der Ziel-Minimalkardinalität auf den Wert „0“). Diese Lösung wurde auch für ISO/CDIF entsprechend [ISOCDIF98b] gewählt.

416 Es handelt sich hierbei um die ‘SUBJECTAREAREFERENCE’-Klausel in EIA/CDIF „SYNTAX.1“ (vgl. hierzu [CDIF94d]) beziehungsweise „ENCODING.1“ (vgl. hierzu [CDIF94e]).

417 Theoretisch könnte die Angabe derartig referenzierter EIA/CDIF-Meta-Modelle dann entfallen, wenn das Importwerkzeug vollständige Kenntnis über das hypothetische Integrierte EIA/CDIF-Meta-Modell, das heißt, sämtliche darin definierten MetaObjekte aufweist.

418 Vgl. hierzu Abschnitte 3.4.3 und 3.4.4 in [CDIF94b], Seiten 19 und 20.

419 Dies ist dann der Fall, wenn der Minimumwert der Kardinalität für diese MetaEntität größer Null ist.

420 Es handelt sich hierbei um jene MetaAttribute, die den Wert „False“ im Meta-Meta-Attribut „IsOptional“ aufweisen.

Fortsetzung folgt auf der nächsten Seite.

– alle über die Generalisierungshierarchie ererbten MetaAttribute werden implizit im GegenstandsBereich benutzt,[421]

– MetaBeziehungen von Supertypen einer MetaEntität werden implizit dann benutzt, wenn auch die zweite beteiligte MetaEntität beziehungsweise ein Subtyp davon im Gegenstandsbereich referenziert wird.[422]

In den bisher standardisierten EIA/CDIF-Meta-Modellen werden die implizit referenzierten SammelbarenMetaObjekte in einem eigenen Abschnitt „Referenced Meta-Object Definitions“ angeführt, der dem Abschnitt „Subject Area Detailed Definitions“ folgt.[423]

3.4.3 Zusammenfassung

Das EIA/CDIF-Meta-Meta-Modell in der Fassung von [CDIF94b] erlaubt die Definition von Meta-Modellen mit Hilfe der Methode der erweiterten konzeptio-

---

Allerdings *müssen* derartige MetaAttribute dann referenziert werden, wenn sie im entsprechenden GegenstandsBereich auch im weiter oben angesprochenen Sinne definiert werden.
Im EIA/CDIF-Standard wird hingegen eine weitere Regel angeführt, die im Widerspruch zu dieser scheint, indem ausgesagt wird, daß MetaAttribute, die in einem GegenstandsBereich benutzt werden müssen, auch zu referenzieren sind. Dies erscheint aber als selbstverständlich, da ja nur durch eine Referenzierung ein MetaAttribut in einen GegenstandsBereich eingeführt werden kann. Handelt es sich hierbei um ein zwingendes MetaAttribut, dann muß es dafür in diesem GegenstandsBereich entsprechend immer einen Wert aufweisen.

421 Daraus folgt, daß ererbte AttribuierbareMetaObjekte und die entsprechend zugeordneten MetaAttribute nicht ausdrücklich referenziert werden müssen.
Wenn allerdings nicht alle MetaAttribute implizit verfügbar gemacht werden sollen, dann müssen die betroffenen übergeordneten AttributierbarenMetaObjekten ausdrücklich gemeinsam mit den gewünschten MetaAttributen referenziert werden, indem entsprechend der Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsDefinedIn*.SubjectArea 1:N“ instanziiert wird.

422 Wenn in einem EIA/CDIF-Transfer mehrere GegenstandsBereiche referenziert werden, gilt diese Regel entsprechend für das Arbeits-Meta-Modell, das sämtliche referenzierte MetaObjekte beinhaltet.

423 EIA/CDIF arbeitet gemeinsam mit ISO/EIA JTC1/SC 7/WG 11 an einer neuen Version (1998) der Dokumentationsrichtlinien, sowohl für EIA/CDIF- als auch ISO/CDIF-Standards, in der unter anderem der Abschnitt „Referenced Meta Object Definitions“ ersatzlos gestrichen wird. In dieser Initiative werden auch die pro AttribuierbarenMetaObjekt ererbten MetaAttribute sowie im Falle von MetaEntitäten die ererbten MetaBeziehungen nicht mehr angeführt. Diese Änderungen erfolgen unter dem Gesichtspunkt der Vereinfachung der Editor-Tätigkeit.
Im Gegensatz dazu werden diese Informationen für die HTML-Darstellungen der EIA/CDIF-Meta-Modelle des Autors beibelassen, die darin eine wichtige Rolle für die Darstellung der Struktur, aber auch für das dynamische Verknüpfen der Definitionen untereinander spielen.

nellen Modellierung. Entsprechend werden dafür die folgenden grundlegenden drei Konzepte benutzt:

- attribuierbare Entitätstypen, die „Meta-Meta-Entitätstypen“,
- Beziehungstypen, die „Meta-Meta-Beziehungstypen“, und
- eine Generalisierungshierarchie.[424]

Das EIA/CDIF-Meta-Meta-Modell definiert die folgenden *Meta-Meta-Entitätstypen* in einer Generalisierungshierarchie, mit deren Hilfe die Definition von Meta-Modellen möglich wird:

- „MetaObject“ (deutsch: „MetaObjekt“), das die Wurzel des EIA/CDIF-Meta-Meta-Modells bildet,
- „SubjectArea“ (deutsch: „GegenstandsBereich“), der den Gegenstandsbereich für die Modellierungskonzepte einer Modellierungssprache in Form eines EIA/CDIF-Meta-Modells bestimmt und ein direkter Subtyp von „MetaObject“ ist,
- „CollectableMetaObject“ (deutsch: „SammelbaresMetaObjekt“), das als Wurzel für jene MetaObjekte dient, die zu einer „SubjectArea“ zusammengefaßt werden sollen, und ein direkter Subtyp von „MetaObject“ ist,
- „AttributableMetaObject“ (deutsch: „AttribuierbaresMetaObjekt“), das selbst über Attribute (Meta-Meta-Entitätstyp „MetaAttribute“) verfügen darf und ein direkter Subtyp von „CollectableMetaObject“ ist,
- „MetaAttribute“ (deutsch: „MetaAttribut“), das ein Attribut repräsentiert, das einem „AttributableMetaObject“ über den Meta-Meta-Beziehungstyp „IsLocalAttributeOf“ zugeordnet sein muß und ein direkter Subtyp von „CollectableMetaObject“ ist,

---

424 Im Meta-Meta-Modell wird die Generalisierungshierarchie für die Meta-Meta-Entitätstypen eingesetzt, auf der Meta-Modellierungsebene werden neben den Meta-Entitätstypen auch die Meta-Beziehungstypen miteinbezogen. Diese wird durch die Analyse des Meta-Meta-Modells klar, da die Bildung einer Hierarchie mit Hilfe des reflexiven Meta-Meta-Beziehungstyps „0:N AttributableMetaObject.*HasSubtype*.-AttributableMetaObject 0:N“ erfolgt und dadurch für sämtliche Subtypen von „AttributableMetaObject“ gilt, das sind im konkreten Fall „MetaEntity“ und „MetaRelationship“.

– „MetaEntity“ (deutsch: „MetaEntität“), das einen Entitätstyp repräsentiert und ein direkter Subtyp von „AttributableMetaObject“ ist, sowie

– „MetaRelationship“ (deutsch: „MetaBeziehung“), das einen binären Beziehungstyp repräsentiert und deswegen immer zwei – nicht notwendigerweise unterschiedliche – MetaEntitäten miteinander in Beziehung setzt und selbst ein direkter Subtyp von „AttributableMetaObject“ ist.

Das EIA/CDIF-Meta-Meta-Modell definiert die folgenden, gerichteten und binären *Meta-Meta-Beziehungstypen*, mit deren Hilfe die Definition von Meta-Modellen möglich wird:

– „IsUsedIn“ (deutsch: „WirdBenutztIn“), voll qualifizierter Name: „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ (deutsch: „0:N **SammelbaresMetaObjekt**.*WirdBenutztIn*.GegenstandsBereich 1:N“), dient dazu, SammelbareMetaObjekte in GegenstandsBereiche zusammenzufassen; gleichzeitig stellt dieser Meta-Meta-Beziehungstyp in EIA/CDIF einen Sichtmechanismus auf das Integrierte EIA/CDIF-Meta-Modell dar,

– „IsLocalMetaAttributeOf“ (deutsch: „IstLokalesMetaAttributVon“), voll qualifizierter Name: „0:N **MetaAttribute**.*IsLocalMetaAttributeOf*.AttributableMetaObject 1:1“ (deutsch: „0:N **MetaAttribut**.*IstLokalesMetaAttributVon*.AttribuierbaresMetaObjekt 1:1“), dient dazu, jedes MetaAttribut dem entsprechenden AttribuierbarenMetaObjekt zuzuordnen,

– „HasSubtype“ (deutsch: „BesitztUntergeordnetes“), voll qualifizierter Name: „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“ (deutsch: „0:N AttribuierbaresMetaObjekt.*BesitztUntergeordnetes*.AttribuierbaresMetaObjekt 0:N“), erlaubt den Aufbau einer Generalisierungshierarchie,

– „HasSource“ (deutsch: „HatAlsQuelle“), voll qualifizierter Name: „0:N **MetaRelationship**.*HasSource*.MetaEntity 1:1“ (deutsch: „0:N **MetaBeziehung**.*HatAlsQuelle*.MetaEntität 1:1“), legt fest, welche MetaEntität als Quelle der gerichteten Beziehung angesehen wird,

– „HasDestination“ (deutsch: „HatAlsZiel“), voll qualifizierter Name: „0:N **MetaRelationship**.*HasDestination*.MetaEntity 1:1“ (deutsch: „0:N **Meta-**

Beziehung.*HatAlsZiel*.MetaEntität 1:1“), legt fest, welche MetaEntität als Ziel[425] der gerichteten Beziehung angesehen wird,

Aufgrund der vorgesehenen Reflexivitätseigenschaft, kann das EIA/CDIF-Meta-Meta-Modell mit den Mitteln definiert werden, die es selbst festlegt. Unabhängig davon werden die EIA/CDIF-Meta-Modelle mit Hilfe des EIA/CDIF-Meta-Meta-Modells gebildet, das daher die Struktureigenschaften von Meta-Modellen festlegt. Anders ausgedrückt, können die Konzepte von Meta-Modellen nur mit Hilfe von Instanzen von Meta-Meta-Entitätstypen und von Meta-Meta-Beziehungstypen gebildet werden. Daraus folgt auch, daß die bedingten Beziehungstypen, wiewohl in [CDIF94b][426] vorgesehen, nicht Bestandteil von EIA/CDIF-Meta-Modellen werden können. Es fehlt dafür das Mittel, dies in Form von Instanzen der Entitäts- beziehungsweise Beziehungstypen des Meta-Meta-Modells auszudrücken.[427]

Das gegenwärtige EIA/CDIF-Meta-Meta-Modell sieht im Unterschied zur „Meta Object Facility“ (abgekürzt: MOF)[428] der OMG keine Möglichkeit vor, auch dynamische Eigenschaften auf der Ebene des Meta-Meta-Modells[429] vorzusehen, beispielsweise Funktionen oder Methoden. Auf der Ebene von EIA/CDIF ist es trotzdem möglich, MetaAttribute mit entsprechenden dynamischen und direkt ausführbaren Spezifikationen zu definieren.[430]

---

425 Das Ziel wird in den graphischen Repräsentationen durch die Pfeilspitze gekennzeichnet.

426 Vgl. ebenda Seiten 17 bis 18,

427 Eine Möglichkeit, einander ausschließende Beziehungstypen ausdrücklich zu dokumentieren, wird in dieser Arbeit in Form des Meta-Modells „M2Level“ im Abschnitt 4.4, „Definition eines EIA/CDIF-konformen Meta-Modells „M2Level““ auf Seite 356ff, gegeben. Es hat unter anderem den Vorteil, daß das EIA/CDIF-Meta-Meta-Modell dafür nicht geändert werden muß.

428 Vgl. in diesem Zusammenhang das 1997 von OMG standardisierte [MOF97a] beispielsweise die MOF-Definitionen für „BehavioralFeature“ auf Seite 3-47.
Mit Hilfe des für diese Arbeit konzipierten EIA/CDIF-konformen Meta-Modells „M2Level“ (vgl. Abschnitt 4.4, Seite 356ff weiter unten) wird es zudem möglich, entsprechende Definitionen ohne Änderungsaufwand für das Meta-Meta-Modell durch einfache Erweiterung von „M2Level“ ausdrücklich vorzusehen.

429 Auf dieser Ebene definiert, könnten dynamische Eigenschaften bereits für die definierten Konzepte von Meta-Modellen selbst definiert werden.

430 Vgl. in diesem Zusammenhang unter anderem auch das standardisierte EIA/CDIF-Meta-Modell „Common“ ([CDIF95]), das unter andem auf Seite 18 die international definierten Programmiersprachen vorsieht, mit deren Hilfe Spezifikationen von Programmcode erfolgen sollen. Dies erfolgt, indem für ein MetaObjekt einerseits ein MetaAttribut definiert wird, das als Wert die gewünschte Programmiersprache beinhaltet, und zum anderen ein MetaAttribut, das die Spezifikationen in der angegebenen Programmiersprache aufweist. Vgl. zum Beispiel die Definition der MetaEntität „Derivation“ in [CDIF95] auf Seite 34ff,

Fortsetzung folgt auf der nächsten Seite.

Im Vergleich zur MOF[431] fällt auch auf, daß Datentypen in EIA/CDIF vordefiniert sind und nicht im Rahmenwerk des EIA/CDIF-Meta-Meta-Modells selbst enthalten sind. Somit können für MetaAttribute nur die vordefinierten EIA/CDIF-Datentypen[432] benutzt werden. Allerdings ist es auf Meta-Modell-Ebene möglich, beliebige Datentypen zu definieren, wie dies im EIA/CDIF-Entwurf „Data Definition“ in [CDIF96h] der Fall ist.[433]

insbesondere die MetaAttribute „DerivationLanguage“ und „DerivationText“.
Sofern auch Signaturen von Funktionen, Prozeduren beziehungsweise von Methoden als wichtig erachtet werden, könnten diese dafür notwendigen Konzepte auch in Form eines EIA/CDIF-Meta-Modells definiert und anschließend angewandt werden. In diesem Zusammenhang ist der EIA/CDIF-Meta-Modell Entwurf „Object-oriented Analysis and Design“ (deutsch: „objektorientierte Analyse und Entwurf“) interessant, der in [CDIF96f] dokumentiert ist.

431 Vgl. hierzu die Definition von „DataType“ in [MOF97a], Seite 3-35 bis 3-36.

432 Vgl. hierzu Abschnitt 3.1.2, „EIA/CDIF-Datentypen“ auf Seite 54ff.

433 Auch hier stellt sich die Frage, inwieweit es für EIA/CDIF-Meta-Modelle sinnvoll ist, auf Meta-Meta-Modellebene freie Datentypvereinbarungen zuzulassen. Aufgrund der Auswertung sämtlicher standardisierter EIA/CDIF-Meta-Modelle läßt sich sogar feststellen, daß bisher die folgenden EIA/CDIF-Datentypen für Meta-Modelle nicht verwendet wurden: „Bitmap“, „IntegerList“ und „PointList“.

4 EIA/CDIF Meta-Modelle

In diesem Kapitel werden zunächst die zwei wichtigsten EIA/CDIF-Meta-Modelle, „Foundation“ und „Common“, im Detail vorgestellt und diskutiert. Daran anschließend wird eine übersichtsmäßige Darstellung der bis Anfang 1998 standardisierten EIA/CDIF-Meta-Modelle gegeben, die gemeinsam mit „Foundation“ und „Common“ zum „Integrierten EIA/CDIF-Meta-Modell“[434] (abgekürzt: „IMM“) zusammengefaßt werden.

Die Präsentation verschiedener struktureller Auswertungen dieses Integrierten EIA/CDIF-Meta-Modells stellt daher einen Schwerpunkt dieses Kapitels dar. Im Anhang werden in Abschnitt 6.1, „Querverweistabellen für das Integrierte EIA/CDIF-Meta-Modell“ auf Seite 395ff, die (vollqualifizierten) Namen aus dem Integrierten EIA/CDIF-Meta-Modell gemeinsam mit ihren Surrogaten[435] kreuzweise dargestellt.[436]

Einen weiteren Schwerpunkt stellt der Abschnitt 4.2[437] dar, in dem das hypothetische „Integrierte EIA/CDIF-Meta-Modell“ zum ersten Mal physisch erstellt und dokumentiert wird.

---

434 Dieses in den EIA/CDIF-Standards hypothetisch existierende Integrierte EIA/CDIF-Meta-Modell wurde im Rahmen dieser Arbeit nach Kenntnis des Autors *erstmals* physisch im Herbst 1997 erstellt und wird mit dieser Arbeit *erstmals* öffentlich dokumentiert. Dazu wurden die elektronischen Formen der standardisierten EIA/CDIF-Standards, die dem Autor von EIA/CDIF freundlicherweise zur Verfügung gestellt wurden, mit Object Rexx-Programmen ausgelesen und in eine relationale Datenbank der Firma Oracle übergeführt, deren Tabellen den Spezifikationen entsprechen, die weiter oben in Abschnitt 3.3.1.1, Seite 108ff, vorgestellt wurden. Somit lassen sich mit Hilfe von Implementierungen in SQL und mit Hilfe von Object Rexx-Programmen unter anderem Auswertungen über sämtliche (Struktur-) Aspekte der EIA/CDIF-Meta-Modell-Spezifikationen durchführen.

435 Damit ist, wie bereits oben ausführlich diskutiert, natürlich das Meta-Meta-Attribut „CDIFMetaIdentifier“ gemeint.

436 Dadurch wird es möglich, ohne Kenntnis der einzelnen, standardisierten EIA/CDIF-Meta-Modelle selbst, die (qualifizierten) Namen und die dafür vergebenen Surrogate kennenzulernen und für eigene Implementierungen bereits zu berücksichtigen.
Im konkreten werden folgende drei Querverweistabellen, die im Rahmen dieser Arbeit entstanden sind, im Anhang dokumentiert:

(1) Abschnitt 6.1.1, ‘Querverweistabelle der standardisierten Meta-Objekte, sortiert nach dem Meta-Meta-Attribut „CDIFMetaIdentifier“’ auf Seite 395ff,

(2) Abschnitt 6.1.2, ‘Querverweistabelle der standardisierten Meta-Objekte, sortiert nach dem Meta-Meta-Attribut „Name“’ auf Seite 405ff und

(3) Abschnitt 6.1.3, ‘Querverweistabelle der standardisierten AttribuierbarenMeta-Objekte gemeinsam mit ihren Meta-Attributen, sortiert nach dem vollqualifizierten Namen’ auf Seite 414ff.

437 Vgl. Seite 267ff weiter unten.

Nach einer zusammenfassenden Diskussion in Abschnitt 4.3[438], in der auch auf die wichtigsten strukturellen Einschränkungen des Meta-Meta-Modells[439] zurückgegriffen wird, folgt als weiterer Schwerpunkt die Definition des Meta-Modells „M2Level“.[440] Mit diesem Meta-Modell können Einschränkungen, die direkt auf das EIA/CDIF-Meta-Meta-Modell zurückführbar sind, umgangen werden, ohne daß das Meta-Meta-Modell selbst geändert werden müßte.

Die Darstellung des Aufbaus der verschiedenen AttribuierbarenMetaObjekte erfolgt mit Hilfe der im EIA/CDIF-Meta-Meta-Modell festgelegten Meta-Meta-Entitätstypen und Meta-Meta-Beziehungstypen. So werden beispielsweise sämtliche Meta-Entitätstypen als Instanzen des Meta-Meta-Entitätstyps „MetaEntity“ und sämtliche Meta-Beziehungstypen als Instanzen des Meta-Meta-Entitätstyps „MetaRelationship“ angesehen. Sämtliche detaillierten tabellarischen Beschreibungen von AttribuierbarenMetaObjekten und Meta-Attributen erfolgen entsprechend der im Kapitel 3 festgelegten Struktur.[441]

In den Auflistungen werden im Gegensatz zu den Darstellungen in den EIA/CDIF-Standards[442] die Werte des Meta-Meta-Attributs „CDIFMetaIdentifier“ für alle MetaObjekte mitangeführt sowie die Bezeichner von Meta-Entitätstypen fett dargestellt, wenn deren Instanzen in zwingend vorgeschriebenen Meta-Beziehungstypen enthalten sein müssen. Meta-Beziehungstypen werden dann als zwingend angesehen, wenn der Minimumwert in einer der beiden Kardinalitäten größer Null ist, sodaß Instanzen des auf der gegenüberliegenden Seite vorkommenden Meta-Entitätstyps in Instanzen des entsprechenden Meta-Beziehungstyps zwingend enthalten sein müssen.[443] Zudem werden gemein-

438 Vgl. Seite 340ff weiter unten.

439 Vgl. Abschnitt 3.4, „Zusammenfassende Diskussion des EIA/CDIF-Meta-Meta-Modells“ auf Seite 129ff weiter oben.

440 Vgl. Abschnitt 4.4, ‘Definition eines EIA/CDIF-konformen Meta-Modells „M2Level“’ auf Seite 356ff weiter unten.

441 Zusätzliche tabellarische Aufbereitungen werden bei ihrem ersten Einsatz in diesem Kapitel erklärt, sofern dies als notwendig erscheint.

442 Dieselbe Aussage trifft ebenso auf die entsprechenden ISO/CDIF Entwürfen zu.

443 Damit sind derartige Meta-Entitätstypen auch existenzabhängig von den entsprechenden zwingenden Meta-Beziehungstypen.

sam mit den vollqualifizierten Bezeichnern der Meta-Beziehungstypen deren Kardinalitäten in der üblichen Minium/Maximum Notation dargestellt.[444]

Die Zusammensetzung in bezug auf die ererbten Meta-Attribute erfolgt entlang der Generalisierungshierarchie, die im Meta-Meta-Modell durch den Meta-Meta-Beziehungstyp „0:N AttributableMetaObject.*HasSubtype*.AttributableMetaObject 0:N“. Im Falle von Mehrfachvererbung, werden zunächst die direkten Supertypen alphabetisch sortiert[445], ehe dann mit dem üblichen Algorithmus zunächst in die Tiefe und dann von rechts nach links die Supertypen aufgelöst werden. In der summarischen Darstellung werden Meta-Attribute, die lokal für den entsprechenden Typ festgelegt sind, prägnant mit ihrem Datentyp und im Falle des EIA/CDIF-Datentyps „Enumerated“ mit den aufzählbaren, erlaubten Werten dargestellt.[446]

Der Aufbau der einzelnen Unterabschnitte erfolgt systematisch nach einem einheitlichen Schema:

– Kurzcharakterisierung des EIA/CDIF-Meta-Modells beziehungsweise detaillierte Darstellung der Meta-Entitätstypen und Meta-Beziehungstypen im Falle der GegenstandsBereiche „Foundation“[447] und „Common“[448],

– Auswertungen der Definitionen mit der Darstellung

  - von referenzierten Meta-Objekten,
  - der Dokumentation von aufgefundenen Fehlern,
  - der Generalisierungshierarchie des Meta-Modells und

444 Sofern der Minimumwert einer Kardinalität ungleich Null ist, wird sie – genauso wie der Bezeichner des betroffenen Meta-Entitätstyps im vollqualifizierten Namen des betroffenen, zwingenden Meta-Beziehungstyps – fett dargestellt.

445 Von EIA/CDIF aus gibt es *keine* vorbestimmte Reihenfolge im Falle von mehreren Supertypen. Daher ist eine alphabetische Reihenfolge gleichermaßen zulässig wie eine, die nach der Anzahl der Buchstaben in den Bezeichnern der Supertypen geordnet wäre.

446 Damit soll es möglich werden, EIA/CDIF-Meta-Modelle zu definieren, ohne die EIA/CDIF-Standards selbst zu besitzen. Sofern die Semantik der einzelnen Typen im Detail verstanden werden soll, ist es unumgänglich, die entsprechenden EIA/CDIF-Standards zu kaufen. Dasselbe trifft auch auf die in gerade in Entwicklung befindlichen ISO/CDIF-Standards, wobei sich die entsprechenden Entwürfe durch die Überarbeitung der entsprechenden ISO/IEC-Arbeitsgruppe leicht von EIA/CDIF unterscheiden können. Siehe in diesem Kontext auch die Ausführungen in Kapitel 5, „Zusammenfassung und Ausblick“ auf Seite 385ff.

447 Vgl. Abschnitt 4.1.1, ‘FND – Das EIA/CDIF-Meta-Modell „Foundation“’ auf Seite 155ff.

448 Vgl. Abschnitt 4.1.2, ‘CMMN – Das EIA/CDIF-Meta-Modell „Common“’ auf Seite 176ff.

- des summarischen[449] Aufbaus der einzelnen AttribuierbarenMetaObjekte.

Die summarische Darstellung der AttribuierbarenMetaObjekte aus den Meta-Modellen für die GegenstandsBereiche „Datenflußmodellierung“[450], „Datenmodellierung“[451] und „PLAC“[452] erfolgt gemeinsam im Rahmen der Darstellung der MetaObjekte des Integrierten EIA/CDIF-Meta-Modells in Abschnitt 4.2.[453]

449 „Summarisch“ wird in diesem Zusammenhang als Gegensatz zu „detailliert“ verstanden, wodurch ausgedrückt werden soll, daß die Darstellung der einzelnen AttribuierbarenMetaObjekte im Gegensatz zum Meta-Meta-Modell und zu den grundlegenden in diesem Abschnitt dargestellten Meta-Modellen „Foundation“ und „Common“ ohne eigenständige, detaillierte Tabellen für jedes einzelne benutzte MetaObjekt erfolgt.

450 Vgl. Abschnitt 4.1.3.1, „DFM – EIA/CDIF-Meta-Modell für die Datenflußmodellierung“ auf Seite 224ff.

451 Vgl. Abschnitt 4.1.3.2, „DMOD – EIA/CDIF-Meta-Modell für die Datenmodellierung“ auf Seite 241ff.

452 Vgl. Abschnitt 4.1.3.3, ‘PLAC – EIA/CDIF-Meta-Modell für „Presentation Location and Connectivity“’ auf Seite 257ff.

453 Vgl. Seite 267ff unten.

4.1 Die standardisierten EIA/CDIF-Meta-Modelle

In diesem Abschnitt werden die von EIA/CDIF mit Jänner 1998 standardisierten Meta-Modelle dargestellt. Die standardisierten EIA/CDIF-Meta-Modelle „Foundation“ sowie „Common“ werden aufgrund ihrer enormen Wichtigkeit für die Erstellung von EIA/CDIF-Meta-Modellen im Detail behandelt. Sämtliche Definitionen dieser standardisierten EIA/CDIF-Meta-Modelle wurden im Rahmen dieser Arbeit mit Hilfe von Object Rexx-Programmen in einer relationalen Datenbank (Oracle) gespeichert, überprüft[454] und ausgewertet[455], wobei für die letzten beiden Tätigkeiten zusätzlich auch SQL herangezogen werden konnte.

4.1.1 FND – Das EIA/CDIF-Meta-Modell „Foundation“

[CDIF94f] beinhaltet die Definitionen für das standardisierte EIA/CDIF-Meta-Modell „Foundation“[456] (deutsch: „Fundament“). Es ist fundamental, da EIA/-CDIF die Verwendung dieses Meta-Modells für die Bildung von eigenen Meta-Modellen vorschreibt und die grundlegende Struktur vorgibt, indem es die zwei fundamentalen Konzepte „Entitätstyp“(„RootEntity“) und „Beziehungstyp“ („IsRelatedTo“ beziehungsweise voll qualifiziert: „0:N RootEntity.-*IsRelatedTo*.-RootEntity 0:N“) einführt, die selbst direkte Subtypen von „RootObject“ sind, der Wurzel eines jeden EIA/CDIF-konformen Meta-Modells.[457] Die Definition von neuen MetaEntitäten und MetaBeziehungen erfolgt durch direkte oder indirekte Spezialisierung von „RootEntity“ respektive „IsRelatedTo“.

Dieses Meta-Modell weist unter anderem drei Besonderheiten auf:

1. es bildet den Wurzelknoten und die zwei ersten Blätter, eines für die Repräsentation von Entitätstypen und eines für die Beziehungstypen, eines *jeden* Meta-Modells,

---

454 Unter anderem wurde dadurch ein Fehler in der Vergabe von Surrogatwerten entdeckt, der darin besteht, daß ein Wert im Meta-Meta-Attribut „CDIFMetaIdentifier“ mehr als einem MetaObjekt zugeordnet ist. Vgl. hierzu den Abschnitt 4.1.3.2.3.2 auf Seite 251ff.

455 In weiterer Folge wurden derartige Überprüfungen und Auswertungen auch mit SQL möglich, wie dies im Anhang im Abschnitt 6.2.1.3, „EIA/CDIF-Analysebeispiele in SQL“ auf Seite 440ff, beispielhaft gezeigt wird.

456 Dieser GegenstandsBereich wurde während seiner Entwicklung vom EIA/CDIF-Komitee mit der Abkürzung „FND“ bezeichnet.

457 Vgl. in diesem Zusammenhang unter anderem [CDIF94f], Abschnitt 4.1 auf Seite 12 unten, [CDIF94a], Abschnitt 4.4.2 auf Seite 16 oder [CDIF94b] Abschnitt 3.4.2 auf Seite 19, 2. Absatz.

2. der Wurzelknoten „RootObject“ ist eine Instanz des Meta-Meta-Entitätstyps „AttributableMetaObject“, das Blatt „RootEntity“ als Supertyp für sämtliche Entitätstypen von Meta-Modellen ist Instanz des Meta-Meta-Entitätstyps „MetaEntity“ und das Blatt „0:N RootEntity.*IsRelatedTo*.-RootEntity 0:N“ als Supertyp für sämtliche Beziehungstypen von Meta-Modellen ist Instanz des Meta-Meta-Entitätstyps „MetaRelationship“,

3. es legt für die graphische Darstellung von Entitätstypen Rechtecke fest und für Beziehungstypen den Beziehungstyppfeil, wie wir ihn bereits weiter oben[458] bei der Auseinandersetzung mit dem EIA/CDIF-Meta-Meta-Modell kennengelernt haben. In graphischen Darstellungen von (Ausschnitten von) Meta-Modellen wird die Generalisierungshierarchie[459] nur für MetaEntitäten, nicht aber für MetaBeziehungen dargestellt.

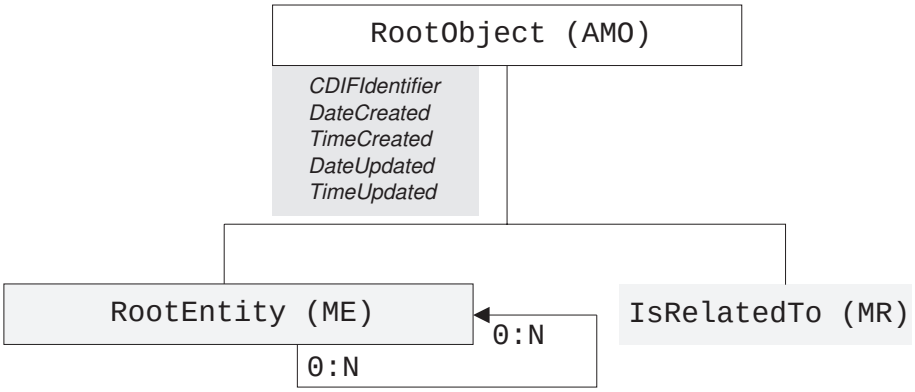


Abbildung 4-1: Das standardisierte EIA/CDIF-Meta-Modell „Foundation“[460]

Da sämtliche EIA/CDIF-konformen Meta-Modelle das fundamentale EIA/CDIF-Meta-Modell referenzieren müssen, stellen sie zwangsläufig Hierarchien dar. Dies deshalb, weil die in ihnen enthaltenen Meta-Entitätstypen und Meta-Beziehungstypen letztendlich „RootEntity“ beziehungsweise „0:N RootEntity.-*IsRelatedTo*.RootEntity 0:N“ spezialisieren, die selbst ihre Wurzel in „RootObject“ finden.

458 Vgl. die Ausführungen über die Nutzung des Pfeiles im Abschnitt 3.1, „Definition des EIA/CDIF-Meta-Meta-Modells“, Seite 56.

459 In der Generalisierungshierarchie wird die Über- und Unterordnung für alle AttribuierbarenMetaObjekte gezeigt.

460 Vgl. [CDIF94f], Seite 13.

Es ist interessant zu beobachten, daß alle drei Meta-Entitätstypen *unterschiedliche* Meta-Meta-Entitätstypen instanziieren, aber daß in weiterer Folge weder die Subtypen von „RootEntity“ noch von „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ den Meta-Meta-Entitätstyp wechseln dürfen.[461] Daraus folgt, daß der Meta-Meta-Entitätstyp „AttributableMetaObject“ demgemäß nur über eine *einzige* Instanz verfügt, nämlich „RootObject“. In Abbildung 4-1 werden die Meta-Meta-Entitätstypen, von denen die MetaObjekte Instanzen sind, in ihren Abkürzungen in Klammern neben den Namen der MetaObjekte angeführt. Betrachtet man die strukturelle Anordnung dieser Meta-Meta-Entitätstypen im Meta-Meta-Modell[462], so läßt sich feststellen, daß die entsprechenden Instanzen der gleichen Anordnung folgen.[463]

4.1.1.1 Definitionen der „fundamentalen“ Konzepte

Die Definitionen für die MetaObjekte des standardisierten EIA/CDIF-Meta-Modells „Foundation“ erfolgen aufgrund des kleinen Umfangs in einer Top-Down-, Links-Rechts-Reihenfolge entsprechend ihrer Position in der hierarchischen Generalisierungshierarchie der obigen Abbildung 4-1.[464]

---

461 Diese Aussage wird durch die im Rahmen dieser Arbeit erfolgten detaillierten Analysen *sämtlicher* standardisierter und in Vorbereitung befindlicher EIA/CDIF-Meta-Modelle untermauert. Zudem wird für die Übertragung des EIA/CDIF-Standards in ISO/CDIF, der für November 1998 vorgesehen ist, in den Entwürfen für das fundamentale Meta-Modell ausdrücklich festgehalten, daß Subtypen von „RootEntity“ und „0:N RootEntity.-*IsRelatedTo*.RootEntity 0:N“ die Meta-Meta-Entitätstypen nicht mehr wechseln dürfen.

462 Vgl. beispielsweise Abbildung 3-1 auf Seite 53 weiter oben, Abbildung 3-5 auf Seite 107 weiter oben oder Abbildung 6-1 – mit den deutschen Übersetzungen der englischen Bezeichner – auf Seite 423 weiter unten.

463 In der M2-Schicht werden die Instanzen des Meta-Meta-Entitätstyps „MetaAttribute“ zu den Instanzen der AttribuierbarenMetaObjekte aggregiert. Eine Entsprechung für diesen Meta-Meta-Entitätstyp „MetaAttribute“ findet sich im fundamentalen Meta-Modell nicht. Im EIA/CDIF-Standard wäre es im übrigen nicht zwingend vorgeschrieben, daß das fundamentale Meta-Modell auf der M2-Ebene dieselbe grundlegende Struktur aufweist wie die M3-Ebene. Nachdem aber das Entity-Relationship-Attribute-Modell der konzeptionellen EIA/CDIF-Modellierung zugrunde gelegt ist, überrascht diese Entsprechung auf der anderen Seite nicht. Ein Vorteil davon liegt möglicherweise darin, daß dieselben Vorstellungen (und zumindest *ähnliche* Bezeichnungen) für das Modellieren der M3- als auch der M2-Schicht verwendet werden, ein Nachteil liegt möglicherweise darin, daß dadurch die strikte Grenze zwischen diesen beiden Schichten übersehen werden und in weiterer Folge zu Verwirrungen führen könnte.

464 Sämtliche Definitionen finden sich auch als Extension zu den Spezifikationen für relationale Datenbanken (und entsprechend auch für Object Rexx-Instanziierungen der Klassen und Relationen) in Abbildung 3-6, „Tabellen, die Meta-Meta-Entitätstypen repräsentieren und die Auszüge der Meta-Objektdefinitionen für das fundamentale EIA/CDIF-Meta-Modell enthalten“ auf Seite119 und Abbildung 3-7, „Tabellen, die Meta-Meta-

Fortsetzung folgt auf der nächsten Seite.

4.1.1.1.1 Meta-Entitätstyp „RootObject“

Der Meta-Entitätstyp „RootObject“[465] bildet die Wurzel eines jeden EIA/CDIF-entsprechenden Meta-Modells. Obwohl die einzige Instanz des Meta-Meta-Entitätstyps „AttributableMetaObject“, fungiert sie innerhalb der M2-Schicht als ein Meta-Entitätstyp[466]. Aufgrund der objektorientierten Eigenschaften des erweiterten Entity-Relationship-Modells erben sämtliche Subtypen alle Meta-Attribute, die für diesen fundamentalen und omnipotenten Meta-Entitätstyp definiert sind. In der graphischen Darstellung wird für diesen Meta-Entitätstyp das Rechteck benutzt.

Tabelle 4-1 beschreibt den Meta-Entitätstyp „RootObject“.

Name des Attributs	**Bedeutung**
Name	***RootObject***
CDIFMetaIdentifier	***1***
SubtypeOf	–
SupertypeOf	*RootEntity* *RootEntity.IsRelatedTo.RootEntity*
Description	*Wurzel eines jeden EIA/CDIF-Meta-Modells, definiert jene Meta-Attribute, die allen Meta-Entitäten gemein sind.*
Usage	*Diese Meta-Entität ist abstrakt und soll daher nicht instanziiert werden.*
Aliases	–
Constraints	*Es dürfen diesem Meta-Entitätstyp durch die Erweiterung im Rahmen eines EIA/CDIF-Austausches keine weiteren Supertypen vorangestellt werden.*

Tabelle 4-1: Meta-Entitätstyp „RootObject“

Beziehungstypen repräsentieren und die Meta-Entitäten von [CDIF94f] entsprechend dem EIA/CDIF-Meta-Meta-Modell miteinander in Beziehung setzen“ auf Seite 123.

465 Vgl. [CDIF94f], Seite 17ff.

466 Dieser Schluß wird einfach aus der graphischen Repräsentation gezogen, die dieselbe wie für Meta-Entitätstypen ist.
Es sei darauf hingewiesen, daß [CDIF94f] auf Seite 17 „RootObject“ unter dem Abschnitt „5.3 *AttributableMetaObject* Definitions“ klassifiziert und dieses Konzept sich daher ausdrücklich von „5.4 *Meta-entity Definitions“* unterscheidet. Aus dieser Einteilung im EIA/CDIF-Standard ist der Schluß möglich, daß „RootObject“ eine Instanz von „AttributableMetaObject“ und nicht etwa von „MetaEntity“ ist.

Folgende Meta-Attribute werden zum Meta-Entitätstyp „RootObject“ aggregiert, wobei mit Ausnahme von „CDIFIdentifier“ alle Meta-Attribute optional sind:

1. „CDIFIdentifier“,
2. „DateCreated“,
3. „DateUpdated“,
4. „TimeCreated“ und
5. „TimeUpdated“.

Sofern im Rahmen eines EIA/CDIF-Transfers sämtliche Meta-Entitätstypen und Meta-Beziehungstypen über zusätzliche Meta-Attribute verfügen sollen, können diese einfach durch die Erweiterung des fundamentalen Meta-Modells festgelegt werden.

4.1.1.1.1.1 Meta-Attribut „CDIFIdentifier“

Das zwingend vorgeschriebene Meta-Attribut „CDIFIdentifier“[467] ist vom EIA/CDIF-Datentyp „Identifier“ und identifiziert sämtliche Instanzen eines Meta-Entitätstyps oder eines Meta-Beziehungstyps aufgrund seiner Surrogateigenschaften eindeutig. Damit wird durch dieses Meta-Attribut in einem EIA/CDIF-Transfer auf Modellebene die Entitätsintegritätsbedingung gewährleistet. Tabelle 4-2 beschreibt das Meta-Attribut „CDIFIdentifier“.

Name des Attributs	**Bedeutung**
Meta-Attribute Name	***CDIFIdentifier***
CDIFMetaIdentifier	***5***
Description	*Enthält eindeutigen Wert im Rahmen eines EIA/CDIF-Transfers.*
Usage	*CDIFIdentifier werden für den „Modell-Bereich“ (M1-Schicht) in EIA/CDIF-Transfers benötigt, um Instanzen von Meta-Entitätstypen beziehungsweise von Meta-Beziehungstypen sowie deren Subtypen eindeutig referenzieren zu können.*[468]

467 Vgl. [CDIF94f], Seite 18.

468 Es existiert für die Modellebene (M1-Schicht) nur dieses Konzept, um Entitäten und Beziehungen in einem EIA/CDIF-Austausch voneinander zu unterscheiden.

Name des Attributs	Bedeutung
Aliases	_ [469]
Constraints	–
DataType	*Identifier*
Domain	–
Length	–
IsOptional	*False*

Tabelle 4-2: Meta-Attribut „CDIFIdentifier“

4.1.1.1.1.2 Meta-Attribut „DateCreated“

Das optionale Meta-Attribut „DateCreated“[470] ist vom EIA/CDIF-Datentyp „Date“ und gibt an, wann die Instanz des Meta-Entitätstyps bzw. Meta-Beziehungstyps angelegt wurde. Tabelle 4-3 beschreibt das Meta-Attribut „DateCreated“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***DateCreated***
CDIFMetaIdentifier	***6***
Description	*Enthält den Erstellungstag des Objektes.*
Usage	–
Aliases	–
Constraints	–
DataType	*Date*
Domain	*Jedes gültige Datum.*
Length	–
IsOptional	*True*

Tabelle 4-3: Meta-Attribut „DateCreated“

469 Als Alias böte sich hier der Begriff „Surrogat“ an, vgl. z.B. [Codd79].

470 Vgl. [CDIF94f], Seite 18.

4.1.1.1.1.3 Meta-Attribut „DateUpdated“

Das optionale Meta-Attribut „DateUpdated“[471] ist vom EIA/CDIF-Datentyp „Date“ und gibt an, wann die Instanz des Meta-Entitätstyps bzw. Meta-Beziehungstyps zuletzt verändert wurde. Tabelle 4-4 beschreibt das Meta-Attribut „DateCreated“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***DateUpdated***
CDIFMetaIdentifier	***7***
Description	*Enthält das jüngste Änderungsdatum des Objektes.*
Usage	–
Aliases	–
Constraints	*Das absolute Datum, das aus der Kombination von „DateUpdated“ mit „TimeUpdated“ entsteht, sollte älter sein, als jenes, das sich aus der Kombination von „DateCreated“ mit „TimeCreated“ ergibt, sofern dieses Meta-Attribut überhaupt einen Wert aufweist.*
DataType	*Date*
Domain	*Jedes gültige Datum.*
Length	–
IsOptional	*True*

Tabelle 4-4: Meta-Attribut „DateUpdated“

4.1.1.1.1.4 Meta-Attribut „TimeCreated“

Das optionale Meta-Attribut „TimeCreated“[472] ist vom EIA/CDIF-Datentyp „Time“ und gibt an, wann die Instanz des Meta-Entitätstyps bzw. Meta-Beziehungstyps angelegt wurde. Tabelle 4-5 beschreibt das Meta-Attribut „TimeCreated“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***TimeCreated***
CDIFMetaIdentifier	***8***

471 Vgl. [CDIF94f], Seite 18.
472 Vgl. [CDIF94f], Seite 19.

Name des Attributs	Bedeutung
Description	*Enthält die Erstellungszeit des Objektes.*
Usage	–
Aliases	–
Constraints	–
DataType	*Time*
Domain	*Jede gültige Zeit.*
Length	–
IsOptional	*True*

Tabelle 4-5: Meta-Attribut „TimeCreated“

4.1.1.1.1.5 Meta-Attribut „TimeUpdated“

Das optionale Meta-Attribut „TimeUpdated“[473] ist vom EIA/CDIF-Datentyp „Date“ und gibt an, wann die Instanz des Meta-Entitätstyps bzw. Meta-Beziehungstyps zuletzt verändert wurde. Tabelle 4-6 beschreibt das Meta-Attribut „TimeCreated“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***TimeUpdated***
CDIFMetaIdentifier	***9***
Description	*Enthält die jüngste Änderungszeit des Objektes.*
Usage	–
Aliases	–
Constraints	*Das absolute Datum, das aus der Kombination von „DateUpdated“ mit „TimeUpdated“ entsteht, sollte älter sein, als jenes, das sich aus der Kombination von „DateCreated“ mit „TimeCreated“ ergibt, sofern dieses Meta-Attribut überhaupt einen Wert aufweist.*
DataType	*Zeit*
Domain	*Jede gültige Zeit.*

473 Vgl. [CDIF94f], Seite 19.

Name des Attributs	Bedeutung
Length	–
IsOptional	*True*

Tabelle 4-6: Meta-Attribut „TimeUpdated“

4.1.1.1.2 Meta-Entitätstyp „RootEntity“

Der Meta-Entitätstyp „RootEntity“[474] bildet die Wurzel für die Meta-Entitätstypen eines Meta-Modells und erbt aufgrund der Generalisierungshierarchie sämtliche Meta-Attribute von „RootObject“. Wie eingangs bereits erwähnt, haben sämtliche Subtypen in den standardisierten EIA/CDIF-Meta-Modellen, die von „RootEntity“ abgeleitet werden, den Meta-Meta-Entitätstyp „MetaEntity“ als ihren Meta-Typen.[475] In der graphischen Darstellung wird für diesen Meta-Entitätstyp und all seinen Subtypen das Rechteck benutzt. Tabelle 4-7 beschreibt den Meta-Entitätstyp „RootEntity“.

Name des Attributs	Bedeutung
Name	***RootEntity***
CDIFMetaIdentifier	***2***
SubtypeOf	*RootObject*
SupertypeOf	–
Description	*Dies ist der Supertyp von allen Meta-Entitätstypen des Integrierten EIA/CDIF-Meta-Modells und all seiner Erweiterungen.*
Usage	„RootEntity“ sollte nicht direkt verwendet werden, stattdessen sollten genauer definierte Subtypen[476] definiert werden.
Aliases	–

474 Vgl. [CDIF94f], Seite 20.

475 Vgl. in diesem Zusammenhang auch die Ausführungen in Fußnote 461 auf Seite 156 und ihren umgebenden Text.

476 Im Standard [CDIF94f] auf Seite 20 wird fälschlicherweise von „Instanzen“ statt von „Subtypen“ gesprochen.

Name des Attributs	Bedeutung
Constraints	*Es dürfen diesem Meta-Entitätstyp durch die Erweiterung im Rahmen eines EIA/CDIF-Austausches keine weiteren Supertypen vorangestellt werden.*
Type	*Kernel*

Tabelle 4-7: Meta-Entitätstyp „RootEntity“

4.1.1.1.3 Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“

Der binäre Meta-Beziehungstyp „IsRelatedTo“ mit dem vollqualifizierten Namen „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“[477] bildet die Wurzel für die Meta-Beziehungstypen eines Meta-Modells und erbt aufgrund der Generalisierungshierarchie sämtliche Meta-Attribute von „RootObject“. Wie eingangs bereits erwähnt, haben sämtliche Subtypen in den standardisierten EIA/CDIF-Meta-Modellen, die von „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ abgeleitet werden, den Meta-Meta-Entitätstyp „MetaRelationship“ als ihren Meta-Typen.[478]

In der graphischen Darstellung wird für diesen Meta-Beziehungsstyp und all seinen Subtypen ein Pfeil benutzt, der den Quell-Meta-Entitätstyp mit dem Ziel-Meta-Entitätstyp verbindet und die Angabe von Kardinalitäten in der Minium-/ Maximum-Notation erlaubt.

Der Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ erlaubt aufgrund der Kardinalitäten „0:N“ zu „0:N“ die uneingeschränkte Zuordnung von Instanzen vom Typ „RootEntity“ zu Instanzen vom Typ “RootEntity“. Aufgrund der Vererbungseigenschaft entlang der Generalisierungshierarchie können damit sämtliche Subtypen von „RootEntity“ über den Beziehungstyp „IsRelatedTo“ oder seinen Subtypen miteinander in Beziehung gesetzt werden.

Die folgende Tabelle 4-8 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „IsRelatedTo“.

---

477 Vgl. [CDIF94f], Seite 20.

478 Vgl. in diesem Zusammenhang auch die Ausführungen in Fußnote 461 auf Seite 156 und ihren umgebenden Text.

Name des Attributs	Bedeutung
Name	***IsRelatedTo***
SubtypeOf	*RootObject*
CDIFMetaIdentifier	***3***
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*N*
Description	*Dies ist der Supertyp von allen Meta-Beziehungstypen des Integrierten EIA/CDIF-Meta-Modells und all seiner Erweiterungen.*
Usage	„RootEntity.*IsRelatedTo*.RootEntity“ sollte nicht direkt verwendet werden, stattdessen sollten genauer definierte Subtypen[479] definiert werden.
Aliases	–
Constraints	*Es dürfen diesem Meta-Beziehungstyp durch die Erweiterung im Rahmen eines EIA/CDIF-Austausches keine weiteren Supertypen vorangestellt werden.*

Tabelle 4-8: Meta- Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“

4.1.1.2 Auswertungen der Definitionen

Zunächst wird in diesem Abschnitt eine strukturelle Übersicht über das Meta-Modell gegeben, indem einfache Auswertungen tabellarisch aufbereitet werden. Daran anschließend werden sämtliche AttribuierbarenMetaObjekte, die ausdrücklich im Meta-Modell benutzt werden,[480] in der dem Meta-Modell entsprechenden Generalisierungshierarchie dargestellt, wobei MetaObjekte auf derselben Stufe alphabetisch sortiert sind. Abschließend werden jene Attribuierba-

479 Im Standard [CDIF94f] auf Seite 20 wird fälschlicherweise von „Instanzen“ statt von „Subtypen“ gesprochen.

480 Es handelt sich also um Instanzen jener SammelbarenMetaObjekte, die in Instanzen des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ für die Instanz des entsprechenden GegenstandBereiches enthalten sind.

renMetaObjekte in alphabetischer Reihenfolge summarisch angeführt, für die ausdrücklich[481] MetaAttribute definiert sind.

Sämtliche Auflistungen in diesem Abschnitt erfolgen anhand der voll qualifizierten Namen der AttribuierbarenMetaObjekte, wobei die Namen von Meta-Beziehungstypen immer kursiv gesetzt sind. Die Namen von Meta-Entitätstypen, deren Instanzen zwingend in Beziehungen auftreten müssen,[482] werden immer fett hervorgehoben. Die Kardinalitäten der Meta-Beziehungstypen werden ausdrücklich angeführt, sowie die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“[483]. Sofern für AttribuierbareMetaObjekte die Mehrfachvererbung vorgesehen ist, wird der gesamte vollqualifizierte Namen kursiv gesetzt. Sämtliche lokal definierten Meta-Attribute werden angeführt, wobei zusätzlich jeweils ihr EIA/CDIF-Datentyp in kleiner Schrift mitangegeben ist.[484]

4.1.1.2.1 Strukturelle Übersicht

Das fundamentale Meta-Modell „Foundation“ definiert insgesamt acht SammelbareMetaObjekte, wobei es als einziges Meta-Modell eine direkte Instanz[485] von dem Meta-Meta-Entitätstyp „AttribuierbaresMetaObjekt“ bildet, nämlich „RootObject“. Tabelle 4-9 stellt die SammelbarenMetaObjekte tabellarisch dar.

---

481 Derartige Meta-Attribute werden in den EIA/CDIF-Standards auch als „lokal“ bezeichnet, um sie damit von über die Generalisierungshierarchie ererbten Meta-Attributen unterscheidbar zu machen.

482 Es handelt sich hierbei um jene MetaEntitäten, die einer Kardinalität mit einem Wert größer Null im Minimum entgegengesetzt angeführt sind.

483 Die Darstellung erfolgt in der vorgesehenen „ENCODING.1“-Verkodierung des EIA/CDIF-Datentyps „Identifier“, indem der Wert in Sterne eingefaßt wird (vgl. [CDIF94e], Seiten 11 und 20 unten, „IdentifierValue“).

484 Im Falle des aufzählbaren Datentyps werden auch die gültigen Werte der entsprechenden Wertemenge dargestellt.

485 Nachdem alle anderen Meta-Objekte „Foundation“ einsetzen müssen, entfällt für diese die Notwendigkeit, eine Wurzel vom Typ „AttributableMetaObject“ zu definieren.

Anzahl direkter Instanzen vom Typ AMO[486]	1
Anzahl Instanzen vom Typ ME	1
Anzahl Instanzen vom Typ MR	1
Anzahl Instanzen vom Typ MA	5
Anzahl (Summe) Instanzen vom Typ CMO	8

Tabelle 4-9: Verteilung der SammelbarenMetaObjekte des fundamentalen Meta-Modells

Tabelle 4-10 führt weitere Analyseergebnisse über die strukturellen Eigenschaften dieses Meta-Modells an.

Anzahl zwingend vorgeschriebener MA		1
Anzahl optionaler MA		4
Meta-Objekte mit/ohne Meta-Attribute	mit MA	ohne MA
Anzahl Instanzen vom Typ AMO	1	0
Anzahl Instanzen vom Typ ME	0	1
Anzahl Instanzen vom Typ MR	0	1
Anzahl von zwingend vorgeschriebenen MR[487]		0
Anzahl von ME, die an zwingend vorgeschriebenen MR teilhaben		0
Anzahl der Blätter (Instanzen vom Typ AMO) im Meta-Modellbaum		2
Anzahl von AMOs mit Mehrfachvererbung		0

Tabelle 4-10: Strukturelle Übersicht über das fundamentale Meta-Modell

Das zwingend vorgeschriebene Meta-Attribut in diesem GegenstandsBereich heißt „CDIFIdentifier“ und ist für das AttribuierbareMetaObjekt „RootObject“ definiert. Nachdem „RootObject“ die Wurzel eines jeden EIA/CDIF-Meta-Modells bildet, ist dieses Surrogat daher über die Vererbung entlang der Gene-

486 Hier wird die einzige direkte Instanz vom Meta-Meta-Entitätstyp „AttribuierbaresMeta-Objekt“ angeführt, nämlich „RootObject“.

487 Zwingend vorgeschriebene Meta-Beziehungstypen verfügen über einen Wert >=1 in mindestens einer der minimalen Kardinalitäten. Aufgrund der gegenüberliegenden Anschrift der Kardinalitäten im graphischen Modell bedeutet dies, daß ein Wert >=1 im Meta-Meta-Attribut „Min*Source*Card“ zur Folge hat, daß Instanzen des *Ziel*-Meta-Entitätstyps an Instanzen dieses Meta-Beziehungstyps teilnehmen müssen. Entsprechend müssen bei einem Wert >=1 im Meta-Meta-Attribut „Min*Dest*Card“ Instanzen des *Quell*-Meta-Entitätstyps an Instanzen des Meta-Beziehungstyps enthalten sein.

ralisierungshierarchie für jeden Meta-Enitätstyp und Meta-Beziehungstyp verfügbar.

4.1.1.2.2 Aufgefundene Fehler

Die im Rahmen dieser Arbeit erfolgte Analyse der Definitionen in bezug auf die korrekte Nutzung der EIA/CDIF-Datentypen für die Meta-Meta-Attribute sowie mit dem Rahmen, den das Meta-Meta-Modell vorgibt, ergaben für das fundamentale EIA/CDIF-Meta-Modell keine Fehler.

4.1.1.2.3 Generalisierungshierarchie des Meta-Modells

Die folgende Abbildung 4-2 stellt die Generalisierungshierarchie dieses Meta-Modells dar. EIA/CDIF-Meta-Modelle erweitern diese, indem sie den Meta-Entitätstyp „RootEntity“ und den Meta-Beziehungstyp „RootEntity.*IsRelatedTo*.-RootEntity“ spezialisieren. Die maximale Höhe des Generalisierungsbaumes ist zwei Ebenen hoch. Der Teilbaum für Meta-Entitätstypen stellt sich als gleich spezialisiert dar wie der für Meta-Beziehungstypen.

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

Abbildung 4-2: Generalisierungshierarchie des Meta-Modells „Foundation“

4.1.1.2.4 Summarische Darstellung der AttribuierbarenMetaObjekte

Die summarische Darstellung der Meta-Entitätstypdefinitionen entspricht dem Kapitel 4.4, „MetaEntity Summary“, eines jeden EIA/CDIF-Standards, die der Meta-Beziehungstypdefinitionen dem Kapitel 4.5, „MetaRelationship Summary“. Hierbei erfolgt die Reihung in der alphabetisch sortierten Reihenfolge der qualifizierten Namen der AttribuierbarenMetaObjekte. In dieser Darstellung wird summarisch gezeigt, welche Meta-Attribute über die Generalisierungshierarchie ererbt und welche lokal definiert wurden. Ererbte Meta-Attribute werden hierbei – wie in den EIA/CDIF-Standards üblich – kursiv dargestellt, lokale mit normaler Schriftauszeichnung.

Im Unterschied zu den EIA/CDIF-Standards ist in der Aufstellung zusätzlich angegeben, von welchen übergeordneten Typen die ererbten Meta-Attribute

stammen und welche Werte[488] für das entsprechende Meta-Meta-Attribut „CDIFMetaIdentifier“ vorgesehen sind. Des weiteren werden die EIA/CDIF-Datentypen für die lokal definierten Meta-Attribute angeführt, im Falle von zwingend vorgeschriebenen Meta-Attributen werden diese lokal fett gesetzt, um sie besonders hervorzuheben.

Sämtliche Meta-Beziehungstypen werden gemeinsam mit den für sie definierten Kardinalitäten dargestellt. Sofern in einem Meta-Beziehungstyp Meta-Entitätstypen zwingend[489] teilhaben müssen, werden die betroffenen Meta-Entitätstypen in allen Darstellungen fett dargestellt.

Somit erlauben es die folgenden Tabellen, auf einen Blick die wesentlichsten Definitionen zu erkennen.

4.1.1.2.4.1 Summarische Definition der direkten Instanz vom Meta-Meta-Entitätstyp „AttribuierbaresMetaObjekt“

Tabelle 4-11 stellt die einzige Instanz des Meta-Meta-Entitätstyps „Attributable-MetaObject“ dar.

RootObject *1* (AMO)	
CDIFIdentifier *5* – Identifier	**zwingend** [lokal]
DateCreated *6* – Date	optional
DateUpdated *7* – Date	optional
TimeCreated *8* – Time	optional
TimeUpdated *9* – Time	optional

Tabelle 4-11: Summarische Darstellung von „RootObject“

4.1.1.2.4.2 Summarische Definition der direkten Instanzen vom Meta-Meta-Entitätstyp „MetaEntity“

Tabelle 4-12 stellt die erste Instanz des Meta-Meta-Entitätstyps „MetaEntity“ dar und bildet zugleich den Wurzeltyp für sämtliche Meta-Entitätstypen.

---

488 Diese Surrogatwerte werden in kleiner Schriftgröße angegeben, wobei die Werte entsprechend [CDIF94e], „ENCODING.1“, in Sternen eingeschlossen sind.

489 Der Minimalwert der gegenüberliegenden Kardinalität ist in einem solchen Fall größer Null.

RootEntity *2* (ME)		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

Tabelle 4-12: Summarische Darstellung des Meta-Entitätstyps „RootEntity“

4.1.1.2.4.3 Summarische Definition der direkten Instanzen vom Meta-Meta-Entitätstyp „MetaRelationship“

Tabelle 4-13 stellt die erste Instanz des Meta-Meta-Entitätstyps „Meta-Relationship“ dar und bildet zugleich den Wurzeltyp für sämtliche Meta-Beziehungstypen.

0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3* (MR)		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

Tabelle 4-13: Summarische Darstellung des Meta-Beziehungstyps „RootEntity.-*IsRelatedTo*.RootEntity“

4.1.1.3 OMG IDL-Definitionen für EIA/CDIF-Meta-Modelle[490]

Grundsätzlich gelten für die OMG IDL-Definitionen für EIA/CDIF-Meta-Modelle sämtliche Festlegungen, die in Abschnitt 3.2, „Verteilung des EIA/CDIF-Meta-Meta-Modells mit Hilfe von CORBA“ auf Seite 98ff, insbesondere für das OMG IDL Modul „CDIF“[491] angegeben sind, sowie für die Datentypen[492] und die Ausnahmebedingungen[493].

Für jedes Meta-Modell wird ein eigener Namensraum in Form eines Moduls angelegt, in dem entsprechend der Architektur von EIA/CDIF die OMG IDL Definitionen für das fundamentale Meta-Modell spezialisiert werden. Somit erben sämtliche Subtypen indirekt die Meta-Attribute von „RootObject“ in Form der dafür definierten Schnittstellen, indem einerseits der Meta-Entitätstyp „RootEntity“ und andererseits der Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ spezialisert werden.[494]

Grundsätzlich sieht die Struktur der OMG IDL-Module für die Abbildung von Meta-Modellen wie folgt aus:[495]

```
module <Meta_Modell__SubjectArea> {
     <Interface-Definitionen für ME 1>
     <Interface-Definitionen für ME 2>
...
     <Interface-Definitionen für MR 1>
     <Interface-Definitionen für MR 2>
...
 };
```

Abbildung 4-3: OMG IDL-Moduldefinitionen für EIA/CDIF-Meta-Modelle

OMG IDL-Schnittstellendefinitionen für standardisierte EIA/CDIF-Meta-Modelle werden im EIA/CDIF-Modul „CDIF“ angegeben, Erweiterungen zu Meta-

---

490 Vgl. in diesem Zusammenhang [CDIF97b], Seiten 14 bis 20, sowie Seite 21ff.

491 Vgl. Abschnitt 3.2.3, Seite 101ff weiter oben.

492 Vgl. Abschnitt 3.2.1, „EIA/CDIF-Datentypdefinitionen in OMG IDL“ auf Seite 98ff weiter oben.

493 Vgl. Abschnitt 3.2.2, „EIA/CDIF-Ausnahmendefinitionen in OMG IDL“ auf Seite 100ff weiter oben.

494 Entsprechend muß das in diesem Abschnitt vorgestellte OMG IDL Modul „Foundation_01_00“ von allen anderen Meta-Modell-Moduldefinitionen referenziert werden.

495 Vgl. [CDIF97b], Seite 17 unten.

Modellen beziehungsweise Schnittstellendefinitionen für eigenständige Meta-Modelle außerhalb.[496] Sämtliche Definitionen für Meta-Modelle erlauben grundsätzlich das Abfragen und Ändern von Meta-Attributwerten.

Abbildung 4-4 stellt die Schnittstellendefinitionen in der OMG IDL für das fundamentale EIA/CDIF-Meta-Modell dar.[497, 498]

```
[1]  module Foundation_01_00 {
          interface RootEntity;
          interface RootEntity_IsRelatedTo_RootEntity;
          typedef sequence<RootEntity> RootEntity_Set;
          typedef sequence<RootEntity_IsRelatedTo_RootEntity>
          RootEntity_IsRelatedTo_RootEntity_Set;
[2]       interface RootObject {
            AnyDT getCDIFIdentifier();
            void setCDIFIdentifier( in AnyDT aValue )
                 raises( IllegalDataType, ValueMustBePresent );
            AnyDT getDateCreated();
            void setDateCreated( in AnyDT aValue )
                 raises( IllegalDataType );
            AnyDT getDateUpdated();
            void setDateUpdated( in AnyDT aValue )
                 raises( IllegalDataType );
            AnyDT getTimeCreated();
            void setTimeCreated( in AnyDT aValue )
                 raises( IllegalDataType );
            AnyDT getTimeUpdated();
            void setTimeUpdated( in AnyDT aValue )
                 raises( IllegalDataType );
[3]         MetaMetaModel_02_00::AttributableMetaObject
                 get_attributableMetaObject();
[4]         boolean is_identical( in RootObject otherObject );
[5]         AnyDT get_metaAttributeValue(
                 in MetaMetaModel_02_00::MetaAttribute aMetaAttribute )
              raises( IllegalAccess );
[6]         void set_metaAttributeValue(
                 in MetaMetaModel_02_00::MetaAttribute aMetaAttribute,
```

496 Dies hat zur Folge, daß EIA/CDIF-standardisierte Meta-Modelle beispielsweise den Meta-Entitätstyp „RootEntity“ mit „Foundation_01_00::RootEntity“ referenzieren können, wohingegen alle anderen die Referenz „***CDIF::***Foundation_01_00::RootEntity“ benötigen.

497 Die OMG IDL-Definitionen für das Meta-Modell „Foundation“ werden innerhalb des Moduls „CDIF“ angegeben, wie dies auch in **Abbildung 3-3**, „OMG IDL – Geltungsbereiche der Moduldefinitionen“ auf Seite 102 schematisch dargestellt ist.

498 Die Numerierung ist nicht Teil der Schnittstellendefinition, sondern wird weiter unten zur eindeutigen Referenzierung der besprochenen Festlegungen benutzt.

```
                in AnyDT aValue )
              raises( IllegalAccess, IllegalDataType,
                    ValueMustBePresent );
        };
[7]     interface RootEntity : RootObject {
[8]       MetaMetaModel_02_00::MetaEntity get_metaEntity();
[9]       RootEntity_Set traverseSrcToDest(
            in MetaMetaModel_02_00::MetaRelationship
              aMetaRelationship )
            raises( IllegalTraversal );
[10]      RootEntity_Set traverseDestToSrc(
            in MetaMetaModel_02_00::MetaRelationship
              aMetaRelationship )
            raises( IllegalTraversal );
[11]      RootEntity_IsRelatedTo_RootEntity_Set
            traverseSrcToRelationship(
              in MetaMetaModel_02_00::MetaRelationship
                   aMetaRelationship )
              raises( IllegalTraversal );
[12]      RootEntity_IsRelatedTo_RootEntity_Set
             traverseDestToRelationship(
              in MetaMetaModel_02_00::MetaRelationship
                   aMetaRelationship )
              raises( IllegalTraversal );
        };
[13]    interface RootEntity_IsRelatedTo_RootEntity : RootObject {
[14]      MetaMetaModel_02_00::MetaRelationship
            get_metaRelationship();
[15]      RootEntity traverseToDest();
[16]      RootEntity traverseToSrc();
        };
    };
```

Abbildung 4-4: OMG IDL-Definitionen für das fundamentale EIA/CDIF-Meta-Modell[499]

Für den lesenden und schreibenden Zugriff auf Meta-Attribute sind die entsprechenden „get“- und „set“-Schnittstellen definiert, wobei für Argumente als Datentyp ausschließlich „AnyDT“[500] benutzt wird. Zusätzlich finden sich in den OMG IDL Definitionen der Abbildung 4-4 folgende Schnittstellen:

499 Vgl. [CDIF97b], Seite 15-17.

500 Vgl. die entsprechenden OMG/IDL-Datentypdefinitionen von EIA/CDIF im Abschnitt 3.2, „Verteilung des EIA/CDIF-Meta-Meta-Modells mit Hilfe von CORBA“ auf Seite 98ff weiter oben.

1. *„get_attributableMetaObject()“* in Nummer [3]
   Diese Funktion liefert als Ergebnis den Metatyp des Objekts „RootObject“. Es handelt sich hierbei um eine Instanz, die den Meta-Meta-Entitätstyp „AttributableMetaObject“ repräsentiert.

2. *„is_identical()“* in Nummer [4]
   Diese Funktion liefert den Boole'schen Wert *True*, wenn der Wert im Meta-Attribut *„CDIFIdentifier“* mit dem Wert im Meta-Attribut *„CDIFIdentifier“* des Arguments übereinstimmt, *False* sonst.[501]

3. *„get_metaAttributeValue()“* in Nummer [5]
   Diese Funktion liefert den Wert, den das als Argument übergebene Meta-Attribut aufweist.

4. *„set_metaAttributeValue()“* in Nummer [6]
   Diese Funktion setzt den als zweites Argument übergebenen Wert für das als erstes Argument übergebene Meta-Attribut.

5. *„get_metaEntity()“* in Nummer [8]
   Diese Funktion liefert als Ergebnis den Metatyp des Objekts „RootEntity“. Es handelt sich hierbei um eine Instanz, die den Meta-Meta-Entitätstyp „MetaEntity“ repräsentiert.

6. *„traverseSrcToDest()“* in Nummer [9]
   Liefert sämtliche Instanzen vom Zieltyp „RootEntity“, die über den als Argument angegebenen Meta-Beziehungstyp mit der aktuellen Instanz[502] vom Quelltyp „RootEntity“ in Beziehung stehen. Es handelt sich dementsprechend um eine Menge von Instanzen vom (Sub-) Typ „RootEntity“.

7. *„traverseDestToSrc()“* in Nummer [10]
   Liefert sämtliche Instanzen vom Quelltyp „RootEntity“, die über den als Argument angegebenen Meta-Beziehungstyp mit der aktuellen Instanz

---

501 Es ist in diesem Zusammenhang wichtig, daß die Identität zweier CORBA-Objekte nicht aufgrund der ORB-Referenzen festgestellt wird, sondern der Test aufgrund der Werte erfolgt, die die Implementierungen dem Meta-Attribut „CDIFIdentifier“ zugewiesen haben. Somit können verschiedene, verteilte CORBA-Objekte existieren, die aufgrund dieser Eigenschaft *identische* Konzepte repräsentieren. Vgl. hierzu auch die Ausführungen in [CDIF97b], Abschnitt „5.2 Object Identity“ auf Seite 21.

502 Der Begriff „aktuelle Instanz“ (englisch: „current instance“) wird in [CDIF97b], Seite 22, eingeführt und bezeichnet den Empfänger (das CORBA-Objekt) der Nachricht. In diesem Kontext ist es beispielsweise eine Instanz vom (Sub-) Typ „RootEntity“.

vom Zieltyp „RootEntity“ in Beziehung stehen. Es handelt sich dementsprechend um eine Menge von Instanzen vom (Sub-) Typ „RootEntity“.

8. *„traverseSrcToRelationship()“* in Nummer [11]
   Liefert sämtliche Instanzen von dem als Argument angegebenen Meta-Beziehungstyp, die die aktuelle Instanz vom Quelltyp „RootEntity“ mit Instanzen vom Zieltyp „RootEntity“ verbindet. Es handelt sich entsprechend um eine Menge von Instanzen vom (Sub-) Typ „RootEntity.*IsRelatedTo*.-RootEntity“.

9. *„traverseDestToRelationship()“* in Nummer [12]
   Liefert sämtliche Instanzen von dem als Argument angegebenen Meta-Beziehungstyp, die die aktuelle Instanz vom Zieltyp „RootEntity“ mit Instanzen vom Quelltyp „RootEntity“ verbindet. Es handelt sich entsprechend um eine Menge von Instanzen vom (Sub-) Typ „RootEntity.-*IsRelatedTo*.RootEntity“.

10. *„get_metaRelationship()“* in Nummer [14]
    Diese Funktion liefert als Ergebnis den Metatyp des Objekts „RootEntity.-*IsRelatedTo*.RootEntity“. Es handelt sich hierbei um eine Instanz, die den Meta-Meta-Entitätstyp „MetaRelationship“ repräsentiert.

11. *„traverseToDest()“* in Nummer [15]
    Liefert die Instanz vom Typ „RootEntity“, die für die Instanz vom Typ „RootEntity.*IsRelatedTo*.RootEntity“ das Ziel bildet.

12. *„traverseToSrc()“* in Nummer [16]
    Liefert die Instanz vom Typ „RootEntity“, die für die Instanz vom Typ „RootEntity.*IsRelatedTo*.RootEntity“ die Quelle bildet.

4.1.2 CMMN – Das EIA/CDIF-Meta-Modell „Common“

[CDIF95] beinhaltet die Definitionen für das standardisierte EIA/CDIF-Meta-Modell „Common“[503] (deutsch: „Allgemein“). In diesem Gegenstandsbereich wurden jene Konzepte festgelegt, die mehreren Gegenstandsbereichen gemein[504] sind beziehungsweise für die Erzeugung von EIA/CDIF-Transfers[505] als allgemein nutzbringend erscheinen.

Die Konzepte des allgemeinen GegenstandsBereiches sind in Abbildung 4-5 auf Seite 179 weiter unten graphisch dargestellt. Im folgenden werden die darin dargestellten MetaEntitäten und MetaBeziehungen kurz charakterisiert:

1. Der Meta-Entitätstyp „AlternateName“ erlaubt es, Instanzen von Meta-Entitätstypen Aliasnamen mit Hilfe des Meta-Beziehungstyps „1:1 Root-Entity.*Has.***AlternateName** 0:N“ zuzuordnen. In weiterer Folge ist es mit Hilfe des Meta-Beziehungstyps „0:N Root.Entity.*Uses.*AlternateName 0:1“ möglich, jenen Aliasnamen zu bestimmen, der in den Modelldaten benutzt wird.

2. Der Meta-Entitätstyp „ToolUser“ erlaubt die Definition von Benutzern von Werkzeugen, mit denen Modelle erstellt wurden. Es definiert optionale Meta-Attribute zur Aufnahme der vollen, zivilen Namen. Gemeinsam mit den Meta-Beziehungstypen „0:N RootEntity.*CreatedBy*.ToolUser 0:1“ und „0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1“ können Instanzen vom (Sub-) Typ „RootEntity“ jenen Benutzern zugeordnet werden, die sie erstellt oder zuletzt geändert haben.

3. Der Meta-Entitätstyp „PresentationInformationObject“ soll als Ausgangspunkt für jene Meta-Entitätstypdefinitionen dienen, die in Meta-Modellen für den Transport von graphischen Modelldaten gedacht sind.

4. Der Meta-Entitätstyp „SemanticInformationObject“ ist als Ausgangspunkt für jene Meta-Entitätstypdefinitionen gedacht, die in Meta-Modellen Kon-

---

503 Dieser GegenstandsBereich wurde während seiner Entwicklung vom EIA/CDIF-Komitee mit der Abkürzung „CMMN“ bezeichnet.

504 Der Meta-Entitätstyp „SemanticInformationObject“ wird mit Ausnahme des fundamentalen Meta-Modells in praktisch allen anderen Meta-Modellen eingesetzt.

505 In diesem Zusammenhang erlaubt beispielsweise der Meta-Entitätstyp „ToolUser“ weiter unten, Audit-Daten über die Modelldaten des EIA/CDIF-Transfers mitzutauschen.

zepte repräsentieren, die eine eigenständige Bedeutung im Sinne der Modellierung mit bestimmten Methodologien aufweisen. Damit stellen diese selbst Konzepte dar, mit denen Modelldaten erzeugt beziehungsweise strukturiert werden. Gemeinsam mit den Meta-Beziehungstypen „0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject 1:N“ und „0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N“ können zudem Einschränkungen für und Kategorisierungen von Instanzen dieses Typs erfolgen.

Die Meta-Entitätstypen „Derivation“, „DataObject“ und „ProcessObject“ sind Subtypen von „SemanticInformationObject“ und erben aufgrund der Generalisierungshierarchie auch dessen Meta-Beziehungstypen.

5. Der MetaEntitätstyp „AbstractionLevel“ erlaubt es, Instanzen vom Typ „SemanticInformationObject“ mit Hilfe des Meta-Beziehungstyps „0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N“ in „konzeptionelle“, „logische“ und „physische“ Modelldaten zu klassifizieren.

6. Der Meta-Entitätstyp „TextualConstraint“ ermöglicht die Angabe von textuellen Einschränkungen entweder in freier Sprache oder mit Hilfe von Programmiersprachen.[506]

7. Der Meta-Entitätstyp „ProcessObject“ soll als Ausgangspunkt für jene Meta-Entitätstypdefinitionen dienen, die in Meta-Modellen für den Transport von prozeßbezogenen Modelldaten gedacht sind.[507]

8. Der Meta-Entitätstyp „DataObject“ wurde ursprünglich[508] als Supertyp für jene Typen vorgesehen, die in irgendeiner Form Datenobjekte repräsentie-

---

506 [CDIF95] legt in Abschnitt „4.1.11 Computable Languages“ auf Seite 18 die folgenden von ISO oder ANSI definierten Sprachen in Form von aufzählbaren Werten (englisch: „enumerated values“) fest: „Ada“, „C“, „COBOL“, „FORTRAN“, „MUMPS“, „PASCAL“, „PL1“ und „SQL“.
Der Wert „Other“ weist darauf hin, daß eine andere als hier angeführte Programmiersprache für die Spezifizierung benutzt wird. Sofern in einer Programmiersprache spezifiziert wird, die hier nicht angeführt ist, empfiehlt es sich, mit Hilfe des Erweiterungsmechanismus' die gewählte(n) Programmiersprache(n) in die Wertemenge aufzunehmen und anschließend zu referenzieren. Dadurch wird anhand des aufzählbaren Wertes selbst sofort klar, um welche Programmiersprache es sich dabei handelt. Es wäre wünschenswert, würde der aufgezählte Wert „Other“ in weiterer Folge nur für das Festlegen von Einschränkungen oder Algorithmen in natürlicher Sprache benutzt werden.

507 Aufgrund der Auswertungen sämtlicher standardisierter und per Anfang Jänner 1998 in Entwurf befindlichen EIA/CDIF-Meta-Modelle kann festgestellt werden, daß dieser Subtyp sonst überhaupt nicht verwendet wird.

ren und daher durch Attribute beschrieben werden, deren Datentypen unter anderem auch durch die Konzepte des GegenstandsBereichs „Data Definition Subject Area“ definiert werden können.[509]

9. Der MetaEntitätstyp „Derivation“ soll gemeinsam mit den Meta-Beziehungstypen „1:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N“ und „1:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N“ sowie in Kombination mit dem vom Supertyp „SemanticInformationObject“ ererbten Meta-Beziehungstypen „0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N“ unterschiedliche Formen von Ableitungen erlauben.[510]

Trotzdem feststellbar ist, daß manche Subtypen dieses GegenstandsBereiches in EIA/CDIF-eigenen Meta-Modellen bis dato nicht eingesetzt wurden, sollen die hier in Form von Meta-Entitätstypen und Meta-Beziehungstypen definierten Konzepte aufgrund ihrer theoretisch vielfältigen Einsetzbarkeit in weiteren Meta-Modellen im Detail vorgestellt werden.[511]

---

508 Vgl. [CDIF95], Abschnitt 4.1.10, „Data Object Content“ auf Seite 17 unten.

509 Tatsächlich konnte aufgrund der im Rahmen dieser Arbeit erfolgten Auswertungen festgestellt werden, daß der Meta-Entitätstyp „DataObject“ sonst überhaupt nicht verwendet wird.

510 Vgl. zu diesen Ausführungen auch [CDIF95], Abschnitt 4.1.8, „Derivation“ auf Seite 17.
Aufgrund der Auswertungen sämtlicher standardisierter und per Anfang Jänner 1998 in Entwurf befindlichen EIA/CDIF-Meta-Modelle kann festgestellt werden, daß dieser Subtyp sonst überhaupt nicht verwendet wird.
Es scheint, als wäre im Zusammenhang mit der Definition der Meta-Modelle für die GegenstandsBereiche „Data Modeling“ ([CDIF96b]) und „Data Flow“ ([CDIF96c]) mit dem „General Structuring Mechanism“ (vgl. zum Beispiel [CDIF96c] Seiten 17 bis 20 beziehungsweise weiter unten im Abschnitt 4.1.3.1.2, „Der Allgemeine Strukturierungsmechanismus“ auf Seite 229ff) eine alternative Modellierungsmöglichkeit gefunden worden, die die Benutzung dieses Konzepts in allen anderen Meta-Modellen möglicherweise hinfällig macht.

511 Dies ist im Rahmen dieser Arbeit auch deswegen möglich, da „Common“ mit 45 SammelbarenMetaObjekten im Vergleich zu den Meta-Modellen für die GegenstandsBereiche „Data Flow Modeling“ (84 SammelbareMetaObjekte) und „Data Modeling“ (146 SammelbareMetaObjekte) über wenige Typen aufweist.

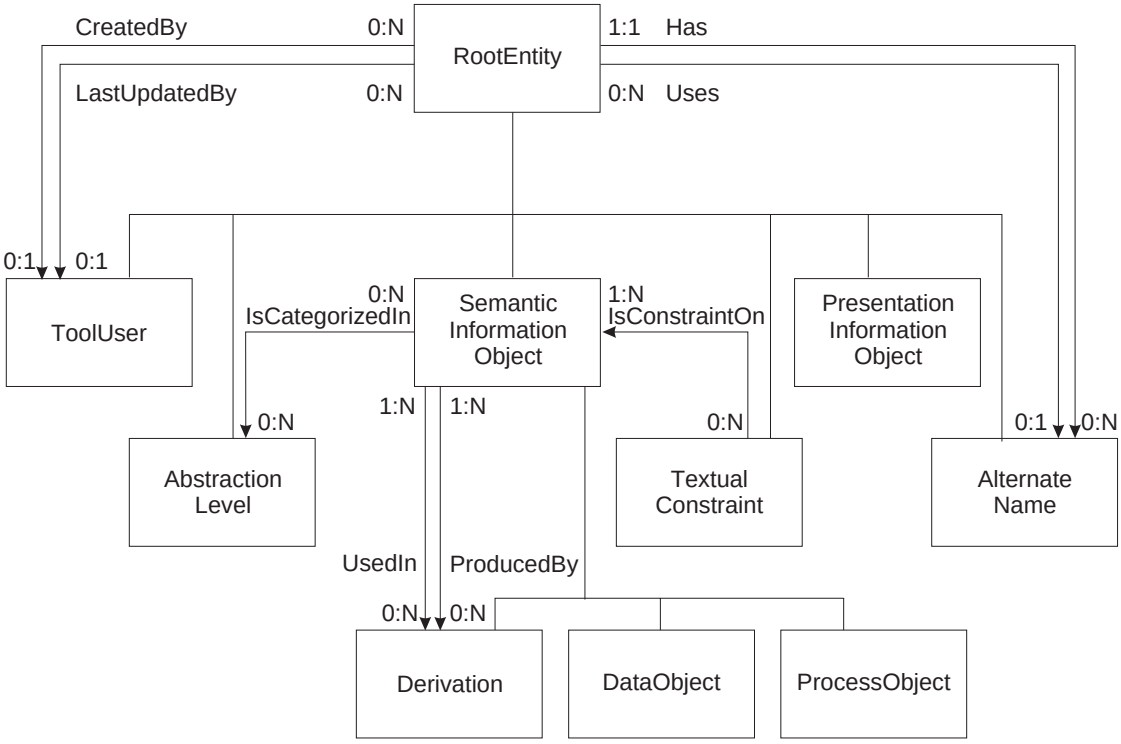


Abbildung 4-5: Das standardisierte EIA/CDIF-Meta-Modell „Common“[512]

4.1.2.1 Definitionen der „allgemeinen“ Konzepte

Die Definitionen für die MetaObjekte des standardisierten EIA/CDIF-Meta-Modells „Common“ erfolgen im Unterschied zur obigen Darstellung von „Foundation“ in alphabetischer Reihenfolge, getrennt nach Meta-Entitätstypen und Meta-Beziehungstypen.[513]

4.1.2.1.1 Meta-Entitätstyp „AbstractionLevel“

Der Meta-Entitätstyp „AbstractionLevel“[514, 515] erlaubt die Zuordnung von Entitäten zu unterschiedlichen Abstraktionsebenen in einem Modelldatenaustausch. Sie erfolgt mit Hilfe des Meta-Beziehungstyps „0:N SemanticInformationObject.-*IsCategorizedIn.*AbstractionLevel 0:N“. Tabelle 4-14 beschreibt den Meta-Entitätstyp „AbstractionLevel“.

---

512 Vgl. [CDIF95], Seite 19.

513 Es handelt sich hierbei gleichzeitig um jene Darstellungsvariante, die EIA/CDIF für die Abfassung ihrer eigenen Standards verwendet.

514 Vgl. [CDIF95], Seite 28.

515 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt.

Name des Attributs	Bedeutung
Name	***AbstractionLevel***
CDIFMetaIdentifier	***12***
SubtypeOf	*RootEntity*
SupertypeOf	–
Description	*Repräsentiert eine Abstraktionsebene.*
Usage	*Instanzen dieses Meta-Entitätstyps repräsentieren unterschiedliche Abstraktionsniveaus der Modellierung, auf die Instanzen anderer Meta-Entitätstypen über den Meta-Beziehungstyp „Semantic-InformationObject.IsCategorizedIn.AbstractionLevel“ bezug nehmen können.*
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 4-14: Meta-Entitätstyp „AbstractionLevel“

Der Meta-Entitätstyp „AbstractionLevel“ verfügt über ein einziges, zwingend vorgeschriebenes Meta-Attribut „Name“.[516]

4.1.2.1.1.1 Meta-Attribut „Name“

Das Meta-Attribut „Name“ vom EIA/CDIF-Datentyp „Enumerated“ weist eines der drei aufzählbaren Werte „Conceptual“, „Logical“ und „Physical“ auf.[517]

Der Meta-Entitätstyp „AbstractionLevel“ wird aufgrund der Kardinalitäten des Meta-Beziehungstyps „0:N SemanticInformationObject.*IsCategorizedIn.*AbstractionLevel 0:N“ und aufgrund der Tatsache, daß es nur über ein Meta-Attribut vom EIA/CDIF-Datentyp „Enumerated“ mit maximal drei aufzählbaren Werten

516 Im Rahmen der Standardisierung von EIA/CDIF innerhalb von ISO/IEC wird dieses Meta-Attribut voraussichtlich zu einem optionalen und der Meta-Entitätstyp erhält voraussichtlich ein weiteres Meta-Attribut „Level“ vom EIA/CDIF-Datentyp „Integer“, das die einzelnen Abstraktionsebenen in weitere Ebenen unterteilen läßt (vgl. [ISOCDIF98g]).

517 Vgl. [CDIF95], Seite 29.

verfügt, maximal drei Instanzen aufweisen, eine Instanz für jeweils eine der drei vorgesehenen Abstraktionsebenen.[518]

Tabelle 4-15 beschreibt das Meta-Attribut „Name“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***Name***
CDIFMetaIdentifier	***13***
Description	*Der Name der Abstraktionsebene.*
Usage	–
Aliases	–
Constraints	–
DataType	*Enumerated*
Domain	*Concept, Logical, Physical*
Length	–
IsOptional	*False*

Tabelle 4-15: Meta-Attribut „Name“

4.1.2.1.2 Meta-Entitätstyp „AlternateName“

Der Meta-Entitätstyp „AlternateName“[519, 520] ermöglicht die Definition von Aliasnamen, die in die Meta-Attribute „OtherName“ und „OtherLongName“ eingetragen werden können. Derartige alternative Namen sollen nur für Meta-Entitätstypen angegeben werden, die selbst über Meta-Attribute verfügen, die für die Aufnahme von Namen gedacht sind.

Mit Hilfe des Meta-Beziehungstyps „1:1 RootEntity.*Has*.**AlternateName** 0:N“ können einzelnen Instanzen vom Typ „RootEntity“ beliebig viele Namen zugeordnet werden, wobei der Meta-Beziehungstyp „0:N RootEntity.*Uses*.-

518 Werden durch die Erweiterung des Meta-Modells „Common“ im Rahmen eines EIA/CDIF-Austausches zusätzliche aufzählbare Werte vorgesehen, erhöht sich die maximale Anzahl an möglichen Instanzen entsprechend.

519 Vgl. [CDIF95], Seite 30, und Abschnitt 4.1.5, „Naming“ von Seite 15 bis 16.

520 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt.

AlternateName 0:1“ jene Instanz vom Typ „AlternateName“ festlegt, die für die Bezeichnung einer bestimmten Entität vom Typ „RootEntity“ benutzt wird.

Tabelle 4-16 beschreibt den Meta-Entitätstyp „AlternateName“.

Name des Attributs	Bedeutung
Name	***AlternateName***
CDIFMetaIdentifier	***14***
SubtypeOf	*RootEntity*
SupertypeOf	–
Description	*Ein alternativer Name für ein Objekt.*
Usage	*Wenn ein Objekt über mehrere Namen verfügt, werden diese mit Hilfe dieses Meta-Entitätstyps dokumentiert.*
Aliases	*Alias, AlsoKnownAs, AKA, Synonym, Pseudonym*
Constraints	*Eine Instanz von diesem Meta-Entitätstyp soll nur dann gebildet werden, wenn das damit beschriebene Objekt bereits über Namens-Meta-Attribute verfügt.* *Zumindest eines der beiden optionalen Meta-Attribute „OtherLongName“ und „OtherName“ muß über einen Wert aufweisen.*
Type	*Characteristic*

Tabelle 4-16: Meta-Entitätstyp „AlternateName“

Der Meta-Entitätstyp „AlternateName“ verfügt über zwei optionale Meta-Attribute, nämlich „OtherLongName“ und „OtherName“.

4.1.2.1.2.1 Meta-Attribut „OtherLongName“

Das optionale Meta-Attribut „OtherLongName“[521] vom EIA/CDIF-Datentyp „String“ dient zur Aufnahme des vollständigen Namens. Tabelle 4-17 beschreibt das Meta-Attribut „OtherLongName“.

521 Vgl. [CDIF95], Seite 31.

Name des Attributs	Bedeutung
Meta-Attribute Name	***OtherLongName***
CDIFMetaIdentifier	***15***
Description	*Ein weiterer, alternativer Name für ein Objekt.*
Usage	*Ermöglicht die Aufnahme eines weiteren, alternativen, langen Bezeichners.*
Aliases	*FullName*
Constraints	*Ein Wert muß dann angegeben werden, wenn das Meta-Attribut „OtherName“ keinen Wert aufweist.*
DataType	*String*
Domain	–
Length	*1024*
IsOptional	*True*

Tabelle 4-17: Meta-Attribut „OtherLongName“

4.1.2.1.2.2 Meta-Attribut „OtherName“

Das optionale Meta-Attribut „OtherName“[522] vom EIA/CDIF-Datentyp „String“ dient zur Aufnahme eines weiteren Namens für eine Instanz vom Typ „RootEntity“. Tabelle 4-18 beschreibt das Meta-Attribut „OtherName“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***OtherName***
CDIFMetaIdentifier	***16***
Description	*Ein weiterer, alternativer Name für ein Objekt.*
Usage	*Ermöglicht die Aufnahme eines weiteren, alternativen Bezeichners. Vorzugsweise soll dieses Meta-Attribute benutzt werden.*
Aliases	*Label*
Constraints	*Ein Wert muß dann angegeben werden, wenn das Meta-Attribut „OtherLongName“ keinen Wert aufweist.*
DataType	*String*

522 Vgl. [CDIF95], Seite 31.

Name des Attributs	Bedeutung
Domain	–
Length	*256*
IsOptional	*True*

Tabelle 4-18: Meta-Attribut „OtherName“

4.1.2.1.3 Meta-Entitätstyp „DataObject“

Der Meta-Entitätstyp „DataObject“[523, 524] dient als Supertyp für sämtliche Meta-Entitätstypen, die in irgendeiner Form Datenobjekte repräsentieren, die durch Attribute näher beschrieben werden.[525] Tabelle 4-19 beschreibt den Meta-Entitätstyp „DataObject“.

Name des Attributs	Bedeutung
Name	***DataObject***
CDIFMetaIdentifier	***22***
SubtypeOf	*SemanticInformationObject*
SupertypeOf	–
Description	*Dieser Meta-Entitätstyp soll als Supertyp für sämtliche Meta-Entitätstypen dienen, die Datenobjekte repräsentieren und daher durch Attribute beschrieben werden oder aufweisen.*
Usage	–
Aliases	*View*
Constraints	*Dieser Meta-Entitätstyp sollte selbst nicht instanziiert werden, da dafür seine Subtypen vorgesehen sind.*
Type	*Kernel*

Tabelle 4-19: Meta-Entitätstyp „DataObject“

523 Vgl. [CDIF95], Seite 32, und Abschnitt 4.1.10, „Data Object Content“ auf Seite 17 unten.

524 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt. In [ISOCDIF98g] wird dieser Meta-Entitätstyp nicht mehr angeführt.

525 Beispielsweise könnte dieser Meta-Entitätstyp als Ausgangspunkt für die Definition eines EIA/CDIF-Meta-Modells für den Austausch von Modelldaten der „Metadata Coalition“ dienen.

Der Meta-Entitätstyp „DataObject“ verfügt über ein einziges, optionales Meta-Attribut „Name“.

4.1.2.1.3.1 Meta-Attribut „Name“

Das optionale Meta-Attribut „Name“[526] vom EIA/CDIF-Datentyp „String“ dient zur Aufnahme des vollständigen Namens und ist in der folgenden Tabelle 4-20 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***Name***
CDIFMetaIdentifier	***23***
Description	*Der Name des (Daten-) Objekts, der gleichzeitig auch dessen Identität bildet.*
Usage	–
Aliases	–
Constraints	–
DataType	*String*
Domain	–
Length	*256*
IsOptional	*True*

Tabelle 4-20: Meta-Attribut „Name“

4.1.2.1.4 Meta-Entitätstyp „Derivation“

Der Meta-Entitätstyp „Derivation“[527, 528] ermöglicht die Definition von Ableitungen, deren Regeln mit Hilfe der Meta-Attribute „DerivationLanguage“ und „DerivationText“ festgelegt sowie mit dem Meta-Attribut „IsRealizationOf“ näher spezifiziert werden können.

Mit Hilfe des Meta-Beziehungstyps „1:N SemanticInformationObject.*UsedIn*.-**Derivation** 0:N“ können jene Instanzen vom Typ „SemanticInformationObject“

526 Vgl. [CDIF95], Seite 33.
527 Vgl. [CDIF95], Seite 30, und Abschnitt 4.1.5, „Naming“ von Seite 15 bis 16.
528 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt.

angeben werden, aus denen mit Hilfe des Meta-Beziehungstyps „1:N SemanticInformationObject.ProducedBy.**Derivation** 0:N“ Instanz(en) vom Typ „SemanticInformationObject“ abgeleitet werden.

Tabelle 4-21 beschreibt den Meta-Entitätstyp „Derivation“.

Name des Attributs	Bedeutung
Name	***Derivation***
CDIFMetaIdentifier	***26***
SubtypeOf	*SemanticInformationObject*
SupertypeOf	–
Description	*Mit Hilfe dieses Meta-Entitätstyps und den dazu definierten Meta-Beziehungstypen können die Instanzen vom Typ SemanticInformationObject, die als Ausgangspunkte von Ableitungen dienen, sowie die Instanzen vom Typ SemanticInformationObject angegeben werden, die als Ergebnis des Ableitungsvorgangs entstehen.*
Usage	*Dieses Konzept wird für sämtliche Formen von Ableitungen eingesetzt.*
Aliases	–
Constraints	*Soferne mehrere unterschiedliche GegenstandsBereiche benutzt werden, sollen diese daraufhin untersucht werden, ob sie selbst bereits über detailliertere Ableitungsdefinitionen verfügen. In einem solchen Fall sollen sie auch benutzt werden.*
Type	*Associative*

Tabelle 4-21: Meta-Entitätstyp „Derivation“

Der Meta-Entitätstyp „Derivation“ verfügt über drei optionale Meta-Attribute, nämlich „DerivationLanguage“, „DerivationText“ und „IsRealizationOf“.

4.1.2.1.4.1 Meta-Attribut „DerivationLanguage“

Das optionale Meta-Attribut „DerivationLanguage“[529] vom EIA/CDIF-Datentyp „Enumerated“ dient zur Festlegung der Sprache[530], in der die Ableitung spezifiziert wird. Tabelle 4-22 beschreibt das Meta-Attribut „DerivationLanguage“.

529 Vgl. [CDIF95], Seite 35.

Name des Attributs	Bedeutung
Meta-Attribute Name	***DerivationLanguage***
CDIFMetaIdentifier	***29***
Description	*Hier wird die Sprache angegeben, in der der Text des Meta-Attributs „DerivationText“ abgefaßt ist.*
Usage	–
Aliases	–
Constraints	*Hierfür soll ein Wert dann angegeben werden, wenn ein Wert auch für das Meta-Attribut „DerivationText“ existiert.*
DataType	*Enumerated*
Domain	*Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other*[531]
Length	–
IsOptional	*True*

Tabelle 4-22: Meta-Attribut „DerivationLanguage“

4.1.2.1.4.2 Meta-Attribut „DerivationText“

Das optionale Meta-Attribut „DerivationText“[532] vom EIA/CDIF-Datentyp „Text“ erlaubt das Spezifizieren von Ableitungsregeln. Tabelle 4-23 beschreibt das Meta-Attribut „DerivationText“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***DerivationText***
CDIFMetaIdentifier	***27***
Description	*Enthält die Beschreibung der Ableitungsregel(n).*
Usage	–
Aliases	–
Constraints	*Wenn dieses Meta-Attribut einen Wert aufweist, dann soll im Meta-Attribut „DerivationLanguage“ die dafür benutzte Sprache*

530 Die erlaubten, aufzählbaren Werte sind in Fußnote 506 auf Seite 177 angeführt.
531 Vgl. Fußnote 506 auf Seite 177 oben.
532 Vgl. [CDIF95], Seite 36.

Name des Attributs	Bedeutung
	eingetragen werden.
DataType	*Text*
Domain	–
Length	–
IsOptional	*True*

Tabelle 4-23: Meta-Attribut „DerivationText“

4.1.2.1.4.3 Meta-Attribut „IsRealizationOf“

Das optionale Meta-Attribut „IsRealizationOf“[533] vom EIA/CDIF-Datentyp „Boolean“ gibt an, ob eine Ableitung auf derselben Abstraktionsebene (Wert fehlt oder ist „False“) oder einer niedrigstufigeren (Wert ist „True“) konzeptionell angesiedelt ist, als die Instanz(en) vom Typ „SemanticInformationObject“, die als Ausgangspunkt der Ableitung dient (dienen). Tabelle 4-24 beschreibt das Meta-Attribut „IsRealizationOf“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***IsRealizationOf***
CDIFMetaIdentifier	***185***[534]
Description	*Der Wert „True“ gibt an, daß die abgeleiteten Instanzen vom Typ SemanticInformationObject auf einer niedrigeren Abstraktionsebene angesiedelt sind.*[535]
Usage	–
Aliases	–
Constraints	–
DataType	*Boolean*

533 Vgl. [CDIF95], Seite 36.

534 Der Wert außerhalb des Nummernkreises 20 bis 100 weist darauf hin, daß dieses Meta-Attribut ursprünglich nicht vorgesehen war, sondern im Laufe der Jahre dem Meta-Entitätstyp hinzugefügt wurde. Im Kontext der Definitionen dieses Meta-Entitätstyps wäre ansonsten ein Wert in den 20ern zu erwarten.

535 Derartige Wechsel in den Abstraktionsebenen der Modellierung könnte zudem mit Hilfe des Meta-Beziehungstyps „0:N SemanticInformationObject.-*IsCategorizedIn.*Abstraction-Level 0:N“ konkretisiert werden, indem man entsprechende Instanzen dafür bildet.

Name des Attributs	Bedeutung
Domain	–
Length	–
IsOptional	*True*

Tabelle 4-24: Meta-Attribut „IsRealizationOf“

4.1.2.1.5 Meta-Entitätstyp „PresentationInformationObject“

Der Meta-Entitätstyp „PresentationInformationObject“[536] dient als Supertyp für sämtliche Meta-Entitätstypen, die in irgendeiner Form Meta-Entitätstypen des Integrierten EIA/CDIF-Meta-Modells graphisch repräsentieren. Es definiert keine Meta-Attribute. Tabelle 4-25 beschreibt den Meta-Entitätstyp „PresentationInformationObject“.

Name des Attributs	Bedeutung
Name	***PresentationInformationObject***
CDIFMetaIdentifier	***30***
SubtypeOf	*RootEntity*
SupertypeOf	–
Description	*Stellt den Supertyp für MetaEntitäten dar, die Modelldaten graphisch repräsentieren.*
Usage	–
Aliases	–
Constraints	*Dieser Meta-Entitätstyp sollte selbst nicht instanziiert werden, da dafür seine Subtypen vorgesehen sind.*
Type	*Kernel*

Tabelle 4-25: Meta-Entitätstyp „PresentationInformationObject“

536 Vgl. [CDIF95], Seite 37.

4.1.2.1.6 Meta-Entitätstyp „ProcessObject“

Der Meta-Entitätstyp „ProcessObject“[537, 538] dient als Supertyp für sämtliche Meta-Entitätstypen, die in irgendeiner Form Prozesse in weiteren EIA/CDIF-Meta-Modellen repräsentieren. Tabelle 4-26 beschreibt den Meta-Entitätstyp „PresentationInformationObject“.

Name des Attributs	Bedeutung
Name	***ProcessObject***
CDIFMetaIdentifier	***31***
SubtypeOf	*SemanticInformationObject*
SupertypeOf	–
Description	*Generisches Prozeßobjekt, das als Supertyp für detailiertere Definitionen in weiteren EIA/CDIF-Meta-Modellen fungieren soll.*
Usage	–
Aliases	–
Constraints	*Dieser Meta-Entitätstyp sollte selbst nicht instanziiert werden, da dafür seine Subtypen vorgesehen sind.*
Type	*Kernel*

Tabelle 4-26: Meta-Entitätstyp „ProcessObject“

Der Meta-Entitätstyp „ProcessObject“ verfügt über fünf optionale Meta-Attribute, nämlich „ExecutionTimeInterval“, „ExecutionTimeUnit“, „Name“, „Specification-Language“ und „SpecificationText“.

4.1.2.1.6.1 Meta-Attribut „ExecutionTimeInterval“

Das optionale Meta-Attribut „ExecutionTimeInterval“[539] vom EIA/CDIF-Datentyp „Float“ dient zur Festlegung des Intervalls zwischen dem Beginn der Ausführungen von Prozeßobjekten. Tabelle 4-27 beschreibt das Meta-Attribut „ExecutionTimeInterval“.

---

537 Vgl. [CDIF95], Seite 38.

538 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt. In [ISOCDIF98g] wird dieser Meta-Entitätstyp nicht mehr angeführt.

539 Vgl. [CDIF95], Seite 39.

Name des Attributs	Bedeutung
Meta-Attribute Name	***ExecutionTimeInterval***
CDIFMetaIdentifier	***34***
Description	*Das Intervall gibt in Einheiten des Meta-Attributs „ExecutionTime-Unit“ die Zeit zwischen dem Beginn von Ausführungen von Einheiten vom Typ „ProcessObject“ an.*
Usage	–
Aliases	–
Constraints	–
DataType	*Float*
Domain	–
Length	–
IsOptional	*True*

Tabelle 4-27: Meta-Attribut „ExecutionTimeInterval“

4.1.2.1.6.2 Meta-Attribut „ExecutionTimeUnit“

Das Meta-Attribut „ExecutionTimeInterval“[540] vom EIA/CDIF-Datentyp „Enumerated“ weist eines von elf aufzählbaren Werten auf und ist in der folgenden Tabelle 4-28 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***ExecutionTimeUnit***
CDIFMetaIdentifier	***32***
Description	*Dieses Meta-Attribut legt die Ausführungszeiteinheiten fest, deren Anzahl im Meta-Attribut „ExecutionTimeInterval“ angegeben ist.*
Usage	–
Aliases	–
Constraints	–
DataType	*Enumerated*

540 Vgl. [CDIF95], Seite 39.

Name des Attributs	Bedeutung
Domain	*Picosecond, Nanosecond, Microsecond, Millisecond, Second, Minute, Hour, Day, Week, Month, Year*
Length	–
IsOptional	*True*

Tabelle 4-28: Meta-Attribut „ExecutionTimeUnit“

4.1.2.1.6.3 Meta-Attribut „Name“

Das optionale Meta-Attribut „Name“[541] vom EIA/CDIF-Datentyp „String“ dient zur Aufnahme des vollständigen Namens und ist in der folgenden Tabelle 4-29 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***Name***
CDIFMetaIdentifier	***33***
Description	*Der Name des (Prozeß-) Objekts.*
Usage	*Dieses Meta-Attribut enthält den benutzerdefinierten Namen des Prozeßobjekts.*
Aliases	–
Constraints	–
DataType	*String*
Domain	–
Length	*256*
IsOptional	*True*

Tabelle 4-29: Meta-Attribut „Name“

4.1.2.1.6.4 Meta-Attribut „SpecificationLanguage“

Das optionale Meta-Attribut „SpecificationLanguage“[542] vom EIA/CDIF-Datentyp „Enumerated“ dient zur Festlegung der Sprache[543], in der das ProzeßObjekt

541 Vgl. [CDIF95], Seite 33.
542 Vgl. [CDIF95], Seite 41.

spezifiziert wird. Tabelle 4-30 beschreibt das Meta-Attribut „Specification-Language“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***SpecificationLanguage***
CDIFMetaIdentifier	***36***
Description	*Hier wird die Sprache angegeben, in der der Text des Meta-Attributs „SpecificationText“ abgefaßt ist.*
Usage	–
Aliases	–
Constraints	*Hierfür soll ein Wert dann angegeben werden, wenn ein Wert auch für das Meta-Attribut „SpecificationText“ existiert.*
DataType	*Enumerated*
Domain	*Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other*[544]
Length	–
IsOptional	*True*

Tabelle 4-30: Meta-Attribut „SpecificationLanguage“

4.1.2.1.6.5 Meta-Attribut „SpecificationText“

Das optionale Meta-Attribut „SpecificationText“[545] vom EIA/CDIF-Datentyp „Text“ erlaubt das Spezifizieren des ProzeßObjektes. Tabelle 4-31 beschreibt das Meta-Attribut „SpecificationText“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***SpecificationText***
CDIFMetaIdentifier	***35***
Description	*Enthält die Spezifikation der Instanz vom Typ „ProcessObject“.*
Usage	–
Aliases	*P-Spec, Mini-Spec*

543 Die erlaubten aufzählbaren Werte sind in Fußnote 506 auf Seite 177 angeführt.
544 Vgl. Fußnote 506 auf Seite 177 oben.
545 Vgl. [CDIF95], Seite 41.

Name des Attributs	Bedeutung
Constraints	*Wenn dieses Meta-Attribut einen Wert aufweist, dann soll im Meta-Attribut „SpecificationLanguage“ die dafür benutzte Sprache eingetragen werden.*
DataType	*Text*
Domain	–
Length	–
IsOptional	*True*

Tabelle 4-31: Meta-Attribut „SpecificationText“

4.1.2.1.7 Meta-Entitätstyp „RootEntity“

Dieser Meta-Entitätstyp wird im GegenstandsBereich „Common“ explizit verwendet[546] und wurde bereits im fundamentalen Meta-Modell oben definiert. Entsprechend findet sich die vollständige Definition dafür im obigen Abschnitt 4.1.1.1.2, Meta-Entitätstyp „RootEntity“, auf Seite 163.

4.1.2.1.8 Meta-Entitätstyp „SemanticInformationObject“

Der Meta-Entitätstyp „SemanticInformationObject“[547, 548] gilt als Wurzel für sämtliche Meta-Entitätstypen, die semantische Informationen für die Modellierung von unterschiedlichen GegenstandsBereichen aufweisen.

Es definiert die beiden optionalen Meta-Attribute „BriefDescription“ und „FullDescription“, die genauso wie die Meta-Beziehungstypen „0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N“, „0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject 1:N“, „1:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N“, und „1:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N“ von sämtlichen Subtypen über die Generalisierungshierarchie ererbt werden. Die folgende Tabelle 4-32 beschreibt den Meta-Entitätstyp „SemanticInformationObject“.

---

546 Dies bedeutet, daß dafür eine Instanz vom Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ gebildet wird.

547 Vgl. [CDIF95], Seite 44, und Abschnitt 4.1.3, „Semantic Information“ auf Seite 14.

548 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt.

Name des Attributs	Bedeutung
Name	***SemanticInformationObject***
CDIFMetaIdentifier	***4***
SubtypeOf	*RootEntity*
SupertypeOf	*DataObject, Derivation, ProcessObject*
Description	*Dieser Meta-Entitätstyp wird für die Kategorisierung all jener Meta-Entitätstypen des Integrierten EIA/CDIF-Meta-Modells benutzt, die Bedeutung für die Erstellung von Modelldaten selbst tragen.*
Usage	–
Aliases	–
Constraints	*Dieser Meta-Entitätstyp sollte selbst nicht instanziiert werden, da dafür seine Subtypen vorgesehen sind.*
Type	*Kernel*

Tabelle 4-32: Meta-Entitätstyp „SemanticInformationObject“

4.1.2.1.8.1 Meta-Attribut „BriefDescription“

Das optionale Meta-Attribut „BriefDescription“[549] vom EIA/CDIF-Datentyp „String“ erlaubt eine kurze Beschreibung der Instanzen dieses Typs und wird in der nachfolgenden Tabelle 4-33 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***BriefDescription***
CDIFMetaIdentifier	***44***
Description	*Eine kurze, unformatierte Beschreibung des Objekts.*
Usage	*Eine summarische Beschreibung des Zweckes und der Funktion des Objekts.*
Aliases	*Summary*
Constraints	–
DataType	*String*
Domain	–

549 Vgl. [CDIF95], Seite 45.

Name des Attributs	Bedeutung
Length	*1024*
IsOptional	*True*

Tabelle 4-33: Meta-Attribut „BriefDescription“

4.1.2.1.8.2 Meta-Attribut „FullDescription“

Das optionale Meta-Attribut „FullDescription“[550] vom EIA/CDIF-Datentyp „Text“ dient zur Aufnahme der umfassenden und vollständigen Beschreibung von Instanzen dieses Meta-Entitätstyps und wird in der folgenden Tabelle 4-34 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***FullDescription***
CDIFMetaIdentifier	***45***
Description	*Eine detaillierte, wenn nötig formatierte, Beschreibung des Objekts.*
Usage	*Beinhaltet eine umfassende und detaillierte Beschreibung, allerdings keine Spezifikationen.*
Aliases	*Comment,Definition*
Constraints	–
DataType	*Text*
Domain	–
Length	–
IsOptional	*True*

Tabelle 4-34: Meta-Attribut „FullDescription“

4.1.2.1.9 Meta-Entitätstyp „TextualConstraint“

Der Meta-Entitätstyp „TextualConstraint“[551, 552] erlaubt es, Einschränkungen, Nebenbedingungen beziehungsweise Regeln für Meta-Entitätstypen vom Typ

550 Vgl. [CDIF95], Seite 45.
551 Vgl. [CDIF95], Seite 46, und Abschnitt 4.1.7, „Constraints“ auf Seite 17.
552 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt.

„SemanticInformationObject“ zu definieren und stellt dafür die vier optionalen Meta-Attribute „BriefDescription“, „ConstraintExpression“, „ConstraintLanguage“ sowie „FullDescription“ bereit. Der Meta-Beziehungstyp „0:N **Textual-Constraint**.*IsConstraintOn*.SemanticInformationObject 1:N“ erlaubt die Zuordnung der Einschränkung zu den entsprechenden Meta-Entitätstypen. Die folgende Tabelle 4-35 beschreibt den Meta-Entitätstyp „TextualConstraint“.

Name des Attributs	Bedeutung
Name	***TextualConstraint***
CDIFMetaIdentifier	***51***
SubtypeOf	*RootEntity*
SupertypeOf	–
Description	*Dieser Meta-Entitätstyp wird für die Definition von Einschränkungen in textueller Form eingesetzt.*
Usage	*Dieser Meta-Entitätstyp wird dazu benutzt, Einschränkungen, Nebenbedingungen und Regeln zu dokumentieren, die selbst nicht ausdrücklich im Meta-Modell modelliert werden.*
Aliases	*Rule*
Constraints	–
Type	*Characteristic*

Tabelle 4-35: Meta-Entitätstyp „TextualConstraint“

4.1.2.1.9.1 Meta-Attribut „BriefDescription“

Das optionale Meta-Attribut „BriefDescription“[553] vom EIA/CDIF-Datentyp „String“ erlaubt eine kurze Beschreibung der Instanzen dieses Typs und wird in der nachfolgenden Tabelle 4-36 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***BriefDescription***
CDIFMetaIdentifier	***52***
Description	*Eine kurze, unformatierte Beschreibung des Objekts.*

553 Vgl. [CDIF95], Seite 47.

Name des Attributs	Bedeutung
Usage	*Eine summarische Beschreibung des Zweckes und der Funktion des Objekts.*
Aliases	*Summary*
Constraints	–
DataType	*String*
Domain	–
Length	*1024*
IsOptional	*True*

Tabelle 4-36: Meta-Attribut „BriefDescription“

4.1.2.1.9.2 Meta-Attribut „ConstraintExpression“

Das optionale Meta-Attribut „ConstraintExpression“[554] vom EIA/CDIF-Datentyp „Text“ erlaubt das Spezifizieren von Einschränkungen, Nebenbedingungen beziehungsweise von Regeln und wird in der folgenden Tabelle 4-37 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***ConstraintExpression***
CDIFMetaIdentifier	***53***
Description	*Enthält die detaillierte Spezifikation der Einschränkung.*
Usage	–
Aliases	–
Constraints	*Wenn dieses Meta-Attribut einen Wert aufweist, dann soll im Meta-Attribut „ConstraintLanguage“ die dafür benutzte Sprache eingetragen werden.*
DataType	*Text*
Domain	–

554 Vgl. [CDIF95], Seite 47.

Name des Attributs	Bedeutung
Length	–
IsOptional	*True*

Tabelle 4-37: Meta-Attribut „ConstraintExpression“

4.1.2.1.9.3 Meta-Attribut „ConstraintLanguage“

Das optionale Meta-Attribut „ConstraintLanguage“[555] vom EIA/CDIF-Datentyp „Enumerated“ dient zur Festlegung der Sprache[556], in der Einschränkung spezifiziert wird. Tabelle 4-38 beschreibt das Meta-Attribut „ConstraintLanguage“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***ConstraintLanguage***
CDIFMetaIdentifier	***55***
Description	*Hier wird die Sprache angegeben, in der der Text des Meta-Attributs „ConstraintExpression“ abgefaßt ist.*
Usage	–
Aliases	–
Constraints	*Hierfür soll ein Wert dann angegeben werden, wenn ein Wert auch für das Meta-Attribut „ConstraintExpression“ existiert.*
DataType	*Enumerated*
Domain	*Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other*[557]
Length	–
IsOptional	*True*

Tabelle 4-38: Meta-Attribut „ConstraintLanguage“

4.1.2.1.9.4 Meta-Attribut „FullDescription“

Das optionale Meta-Attribut „FullDescription“[558] vom EIA/CDIF-Datentyp „Text“ dient zur Aufnahme der umfassenden und vollständigen Beschreibung von

555 Vgl. [CDIF95], Seite 47.
556 Die erlaubten aufzählbaren Werte sind in Fußnote 506 auf Seite 177 angeführt.
557 Vgl. Fußnote 506 auf Seite 177 oben.

Instanzen dieses Meta-Entitätstyps und wird in der folgenden Tabelle 4-39 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***FullDescription***
CDIFMetaIdentifier	***45***
Description	*Eine detaillierte Beschreibung des Zwecks der Einschränkung.*
Usage	–
Aliases	*Comment,Definition*
Constraints	–
DataType	*Text*
Domain	–
Length	–
IsOptional	*True*

Tabelle 4-39: Meta-Attribut „FullDescription“

4.1.2.1.10 Meta-Entitätstyp „ToolUser“

Der Meta-Entitätstyp „ToolUser“[559, 560] ermöglicht es, Instanzen vom Typ „RootEntity“ die Benutzer zuzuordnen, die die entsprechenden Instanzen angelegt und zuletzt geändert haben. Hierfür stehen die Meta-Attribute „FullName“ und „SystemName“ zur Verfügung.

Mit Hilfe des Meta-Beziehungstyps „0:N RootEntity.*CreatedBy*.ToolUser 0:1“ kann angegeben werden, wer die Entität angelegt hat, und mit „0:N RootEntity.-*LastUpdatedBy*.ToolUser 0:1“ welcher Benutzer zuletzt Änderungen an den Definitionen durchgeführt hat.

Tabelle 4-40 beschreibt den Meta-Entitätstyp „ToolUser“.

---

558 Vgl. [CDIF95], Seite 48.

559 Vgl. [CDIF95], Seite 49.

560 Dieser Meta-Entitätstyp wurde bisher in keinem weiteren EIA/CDIF-Meta-Modell benutzt. Er steht für das Erstellen von EIA/CDIF-Transfers zur Verfügung, für die der Austausch von Auditdaten für die transportierten Konzepte notwendig sind.

Name des Attributs	Bedeutung
Name	***ToolUser***
CDIFMetaIdentifier	***56***
SubtypeOf	*RootEntity*
SupertypeOf	–
Description	*Eine Person, ein Team, ein Projekt, eine Abteilung etc., das als Begründer und Veränderer von Objekten in Frage kommt..*
Usage	–
Aliases	*Department, Team, Project, Person, Group, LoginId*
Constraints	–
Type	*Kernel*

Tabelle 4-40: Meta-Entitätstyp „ToolUser“

4.1.2.1.10.1 Meta-Attribut „FullName“

Das optionale Meta-Attribut „FullName“[561] vom EIA/CDIF-Datentyp „String“ dient zur Aufnahme des vollständigen Namens und wird in der folgenden Tabelle 4-41 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***FullName***
CDIFMetaIdentifier	***57***
Description	*Der volle Name eines Werkzeugbenutzers, also des Namens einer Person, eines Teams, einer Abteilung etc.*
Usage	–
Aliases	–
Constraints	–
DataType	*String*
Domain	–

561 Vgl. [CDIF95], Seite 50.

Name des Attributs	Bedeutung
Length	*256*
IsOptional	*True*

Tabelle 4-41: Meta-Attribut „FullName“

4.1.2.1.10.2 Meta-Attribut „SystemName“

Dieses zwingend vorgeschriebene Meta-Attribut „SystemName“[562] vom EIA/-CDIF-Datentyp „String“ dient zur Aufnahme der Benutzerkennung der WerkzeugBenutzer und wird in der folgenden Tabelle 4-42 beschrieben.

Name des Attributs	Bedeutung
Meta-Attribute Name	***SystemName***
CDIFMetaIdentifier	***57***
Description	*Der Name, unter dem Benutzer dem System gegenüber bekannt sind.*
Usage	–
Aliases	*UserId, UserName, LoginId*
Constraints	–
DataType	*String*
Domain	–
Length	*32*
IsOptional	*False*

Tabelle 4-42: Meta-Attribut „SystemName“

4.1.2.1.11 Meta-Beziehungstyp „0.N RootEntity.*CreatedBy*.ToolUser 0:1“

Der binäre Meta-Beziehungstyp „CreatedBy“ mit dem vollqualifizierten Namen „0:N RootEntity.CreatedBy.ToolUser 0:1“[563] erlaubt die Zuordnung von Instanzen vom Typ „RootEntity“ zu Instanzen vom Typ „ToolUser“. Damit kann fest-

562 Vgl. [CDIF95], Seite 50.
563 Vgl. [CDIF95], Seite 51.

gehalten werden, welcher Benutzer welches Objekt des EIA/CDIF-Austausches angelegt hat.

Aufgrund der Kardinalität von „0:N“ zu „0:1“ kann für eine Instanz vom Typ „RootEntity“ genau eine Instanz vom Typ „ToolUser“ zugeordnet werden. Umgekehrt kann ein WerkzeugBenutzer beliebig viele Instanzen vom Typ „RootEntity“ anlegen. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-43 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „CreatedBy“.

Name des Attributs	**Bedeutung**
Name	***CreatedBy***
CDIFMetaIdentifier	***68***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*1*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität vom Typ „RootEntity“ mit dem WerkzeugBenutzer (Instanz vom Typ „ToolUser“), der die Entität vom Typ „RootEntity“ ursprünglich erzeugt hat.*
Usage	Dieser Meta-Beziehungstyp kann für Protokollzwecke, Änderungsmanagement etc. benutzt werden.
Aliases	*DefinedBy, OriginatedBy*
Constraints	–

Tabelle 4-43: Meta- Beziehungstyp „0:N RootEntity.*CreatedBy.*ToolUser 0:1“

4.1.2.1.12 Meta-Beziehungstyp „1:1 RootEntity.*Has*.-AlternateName 0:N“

Der binäre Meta-Beziehungstyp „Has“ mit dem vollqualifizierten Namen „1:1 RootEntity.*Has*.**AlternateName** 0:N“[564] erlaubt die Zuordnung von Instanzen vom Typ „RootEntity“ zu Instanzen vom Typ „AlternateName“. Damit können Aliasnamen Objekten vom Typ „RootEntity“ zugeordnet werden.

Aufgrund der Kardinalität von „1:1“ zu „0:N“ können für eine Instanz vom Typ „RootEntity“ beliebig viele Instanzen vom Typ „AlternateName“ zugeordnet werden. Umgekehrt *muß* jede Instanz vom Typ „AlternateName“ in *genau* einer Instanz dieses Meta-Beziehungstyps enthalten sein. Somit ist dieser Meta-Beziehungstyp für den Meta-Entitätstyp „AlternateName“ zwingend vorgeschrieben. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-44 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „Has“.

Name des Attributs	Bedeutung
Name	***Has***
CDIFMetaIdentifier	***69***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*1*
MaxSourceCard	*1*
MinDestCard	*0*
MaxDestCard	*N*
Description	*Mit Hilfe dieses Meta-Beziehungstyps können Entitäten vom Typ „RootEntity“ beliebig viele Aliasnamen als Instanzen vom Typ „Alias-Name“ erhalten.*
Usage	–

564 Vgl. [CDIF95], Seite 52.

Name des Attributs	Bedeutung
Aliases	*HasAlias*
Constraints	*Aliasnamen sollen nur für Objekte definiert werden, die selbst über ein Meta-Attribut zur Aufnahme eines Namens verfügen. Zudem ist es nicht erlaubt, Instanzen vom Typ „AliasName“ selbst Aliasnamen zuzuweisen.*

Tabelle 4-44: Meta- Beziehungstyp „1:1 RootEntity.*Has.***AlternateName** 0:N“

4.1.2.1.13 Meta-Beziehungstyp „0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1“

Der binäre Meta-Beziehungstyp „LastUpdatedBy“ mit dem vollqualifizierten Namen „0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1“[565] erlaubt die Zuordnung von Instanzen vom Typ „RootEntity“ zu Instanzen vom Typ „ToolUser“. Damit kann festgehalten werden, welcher Benutzer welches Objekt des EIA/CDIF-Austausches wann zuletzt verändert hat.

Aufgrund der Kardinalität von „0:N“ zu „0:1“ kann einer Instanz vom Typ „RootEntity“ genau eine Instanz vom Typ „ToolUser“ zugeordnet werden. Umgekehrt kann ein WerkzeugBenutzer beliebig viele Instanzen vom Typ „RootEntity“ verändern. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-45 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „LastUpdatedBy“.

Name des Attributs	Bedeutung
Name	***LastUpdatedBy***
CDIFMetaIdentifier	***70***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*0*

565 Vgl. [CDIF95], Seite 53.

Name des Attributs	Bedeutung
MaxDestCard	*1*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität vom Typ „RootEntity“ mit dem WerkzeugBenutzer (Instanz vom Typ „ToolUser“), der die Entität vom Typ „RootEntity“ zuletzt geändert hat.*
Usage	Dieser Meta-Beziehungstyp kann für Protokollzwecke, Änderungsmanagement etc. benutzt werden.
Aliases	*ModifiedBy*
Constraints	–

Tabelle 4-45: Meta- Beziehungstyp „0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1“

4.1.2.1.14 Meta-Beziehungstyp „0:N RootEntity.*Uses*.AlternateName 0:1“

Der binäre Meta-Beziehungstyp „Uses“ mit dem vollqualifizierten Namen „0:N RootEntity.*Uses*.AlternateName 0:1“[566] erlaubt die Zuordnung von Instanzen vom Typ „RootEntity“ zu Instanzen vom Typ „AlternateName“. Damit kann festgehalten werden, welcher Aliasname aus dem Kreis aller möglichen von welchem Objekt des EIA/CDIF-Austausches konkret benutzt wird.

Aufgrund der Kardinalität von „0:N“ zu „0:1“ kann einer Instanz vom Typ „RootEntity“ genau eine Instanz vom Typ „AlternateName“ zugeordnet werden. Umgekehrt kann ein konkreter Aliasname[567] von beliebig vielen Instanzen vom Typ „RootEntity“ benutzt werden. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-46 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „Uses“.

Name des Attributs	Bedeutung
Name	***Uses***
CDIFMetaIdentifier	***71***

566 Vgl. [CDIF95], Seite 54.

567 Es handelt sich hierbei um eine Instanz vom Typ „AlternateName“.

Name des Attributs	Bedeutung
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*1*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität vom Typ „RootEntity“ mit genau einem Aliasnamen als Instanz vom Typ „AlternateName“.*
Usage	*Mit diesem Meta-Beziehungstyp kann festgestellt werden, ob unterschiedliche Instanzen vom Typ „RootEntity“ mit denselben Instanzen vom Typ „RootEntity“ in Beziehung stehen, auch wenn die beteiligten Instanzen unterschiedliche Aliasnamen für dieselbe Instanz verwenden.*
Aliases	–
Constraints	*Die Instanz dieses Meta-Beziehungstyps muß eine Instanz r vom Typ „RootEntity“ mit einer Instanz a vom Typ „AlternateName“ assoziieren, für die gilt, daß a in der Menge A von Aliasnamen enthalten ist, die für r definiert*[568] *ist.*

Tabelle 4-46: Meta- Beziehungstyp „0:N RootEntity.*Uses*.AlternateName 0:1“

4.1.2.1.15 Meta-Beziehungstyp „0:N SemanticInformationObject.-*IsCategorizedIn*.AbstractionLevel 0:N“

Der binäre Meta-Beziehungstyp „IsCategorizedIn“ mit dem vollqualifizierten Namen „0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N“[569] erlaubt die Zuordnung von Instanzen vom Typ „SemanticInformation-Object“ zu Instanzen vom Typ „AbstractionLevel“. Damit kann festgehalten werden, welche SemantischenInformationsObjekte welcher AbstraktionsEbene angehören.

---

568 Diese Zuordnung erfolgt mit Hilfe des Meta-Beziehungstyps „1:1 RootEntity.*Has*.-**AlternateName** 0:N“.

569 Vgl. [CDIF95], Seite 55.

Aufgrund der Kardinalität von „0:N“ zu „0:N“ kann eine Instanz vom Typ „SemanticInformationObject“ mit beliebig vielen Instanzen vom Typ „AbstractionLevel“ in Verbindung stehen.[570] Umgekehrt kann eine Instanz vom Typ „AbstractionLevel“ mit beliebig vielen Instanzen vom Typ „Semantic-InformationObject“ assoziiert sein. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-47 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „IsCategorizedIn“.

Name des Attributs	**Bedeutung**
Name	***IsCategorizedIn***
CDIFMetaIdentifier	***73***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*N*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität vom Typ „SemanticInformationObject“ mit beliebig vielen Instanz vom Typ „AbstractionLevel“.*
Usage	–
Aliases	–
Constraints	–

Tabelle 4-47: Meta- Beziehungstyp „0:N SemanticInformationObject.*IsCategorizedIn*.-AbstractionLevel 0:N“

570 Damit ist es auch möglich, daß eine Instanz vom Typ „SemanticInformationObject“ gleichzeitig in *mehreren* Abstraktionsebenen vom Typ „AbstractionLevel“ zugleich enthalten sein kann.

4.1.2.1.16 Meta-Beziehungstyp „1:N SemanticInformationObject.-*ProducedBy*.Derivation 0:N“

Der binäre Meta-Beziehungstyp „ProducedBy“ mit dem vollqualifizierten Namen „1:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N“[571] erlaubt die Zuordnung von Instanzen vom Typ „SemanticInformationObject“ zu Instanzen vom Typ „Derivation“. Damit kann festgehalten werden, welches SemantischeInformationsObjekt aufgrund welcher Ableitung erzeugt wird.

Aufgrund der Kardinalität von „1:N“ zu „0:N“ kann eine Instanz vom Typ „SemanticInformationObject“ mit beliebig vielen Instanzen vom Typ „Derivation“ in Verbindung stehen.[572] Umgekehrt *muß* eine Instanz vom Typ „Derivation“ mit ein oder beliebig vielen Instanzen vom Typ „SemanticInformationObject“ assoziiert sein. Somit ist dieser Meta-Beziehungstyp für den Meta-Entitätstyp „Derivation“ zwingend vorgeschrieben. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-48 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „ProducedBy“.

Name des Attributs	Bedeutung
Name	***ProducedBy***
CDIFMetaIdentifier	***73***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*1*
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*N*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität s vom Typ „SemanticInformationObject“ mit beliebig vielen Instanzen vom Typ „Derivation“. Somit kann festgehalten werden, wie s abgeleitet wurde.*

571 Vgl. [CDIF95], Seite 56.

572 Damit ist es auch möglich, daß eine Instanz vom Typ „SemanticInformationObject“ aufgrund von *mehreren* Ableitungen vom Typ „Derivation“ entstanden ist.

Name des Attributs	Bedeutung
Usage	–
Aliases	–
Constraints	–

Tabelle 4-48: Meta- Beziehungstyp „1:N SemanticInformationObject.*ProducedBy*.**-Derivation** 0:N“

4.1.2.1.17 Meta-Beziehungstyp „1:N SemanticInformationObject.-*UsedIn*.Derivation 0:N“

Der binäre Meta-Beziehungstyp „UsedIn“ mit dem vollqualifizierten Namen „1:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N“[573] erlaubt die Zuordnung von Instanzen vom Typ „SemanticInformationObject“ zu Instanzen vom Typ „Derivation“. Damit kann festgehalten werden, welches SemantischeInformationsObjekt für welche Ableitung benutzt wird.

Aufgrund der Kardinalität von „1:N“ zu „0:N“ kann eine Instanz vom Typ „SemanticInformationObject“ mit beliebig vielen Instanzen vom Typ „Derivation“ in Verbindung stehen.[574] Umgekehrt *muß* eine Instanz vom Typ „Derivation“ mit ein oder beliebig vielen Instanzen vom Typ „SemanticInformationObject“ assoziiert sein. Somit ist dieser Meta-Beziehungstyp für den Meta-Entitätstyp „Derivation“ zwingend vorgeschrieben. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-49 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „UsedIn“.

Name des Attributs	Bedeutung
Name	***UsedIn***
CDIFMetaIdentifier	***74***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*1*

573 Vgl. [CDIF95], Seite 57.

574 Damit kann eine Instanz vom Typ „SemanticInformationObject“ für verschiedene Ableitungen benutzt werden.

Name des Attributs	Bedeutung
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*N*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität s vom Typ „SemanticInformationObject“ mit beliebig vielen Instanzen vom Typ „Derivation“. Somit kann festgehalten werden, in wievielen Ableitungen s benutzt wird.*
Usage	–
Aliases	–
Constraints	–

Tabelle 4-49: Meta- Beziehungstyp „1:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N“

4.1.2.1.18 Meta-Beziehungstyp „0:N TextualConstraint.*IsConstraintOn*.-SemanticInformationObject 1:N“

Der binäre Meta-Beziehungstyp „UsedIn“ mit dem vollqualifizierten Namen „0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject 1:N“[575] erlaubt die Zuordnung von Instanzen vom Typ „TextualConstraint“ zu Instanzen vom Typ „SemanticInformationObject“. Damit kann festgehalten werden, welches SemantischeInformationsObjekt für welche Ableitung benutzt wird.

Aufgrund der Kardinalität von „0:N“ zu „1:N“ *muß* eine Instanz vom Typ „TextualConstraint“ mit ein oder beliebig vielen Instanzen vom Typ „Semantic-InformationObject“ in Verbindung stehen.[576] Umgekehrt kann eine Instanz vom Typ „SemanticInformationObject“ mit ein oder beliebig vielen Instanzen vom Typ „TextualConstraint“ assoziiert sein. Somit ist dieser Meta-Beziehungstyp für den Meta-Entitätstyp „TextualConstraint“ zwingend vorgeschrieben. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

---

575 Vgl. [CDIF95], Seite 58.

576 Somit kann eine bestimmte textuelle Einschränkung für unterschiedlich viele Instanzen vom Typ „SemanticInformationObject“ gelten.

Die folgende Tabelle 4-50 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „IsConstraintOn“.

Name des Attributs	Bedeutung
Name	***IsConstraintOn***
CDIFMetaIdentifier	***80***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*1*
MaxDestCard	*N*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität t vom Typ „TextualConstraint“ mit beliebig vielen Instanzen vom Typ „Semantic-InformationObject“. Somit kann festgehalten werden, welche Einschränkung t für welche SemantischenInformationsObjekte gilt.*
Usage	–
Aliases	–
Constraints	–

Tabelle 4-50: Meta- Beziehungstyp „0:N **TextualConstraint**.*IsConstraintOn*.Semantic-InformationObject 1:N“

4.1.2.2 Referenzierte Meta-Objekte

In diesem GegenstandsBereich werden zudem über die Vererbung entlang der Generalisierungshierarchie die Meta-Objekte „RootObject“[577] sowie „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“[578] benützt.

4.1.2.3 Auswertungen der Definitionen

Zunächst wird in diesem Abschnitt eine strukturelle Übersicht über das Meta-Modell gegeben, indem einfache Auswertungen tabellarisch aufbereitet werden.

577 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.1 auf Seite 158 vorgestellt.

578 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.3 auf Seite 164 vorgestellt.

Daran anschließend werden sämtliche AttribuierbarenMetaObjekte, die ausdrücklich im Meta-Modell benutzt werden,[579] in der dem Meta-Modell entsprechenden Generalisierungshierarchie dargestellt, wobei MetaObjekte auf derselben Stufe alphabetisch sortiert sind. Abschließend werden jene AttribuierbarenMetaObjekte in alphabetischer Reihenfolge summarisch angeführt, für die ausdrücklich[580] MetaAttribute definiert sind.

Sämtliche Auflistungen in diesem Abschnitt erfolgen anhand der voll qualifizierten Namen der AttribuierbarenMetaObjekte, wobei die einfachen Namen von Meta-Beziehungstypen immer kursiv gesetzt sind. Die Namen von Meta-Entitätstypen, deren Instanzen zwingend in Beziehungen auftreten müssen, werden immer fett hervorgehoben. Die Kardinalitäten der Meta-Beziehungstypen werden ausdrücklich angeführt, sowie die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“ ausdrücklich gezeigt.[581] Sofern für AttribuierbareMetaObjekte die Mehrfachvererbung aufgrund der Definition vorgesehen ist, wird der gesamte vollqualifizierte Namen kursiv gesetzt. Sämtliche lokal definierten Meta-Attribute werden angeführt, wobei zusätzlich ihre EIA/CDIF-Datentypen in kleiner Schrift mitangegeben sind.[582]

4.1.2.3.1 Strukturelle Übersicht

Das Meta-Modell „Common“ definiert insgesamt 45 SammelbareMetaObjekte, wobei die einzige Instanz „RootObject“ des Meta-Meta-Entitätstyps AttribuierbaresMetaObjekt und der Meta-Beziehungstyp „0:N RootEntity.-*IsRelatedTo*.-RootEntity 0:N“, die beide über die Generalisierungshierarchie eingebunden werden, mitgezählt sind. Tabelle 4-51 stellt die Aufstellung dieser SammelbarenMetaObjekte tabellarisch dar.

---

579 Es handelt sich also um Instanzen jener SammelbarenMetaObjekte, die in Instanzen des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ für die Instanz des entsprechenden GegenstandBereiches enthalten sind.

580 Derartige Meta-Attribute werden in den EIA/CDIF-Standards auch als „lokal“ bezeichnet, um sie damit von den über die Generalisierungshierarchie ererbten Meta-Attribute unterscheidbar zu machen.

581 Die Darstellung erfolgt in der vorgesehenen Verkodierung des EIA/CDIF-Datentyps „Identifier“, indem der Wert in Sterne eingefaßt wird (vgl. [CDIF94e], Seiten 11 und 20 unten, „IdentifierValue“).

582 Im Falle des aufzählbaren Datentyps werden auch die gültigen Werte der entsprechenden Wertemenge dargestellt.

Anzahl Instanzen vom Typ AMO[583]	1
Anzahl Instanzen vom Typ ME	10
Anzahl Instanzen vom Typ MR	9
Anzahl Instanzen vom Typ MA	25
Anzahl (Summe) Instanzen vom Typ CMO	45

Tabelle 4-51: Verteilung der SammelbarenMetaObjekte des Meta-Modells „Common“

Daraus läßt sich erkennen, daß für diesen GegenstandsBereich in etwa gleich viele Meta-Entitätstypen wie Meta-Beziehungstypen benötigt wurden.

Tabelle 4-52 führt weitere Analyseergebnisse über die strukturellen Eigenschaften dieses Meta-Modells an, wobei die referenzierten AttribuierbarenMetaObjekte darin nicht mehr enthalten sind.

Anzahl zwingend vorgeschriebener MA		2
Anzahl optionaler MA		18
Meta-Objekte mit/ohne Meta-Attribute	mit MA	ohne MA
Anzahl Instanzen vom Typ AMO	0	0
Anzahl Instanzen vom Typ ME	8	2
Anzahl Instanzen vom Typ MR	0	8
Anzahl von zwingend vorgeschriebenen MR[584]		4
Anzahl von ME, die an zwingend vorgeschriebenen MR teilhaben		3
Anzahl der Blätter (Instanzen vom Typ AMO) im Meta-Modellbaum		16
Anzahl von AMOs mit Mehrfachvererbung		0

Tabelle 4-52: Strukturelle Übersicht über das Meta-Modell für „Common“ [585]

583 Hier wird die einzige direkte Instanz von AMO angeführt, nämlich „RootObject“.

584 Zwingend vorgeschriebene Meta-Beziehungstypen verfügen über einen Wert >=1 in mindestens einer der minimalen Kardinalitäten. Aufgrund der gegenüberliegenden Anschrift der Kardinalitäten im graphischen Modell bedeutet dies, daß ein Wert >=1 im Meta-Meta-Attribut „Min*Source*Card“ zur Folge hat, daß Instanzen des *Ziel*-Meta-Entitätstyps an Instanzen dieses Meta-Beziehungstyps teilnehmen müssen. Entsprechend müssen bei einem Wert >=1 im Meta-Meta-Attribut „Min*Dest*Card“ Instanzen des *Quell*-Meta-Entitätstyps an Instanzen des Meta-Beziehungstyps enthalten sein.

585 In dieser Tabelle werden referenzierte Meta-Objekte nicht berücksichtigt.

Es ist für diesen GegenstandsBereich interessant festzustellen, daß für Meta-Beziehungstypen überhaupt keine Meta-Attribute vorgesehen sind. Hingegen wurden für acht von zehn Meta-Entitätstypen Meta-Attribute für notwendig erachtet. Von den insgesamt 20 im GegenstandsBereich „Common“ direkt definierten Meta-Attribute sind zwei zwingend vorgeschrieben. Es handelt sich hierbei um die folgenden beiden Meta-Attribute, die voll qualifiziert werden:

– „AbstractionLevel.**Name**“ und

– „ToolUser.**SystemName**“.

Damit müssen Instanzen für die beiden Meta-Entitätstypen „AbstractionLevel“ und „ToolUser“ in einem EIA/CDIF-Austausch Werte in den zwingend vorgeschriebenen Meta-Attributen aufweisen.

Zum ersten Mal wird in diesem GegenstandsBereich „Common“ auch von der Möglichkeit Gebrauch gemacht, Meta-Beziehungstypen so zu definieren, daß Instanzen von damit in Beziehung gesetzten Meta-Entitätstypen in jedem Fall in Instanzen des Meta-Beziehungstyps teilhaben müssen. Es sind dies die vier folgenden Meta-Beziehungstypen:

– „**1**:1 RootEntity.*Has*.**AlternateName** 0:N“: Instanzen vom Typ „AlternateName“ müssen *exakt* an einer Instanz dieses Meta-Beziehungstyps partizipieren,

– „**1**:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N“: Instanzen vom Typ „Derivation“ müssen *zumindestens* an einer Instanz dieses Meta-Beziehungstyps partizipieren,

– „**1**:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N“: Instanzen vom Typ „Derivation“ müssen *zumindestens* an einer Instanz dieses Meta-Beziehungstyps partizipieren, und

– „0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject **1**:N“. Instanzen vom Typ „TextualConstraint“ müssen *zumindestens* an einer Instanz dieses Meta-Beziehungstyps partizipieren.

Zusammenfassend müssen also Instanzen der Meta-Entitätstypen „AlternateName“, „Derivation“ und „TextualConstraint“ gleichzeitig auch an Instanzen mindestens eines Meta-Beziehungstyps zwingend teilhaben.

Die Meta-Objekte, die Blätter im Baum dieses Meta-Modells darstellen, sind gleichmäßig auf Meta-Entitätstypen und Meta-Beziehungstypen im Verhältnis von acht zu acht aufgeteilt.

4.1.2.3.2 Aufgefundene Fehler

Die im Rahmen dieser Arbeit erfolgte Analyse der Definitionen in bezug auf die korrekte Nutzung der EIA/CDIF-Datentypen für die Meta-Meta-Attribute sowie mit dem Rahmen, den das Meta-Meta-Modell vorgibt, ergaben für das allgemeine EIA/CDIF-Meta-Modell keinen Fehler.

4.1.2.3.3 Generalisierungshierarchie des Meta-Modells

Die folgende Abbildung 4-6 stellt die Generalisierungshierarchie dieses Meta-Modells dar, bei der aufgrund der Nutzung der Meta-Objekte aus dem fundamentalen GegenstandsBereich „Foundation“ jeweils ein Teilbaum für die Meta-Entitätstypen und einer für die Meta-Beziehungstypen gebildet wird. Die maximale Höhe des Generalisierungsbaumes ist vier, wobei der Zweig für die Meta-Beziehungstypen eine Höhe von drei Ebenen ausmacht. Somit stellt sich der Teilbaum für Meta-Entitätstypen als etwas spezialisierter dar als der für Meta-Beziehungstypen.

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 *Missing_CDIF_1*
        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*.SemanticInformationObject **1**:N *80*

Abbildung 4-6: Generalisierungshierarchie des Meta-Modells „Common“

4.1.2.3.4 Summarische Darstellung der AttribuierbarenMetaObjekte

Die summarische Darstellung der Meta-Entitätstypdefinitionen entspricht dem Kapitel 4.4, „MetaEntity Summary“ eines jeden EIA/CDIF-Standards, die der Meta-Beziehungstypdefinitionen dem Kapitel 4.5, „MetaRelationship Summary“. Hierbei erfolgt die Reihung in der alphabetisch sortierten Reihenfolge der vollqualifizierten Namen der AttribuierbarenMetaObjekte. In dieser Darstellung wird summarisch gezeigt, welche Meta-Attribute über die Generalisierungshierarchie ererbt und welche, wenn überhaupt, lokal definiert wurden. Ererbte Meta-Attribute werden hierbei – wie in den EIA/CDIF-Standards üblich – kursiv dargestellt, lokale mit normaler Schriftauszeichnung. Zwingend vorgeschriebene Meta-Attribute werden zusätzlich fett hervorgehoben.

In den folgenden Tabellen werden auch abweichend von der obigen Darstellung für das fundamentale Meta-Modell[586] aus Platzgründen nur jene Meta-Objekte dargestellt, die selbst über lokale Meta-Attribute verfügen.[587]

Im Unterschied zu den EIA/CDIF-Standards ist in der Aufstellung zusätzlich angegeben, von welchen übergeordneten Typen die ererbten Meta-Attribute stammen und welche Werte[588] für das entsprechende Meta-Meta-Attribut „CDIFMetaIdentifier“ vorgesehen sind. Des weiteren werden die EIA/CDIF-Datentypen für die lokal definierten Meta-Attribute angeführt, im Falle von zwingend vorgeschriebenen Meta-Attributen werden diese lokal fett gesetzt, um darauf aufmerksam zu machen.

---

586 Vgl. hierzu die Ausführungen in Abschnitt 4.1.1.2.4 auf Seite 168ff.

587 Für diesen GegenstandsBereich bedeutet dies, daß es keine Darstellung des Meta-Entitätstyps „PresentationInformationObject“ gibt, da dafür keine Meta-Attribute definiert sind. Die summarische Darstellung von „RootEntity“ erfolgte bereits im fundamentalen Meta-Modell in Abschnitt 4.1.1.1.3 auf Seite 164 oben.

588 Diese Surrogatwerte werden in kleiner Schriftgröße angegeben, wobei die Werte entsprechend [CDIF94e], „ENCODING.1“, mit Sternen eingeschlossen sind.

Sämtliche Meta-Beziehungstypen werden gemeinsam mit den für sie definierten Kardinalitäten dargestellt. Sofern in einem Meta-Beziehungstyp Meta-Entitätstypen zwingend[589] teilhaben müssen, werden die betroffenen Meta-Entitätstypen in allen Darstellungen fett dargestellt.

Somit erlauben es die folgenden Tabellen, auf einen Blick die wesentlichsten Definitionen zu erkennen.

4.1.2.3.4.1 Darstellung der Meta-Entitätstypen

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AbstractionLevel“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es wird ein einziges lokales MetaAttribut definiert, das zugleich zwingend vorgeschrieben ist.

AbstractionLevel *12*		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
Name *13* – Enumerated {Conceptual, Logical, Physical}	**zwingend**	[lokal]

Tabelle 4-53: Summarische Darstellung des Meta-Entitätstyps „AbstractionLevel“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AlternateName“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

AlternateName *14*		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

589 Der Minimalwert der gegenüberliegenden Kardinalität ist in einem solchen Fall größer Null.

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
OtherLongName *15* – String(1024)	optional	[lokal]
OtherName *16* – String(256)	optional	

Tabelle 4-54: Summarische Darstellung des Meta-Entitätstyps „AlternateName“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DataObject“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es wird ein einziges lokales MetaAttribut definiert.

DataObject *22*		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *23* – String(256)	optional	[lokal]

Tabelle 4-55: Summarische Darstellung des Meta-Entitätstyps „DataObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Derivation“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

Derivation *26*		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*

FullDescription *45*	*optional*	
DerivationLanguage *29* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}[590]	optional	[lokal]
DerivationText *27* – Text	optional	
IsRealizationOf *185* – Boolean	optional	

Tabelle 4-56: Summarische Darstellung des Meta-Entitätstyps „Derivation“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ProcessObject“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt fünf lokale Meta-Attribute definiert.

ProcessObject *31*		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
ExecutionTimeInterval *34* – Float	optional	[lokal]
ExecutionTimeUnit *32* – Enumerated {Picosecond, Nanosecond, Microsecond, Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
Name *33* – String(256)	optional	
SpecificationLanguage *36* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}[591]	optional	
SpecificationText *35* – Text	optional	

Tabelle 4-57: Summarische Darstellung des Meta-Entitätstyps „ProcessObject“

590 Vgl. Fußnote 506 auf Seite 177 oben.

591 Vgl. Fußnote 506 auf Seite 177 oben.

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SemanticInformationObject“ dar und wird ausdrücklich in den Gegenstandsbereichen CMMN und DMOD benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

SemanticInformationObject *4*		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44* – String(1024)	optional	[lokal]
FullDescription *45* – Text	optional	

Tabelle 4-58: Summarische Darstellung des Meta-Entitätstyps „SemanticInformation-Object“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „TextualConstraint“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt vier lokale Meta-Attribute definiert.

TextualConstraint *51*		
CDIFIdentifier *5*	*zwingend*	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *52* – String(1024)	optional	[lokal]
ConstraintExpression *53* – Text	optional	

ConstraintLanguage *55* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}[592]	optional
FullDescription *54* – Text	optional

Tabelle 4-59: Summarische Darstellung des Meta-Entitätstyps „TextualConstraint“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ToolUser“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert, wovon eines zwingend vorgeschrieben ist.

ToolUser *56*	
CDIFIdentifier *5*	*zwingend von: RootObject* *1*
DateCreated *6*	*optional*
DateUpdated *7*	*optional*
TimeCreated *8*	*optional*
TimeUpdated *9*	*optional*
FullName *57* – String(256)	optional [lokal]
SystemName *58* – String(32)	zwingend

Tabelle 4-60: Summarische Darstellung des Meta-Entitätstyps „ToolUser“

4.1.2.3.4.2 Darstellung der Meta-Beziehungstypen

Da sämtliche Meta-Beziehungstypen dieses GegenstandsBereiches über keine eigenständigen Meta-Attribute aufweisen und sie gleichzeitig alle direkte Subtypen des fundamentalen Meta-Beziehungstyps „RootEntity.*IsRelatedTo*.-RootEntity“[593] sind, erscheint eine summarische Darstellung über die Vererbung der Meta-Attribute an dieser Stelle nicht für notwendig.

592 Vgl. Fußnote 506 auf Seite 177 oben.

593 Vgl. die entsprechende summarische Darstellung, die in Tabelle 4-13 auf Seite 170 gegeben ist.

Stattdessen wird zusammenfassend in der folgenden Tabelle 4-61 eine Aufstellung der definierten Meta-Beziehungstypen gegeben, gemeinsam mit den entsprechenden Kardinalitäten.[594]

	0:N	RootEntity.*CreatedBy*.ToolUser	0:1	
*	**1**:1	RootEntity.*Has*.**AlternateName**	0:N	
	0:N	RootEntity.*LastUpdatedBy*.ToolUser	0:1	
	0:N	RootEntity.*Uses*.AlternateName	0:1	
	0:N	SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel	0:N	
*	**1**:N	SemanticInformationObject.*ProducedBy*.**Derivation**	0:N	
*	**1**:N	SemanticInformationObject.*UsedIn*.**Derivation**	0:N	
	0:N	**TextualConstraint**.*IsConstraintOn*.SemanticInformationObject	**1**:N	*

Tabelle 4-61: Summarische Darstellung der Meta-Beziehungstypen für den GegenstandsBereich „Common“

[594] Wenn der Quell- (MinDestCard > 0) oder/und der Ziel-Meta-Entitätstyp (MinSourceCard > 0) am Beziehungstyp partizipieren müssen, dann wird auf der entsprechenden Seite diese Tatsache zusätzlich durch einen Stern hervorgehoben. Der Meta-Entitätstyp, der zwingenderweise in Instanzen des entsprechenden Meta-Beziehungstyps enthalten sein muß, wird im vollqualifizierten Namen in der Tabelle fett dargestellt.

4.1.3 Überblick über die weiteren standardisierten EIA/CDIF-Meta-Modelle

In diesem Abschnitt werden die drei verbleibenden, bis Anfang 1998 EIA/CDIF-Standardisierten Meta-Modelle für die GegenstandsBereiche „Data Flow Subject Area“[595], „Data Modeling Subject Area“[596] und „Presentation and Location Subject Area“[597] prägnant vorgestellt. Der Schwerpunkt liegt hierbei in der Darstellung und Präsentation der entsprechenden Generalisierungshierarchien sowie der Darstellung von Meta-Objekten mit ihren Meta-Attributen, sofern sie welche lokal definiert haben. Somit erfolgen für diese GengenstandsBereiche in diesem Abschnitt keine detaillierten Beschreibungen der Meta-Entitätstypen und Meta-Beziehungstypen mehr. Wie bei den detaillierten Darstellungen weiter oben, werden hingegen strukturelle Auswertungen durchgeführt beziehungsweise dokumentiert und diskutiert.[598]

4.1.3.1 DFM – Das EIA/CDIF-Meta-Modell für den GegenstandsBereich „Datenflußmodellierung“

[CDIF96c] definiert in Form eines EIA/CDIF-Meta-Modells jene Konzepte, die grundlegend für die wichtigsten Varianten der Datenflußmodellierung (englisch: „data flow modeling“[599]) sind. Somit soll es theoretisch möglich sein, daß sämtliche Datenflußmodelle unabhängig von der konkreten Modellierungsvariante in

---

595 Das Meta-Modell ist im Standard [CDIF96c] definiert.

596 Das Meta-Modell ist im Standard [CDIF96b] definiert

597 Das Meta-Modell ist im Standard [CDIF96d] definiert

598 Nachdem es in dieser Arbeit um die Darstellung der Strukturen der standardisierten EIA/CDIF-Meta-Modelle geht, verbleibt eine detaillierte Diskussion der Semantik der weiteren GegenstandsBereiche bei den entsprechenden Standards. Die detaillierten Darstellungen der SubjectAreas „Foundation“ und „Common“ sowie die prägnanten Ausführungen über den „General Structuring Mechanism“ (vgl. Abschnitt 4.1.3.1.2 auf Seite 229ff weiter unten) erfolgen deshalb, da die Konzepte und die Zusammenhänge darin aufgrund ihrer allgemeinen Anwendbarkeit für weitere Meta-Modelle von zentraler Bedeutung für das Verständnis der EIA/CDIF-Standards sind. Damit sollen allein mit diesen Informationen EIA/CDIF-konforme Meta-Modelle erstellbar werden.

599 Dieser GegenstandsBereich wurde während seiner Entwicklung vom EIA/CDIF-Komitee mit dem Akronym „DFM“ bezeichnet. Vgl. zu den Ausführungen in diesem Abschnitt auch [CDIF96c], das neben den Standardisierungsdefinitionen auch eine ausführliche Übersicht mit umfangreichen Beispielen gibt, so auf den Seiten 15 bis 38 („4. Subject Area Overview“) und von 131 bis 185 („Appendix A, Examples“).

einem EIA/CDIF-Austausch mit Hilfe dieses Meta-Modells und eventuellen Erweiterungen dazu, getauscht werden können.

Die Methode der Datenflußmodellierung wurde vor allem Ende der 70er Jahre bekannt und in den 80er Jahren populär gemacht und steht seitdem faktisch in den unterschiedlichsten CASE-Systemen[600] zur Verfügung. Ausgehend von [GanSar79], [YouCon79], [DeMar87] wird die Methode der Datenflußmodellierung auch in den 90er Jahren etwa in [Your92] eingesetzt.[601]

In [CDIF96c] werden die grundlegenden Konzepte der Datenflußdiagrammodellierung definiert, wobei entsprechend der Methode insbesondere auf die weitere Zerlegung von Modellelementen in beliebiger Tiefe das Augenmerk gelegt wurde. Zur Abbildung dieser (hierarchischen) Strukturen zur Bildung von Einheiten aus Teilen wurde ein „Allgemeiner Strukturierungsmechanismus“[602] (abgekürzt: „ASM“; englisch: „General Structuring Mechanism“, abgekürzt: „GSM“) entworfen und erstmals im Kontext dieses GegenstandsBereiches eingeführt.

Im Rahmen des GegenstandsBereiches „Datenflußmodellierung“ sind im EIA/CDIF-Meta-Modell folgende Strukturdefinitionen als Subtypen von „DefinitionObject“ für die Repräsentation der grundlegenden Konzepte definiert:[603]

---

600 Vgl. beispielsweise die in [Balz91] auf Seite 596 gegebene Übersicht, die in Summe über dreißig Werkzeuge anführt, in denen die Datenflußdiagramm-Modellierung Bestandteil des entsprechenden CASE-Systems ist.

601 Der hohe Stellenwert, den diese Methode nach wie vor besitzt, läßt sich beispielsweise daraus ersehen, daß [FeJoMi94] die Anwendung der Spezifikationssprache „COLD-1“ auch anhand von Datenflußdiagrammen (Abschnitt 8.2, Seiten 260 bis 264) zeigen.
Vgl. auch [Vern96], Abschnitt 3.3, „Enterprise Modeling Principles“, Seite 80ff, sowie die Kurzbeschreibung von „IDEF0“ (ebenda Seite 98ff), das im amerikanischen Raum (im Rahmen des ICAM-Projektes der US Air Force, vgl. [W3IDEF0]) von großer Bedeutung ist.

602 Eine prägnante Darstellung davon wird in diesem Kapitel im Abschnitt 4.1.3.1.2, „Der Allgemeine Strukturierungsmechanismus“ auf Seite 229ff weiter unten; gegeben.

603 Für jedes Definitionsobjekt werden auch jene Meta-Entitätstypen angeführt, die als Subtypen von „ComponentObject“ die einzelnen Komponenten repräsentieren. Die Aufzählung geht auf die Tabelle 5 in [CDIF96c] auf Seite 20 zurück, somit wird im Zweifel diese Übersichtsdarstellung als korrekt angesehen. Dies ist notwendig, nachdem beispielsweise in der ausführlichen Definition des Meta-Entitätstyps „StoreDefinition“ im Meta-Meta-Attribut „Usage“ auf die Aufzählung von „FlowPort“ als gültige Komponente offensichtlich vergessen wurde.

– „DFMProcessDefinition“[604] (deutsch: „DFMProzeßDefinition“): repräsentiert eine Prozeßdefinition, die sich aus den folgenden Komponenten zusammensetzen kann:

  - „Attribute“ (deutsch: „Attribut“),
  - „DFMProcess“ (deutsch: „DFMProzeß“),
  - „ExternalAgent“ (deutsch: „ExternerAgent“),
  - „Flow“ (deutsch: „Fluß“),
  - „Port“ (deutsch: „Öffnung“[605]) und
  - „Store“ (deutsch: „Speicher“).

– „ExternalAgentDefinition“ (deutsch: „ExterneAgentenDefinition“): repräsentiert eine Agentendefinition, die sich aus den folgenden Komponenten zusammensetzen kann:

  - „Attribute“ (deutsch: „Attribut“),
  - „DFMProcess“ (deutsch: „DFMProzeß“),
  - „ExternalAgent“ (deutsch: „ExternerAgent“),
  - „Flow“ (deutsch: „Fluß“),
  - „FlowPort“ (deutsch: „FlußÖffnung“) und
  - „Store“ (deutsch: „Speicher“).

– „StoreDefinition“ (deutsch: „SpeicherDefinition“): repräsentiert eine Speicherdefinition, die sich aus den folgenden Komponenten zusammensetzen kann:

  - „Attribute“ (deutsch: „Attribut“),
  - „Flow“ (deutsch: „Fluß“),

---

604 Die Definition des Wurzelprozesses erfolgt mit Hilfe des Meta-Beziehungstyps „0:1 DataFlowModel.*HasRoot*.DFMProcessDefinition 0:1“.

605 Nach [Lang89b] hat „port“ die folgenden grundlegenden Bedeutungen: Hafen, Backbord, Tor/Pforte, Öffnung, Portwein und Haltung. In diesem Zusammenhang ist wohl die grundlegende Bedeutung Öffnung gemeint, die am ehesten mit den weiteren Bedeutungsvarianten Auslaß- beziehungsweise Einlaßöffnung konkretisiert wird. Meta-Entitäten vom Typ „Port“ weisen auf die Einlaß-/Auslaßöffnungen von Prozessen/Externen Agenten/Speicher für die (Daten-/Kontroll-/Material-)Flüsse hin.

- „FlowPort“ (deutsch: „FlußÖffnung“) und
- „Store“ (deutsch: „Speicher“).

– „FlowDefinition“ (deutsch: „FlußDefinition“): repräsentiert eine Flußdefinition, die sich aus den folgenden Komponenten zusammensetzen kann:

- „Attribute“ (deutsch: „Attribut“) und
- „Flow“ (deutsch: „Fluß“).

Damit ist bestimmt, welche grundlegende Konzepte für den Austausch von Datenflußmodellen vorgesehen sind. Die Bildung von Datenflußmodellen hängt daher unmittelbar von den Möglichkeiten ab, die der Allgemeine Strukturierungsmechanismus zur Verfügung stellt:

– Daraus erklärt sich unter anderem auch, daß es für diesen GegenstandsBereich sehr wenige Meta-Beziehungstypen gibt. Mit Ausnahme der Meta-Beziehungstypen für den allgemeinen Strukturierungsmechanismus finden sich nur noch vier Beziehungstypen, wobei „Produces“ und „Consumes“ Subtypen vom Meta-Beziehungstyp „ProducesOrConsumes“ sind. Somit wird in diesem GegenstandsBereich auch zum ersten Mal von der Subtypbildung von Meta-Beziehungstypen Gebrauch gemacht.

– Zudem läßt sich daraus auch die interessante Tatsache erklären, daß mit den drei Ausnahmen[606] „ComponentObject“, „DataFlowModel“ und „DefinitionObject“ *sämtliche* Meta-Entitätstypen dieses GegenstandsBereiches entweder direkte oder indirekte Subtypen der Meta-Entitätstypen „ComponentObject“ und „DefinitionObject“ sind, die selbst fundamental für den Aufbau von Strukturen mit Hilfe des allgemeinen Strukturierungsmechanismus sind, der im Abschnitt 4.1.3.1.2 beginnend mit Seite 229 prägnant besprochen wird.

Eine weitere Besonderheit dieses GegenstandsBereiches liegt darin, daß die Definition des Meta-Entitätstyps „FlowPort“ mit Hilfe der Mehrfachvererbung erfolgt, indem als seine unmittelbaren Supertypen die Meta-Entitätstypen „Flow-

606 Alle drei folgenden Meta-Entitätstypen sind Subtypen des Meta-Entitätstyps „SemanticInformationObject“, der ursprünglich im GegenstandsBereich „Common“ definiert wurde (vgl. den entsprechenden Abschnitt 4.1.2 auf Seite 176).

ProducerConsumer“ und „Port“ angeführt werden. Damit werden für sämtliche Arten von Flüssen[607] Öffnungen für die Erzeuger oder Konsumenten geschaffen.

4.1.3.1.1 Kurzcharakterisierung des EIA/CDIF-GegenstandsBereiches „Datenflußmodellierung“

In diesem Abschnitt werden die Konzepte kurz erläutert, die für den Austausch von Modelldaten aus dem GegenstandsBereich „Datenflußmodellierung“ vorgesehen sind. Die grundlegenden Konzepte sind:

- *ExterneAgenten* (englisch: „ExternalAgent“) sind der Ausgangspunkt und das Ziel von *Flüssen* (englisch: „Flow“),
- mit Hilfe von *ExternenAgentenDefinitionen* (englisch: „ExternalAgent-Definition“) wird die Struktur für *ExterneAgenten* festgelegt, wobei dies in diesem Zusammenhang das Definieren von *FlußÖffnungen*[608] (englisch: „FlowPort“) bedeutet, durch die die entsprechenden *Flüsse* laufen,
- *DFMProzesse*[609] (englisch: „DFMProcess“) verarbeiten, was über *Flüsse* herangetragen wird und benutzen dazu gegebenenfalls *Speicher* (englisch: „Store“), um auf benötigte Arbeitsmittel zuzugreifen beziehungsweise entsprechend veränderte oder erzeugte Arbeitsmittel zu speichern, wobei die Ergebnisse der Verarbeitung im *DFMProzeß* über *Flüsse* an weitere *DFMProzesse* oder *ExterneAgenten* weitergeleitet werden,
- *DFMProzeßDefinitionen* (englisch: „DFMProcessDefinition“) legen die Struktur der *DFMProzesse* fest und weisen in jedem Fall *FlußÖffnungen* auf, zudem geben sie unter anderem darüber Auskunft, um welche Art[610]

---

607 Es handelt sich hierbei konkret um die Subtypen „FlowOutputPort“, „FlowInputPort“ und dessen Subtypen „ConstraintPort“, ControlPort“ sowie „SupportPort“.

608 *Öffnungen* werden im Kontext von Strukturdefinitionen als „formal“ gekennzeichnet, indem das optionale boole’sche Meta-Attribut „IsFormal“ beim Meta-Entitätstyp „Port“ auf den Wert „True“ gesetzt wird; im Kontext der Benutzung mit Komponenten hingegen als „aktuell“ bezeichnet (Port.IsFormal=False).

609 Im Kontext dieses Standards wird der Begriff „Prozeß“ und „Funktion“ synonym benutzt, vgl. hierzu auch [CDIF96c], Seite 71 oben.

610 Es wird die Verarbeitung von Daten, Materialien und Kontrollinformationen unterschieden.

von *DFMProzeß* es sich handelt beziehungsweise ob in diesem *DFMProzeß* nebenläufige Teilprozesse auftreten,

- *Flüsse* können über *FlußÖffnungen* in *ExterneAgenten*, *DFMProzesse* und in *Speicher* einfließen beziehungsweise daraus fließen,
- *FlußDefinitionen* (englisch: „FlowDefinition“) legen die Struktur von Flüssen fest, wobei zwischen Analog-, Kontroll-, Daten- und Materialflüssen unterschieden werden kann,
- *FlußÖffnungen* lassen sich weiter verfeinern in *FlußAusgabeÖffnungen* (englisch: „FlowOutputPort“) und *FlußEingabeÖffnungen* (englisch: „FlowInputPort“), die selbst wiederum in *KontrollÖffnungen* (englisch: „ControlPort“)*, UnterstützungsÖffnungen* (englisch: „SupportPort“) und *EinschränkungsÖffnungen* (englisch: „ConstraintPort“) unterteilt werden können.

Ausgangspunkt der Bildung und Zerlegung von Datenflußmodellen ist ein *DatenFlußModell* (englisch: „DataFlowModel“)*,* das als Wurzel eine *DFMProzeß-Definition* aufweist, die über eine Instanz des Meta-Beziehungstyps „HasRoot“ zugeordnet wird.[611] Sämtliche Strukturen können als Komponenten *Attribute* aufweisen, sodaß bei Bedarf weitere Charakterisierungen möglich sind.

4.1.3.1.2 Der Allgemeine Strukturierungsmechanismus

Der „General Structuring Mechanism“[612, 613, 614] (abgekürzt: „GSM“; deutsch: „Allgemeiner Strukturierungsmechanismus“, abgekürzt: „ASM“) wurde von

611 Diese Festlegung kann auch aufgrund der Kardinalität von 0:1 und 0:1 entfallen.

612 Der allgemeine Strukturierungsmechanismus gewann für die Entwicklung von weiteren EIA/CDIF-Standards an so zentraler Bedeutung, daß er im Entwurf von ISO/CDIF in den GegenstandsBereich „Common Subject Area“ (vgl. [ISOCDIF98g]) aufgenommen werden soll.

613 In [Ernst98] wird die Entwicklung des Allgemeinen Strutkurierungsmechanismus ausführlich dokumentiert, Weiterentwicklungsmöglichkeiten dargestellt und umfassend diskutiert (vgl. ebenda Abschnitt 3.6.6, „A General Structuring Mechanism“, Seite 82ff). Ernst gilt als einer der Väter des Allgemeinen Strukturierungsmechanismus’, sodaß die Ausführungen und Weiterentwicklungsvorschläge ebenda auch aus diesem Blickwinkel interessant sind.

614 Vgl. auch [CDIF96c], Abschnitt 4.1.7, „The General Structuring Mechanism“ von Seite 17 bis 20, und in weiterer Folge die Anwendungen davon ebenda von Seite 20 bis 36.

EIA/CDIF ursprünglich für den GegenstandsBereich „Datenflußmodellierung“ entwickelt und hat mittlerweile in weitere Meta-Modelle Eingang gefunden.[615]

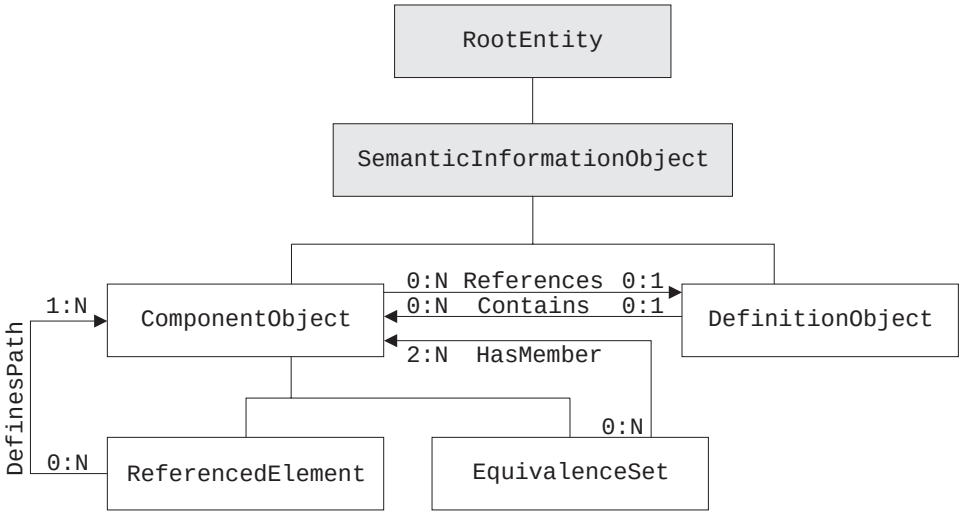


Abbildung 4-7: Ausschnitt des Meta-Modells für die Abbildung des Allgemeinen Strukturierungsmechanismus[616]

Mit dem allgemeinen Strukturierungsmechanismus können einerseits Ganze-/Teilebeziehungen, andererseits auch Äquivalenzen von unterschiedlichen Instanzen von Meta-Entitätstypen sowie eindeutige Referenzierungen von konkreten Instanzen von Strukturbestandteilen über ausdrückliche Pfadangaben erfolgen. Die Meta-Entitätstypen „ComponentObject“, „DefinitionObject“, „EquivalenceSet“, „ReferencedElement“ und die Meta-Beziehungstypen „Contains“, „References“, „HasMember“ und „DefinesPath“, die diese Meta-

615 ISO/IETC SC7/WG11 arbeitet daher daran, den General Structuring Mechanism in einer leicht erweiterten Form in den ISO/CDIF-GegenstandsBereich „Common“ einzubetten (vgl. hierzu [ISOCDIF98g]).

616 Vgl. [CDIF96c], Seite 18. Die Meta-Entitätstypen „RootEntity“ und „SemanticInformationObject“ sind in Abbildung 4-7 entsprechend den EIA/CDIF-Konventionen grau hinterlegt. Damit wird ausgedrückt, daß diese Entitätstypen nicht in diesem GegenstandsBereich definiert sind, sondern lediglich zur leichteren Verständlichkeit des Diagramms darin vorkommen.

Entitätstypen als Quell- oder Zielentitätstypen aufweisen, erlauben die Bildung derartiger, beliebig verschachtelter Strukturen.[617]

4.1.3.1.2.1 Definition von Strukturen in Form von Ganzen und ihren Teilen

Die Definition von Strukturen, die in Form von Ganze-/Teilebeziehungen *innerhalb* einer Modellierungsebene festgelegt werden, erfolgt durch die Instanziierung von Subtypen der abstrakten Meta-Entitätstypen „DefinitionObject“ und „ComponentObject“ sowie einer Instanz vom Meta-Beziehungstyp „0:1 DefinitionObject.*Contains*.ComponentObject 0:N“.[618, 619] Aus den Kardinalitäten folgt, daß ein DefinitionsObjekt aus beliebig vielen verschiedenen KomponentenObjekten zusammengesetzt sein kann. Umgekehrt darf eine Instanz vom Typ „ComponentObject“ maximal ein einziges Mal als Bestandteil einer Instanz vom Typ „DefinitionObject“ auftreten.

Derartige DefinitionsObjekte bilden die Struktur von jenen KomponentenObjekten, die an entsprechenden Instanziierungen des Meta-Beziehungstyps „0:N ComponentObject.*References*.DefinitionObject 0:1“ partizipieren.[620] Damit wird

---

617 Die Meta-Entitätstypen „DefinitionObject“ und „ComponentObject“ sind als abstrakt anzusehen, sodaß für die Modellierung jeweils Instanzen von den entsprechenden Subtypen für den Aufbau von zusammengesetzten Strukturen herangezogen werden.

618 [Wede92] setzt sich mit derartigen mereologischen Strukturen und mereologischen Operatoren auseinander. Ebenda werden mereologische Strukturen immer wieder angesprochen und diskutiert, beispielsweise auf den Seiten 43 oder 116 bis 117, wobei ihnen schließlich sogar ein eigener Abschnitt, „3.7 Mereologische Strukturen, ihre Auflösung und Geltungssicherung“, beginnend mit Seite 150 gewidmet wird. Eine erste Diskussion erfolgt bereits im Abschnitt 2.7, „Vierter Schritt: Zusammensetzung von Objekten“ auf Seite 40ff, in der auch in diesem Zusammenhang das bekannte BOM-(englisches Akronym für: „Bill of Materials“, deutsch: „Stücklisten“) Problem angesprochen wird.

619 Der Meta-Entitätstyp „DefinitionObject“ weist ein Meta-Attribut „Operator“ auf, das spätestens dann einen der drei möglichen Werte „AND“, „XOR“ und „OR“ annehmen soll, sobald über den Meta-Beziehungstyp „0:1 DefinitionObject.*Contains*.ComponentObject 0:N“ ein KomponentenObjekt zugeordnet wurde: beim Wert „AND“ werden *alle* KomponentenObjekte verwendet, bei „XOR“ *genau eine* der zugeordneten KomponentenObjekte und bei „OR“ schließlich muß *mindestens eine* der vorgesehenen KomponentenObjekte instanziiert werden. Vgl. hierzu auch die Beschreibung des Meta-Attributs „Operator“ in [CDIF96c], Seite 66.

620 Wie weiter oben bereits erwähnt wurde, besteht ein Datenflußmodell für den GegenstandsBereich „Datenflußmodellierung“ aus einer Instanz vom Typ „DataFlowModel“, die über eine Instanz vom Meta-Beziehungstyp „HasRoot“ mit einer Instanz vom Typ „DFM-ProcessDefinition“ verbunden ist. Vgl. hierzu auch die Definitionen in [CDIF96c], beispielsweise auf den Seiten 39 unten, 112, aber auch die Festlegungen und Erläuterungen auf den Seiten 63ff und 71ff.

die Möglichkeit geschaffen, definierte Strukturen beliebig oft wiederzuverwenden.

4.1.3.1.2.2 Festlegen von Pfaden

Unterschiedliche ComponentObjects können ein und dieselbe Struktur aufweisen, indem – wie im vorhergehenden Abschnitt ausgeführt – der Meta-Beziehungstyp „References“ dafür öfters instanziiert wird und als Zielinstanz immer dasselbe DefinitionsObjekt aufweist. Abhängig von unterschiedlichen Problemstellungen kann es notwendig werden, ausdrücklich und eindeutig einzelne Komponenten einer Struktur zu adressieren. Für den Fall, daß ein DefinitionsObjekt für mehr als ein KomponentenObjekt benutzt wird, stellt sich hierbei ein Referenzierungsproblem: welche Teilkomponente welcher strukturgleichen[621] Komponente soll referenziert werden?

Der allgemeine Strukturierungsmechanismus sieht für diesen Fall vor, daß eine Instanz vom Typ „ReferencedElement“ in entsprechend vielen Beziehungen vom Typ „DefinesPath“ enthalten ist, wie dies für die eindeutige Adressierung der einzelnen Komponenten als notwendig erscheint. Die Reihenfolge, in der die einzelnen Komponenten aufzusuchen sind, wird durch das zwingend vorgeschriebene Meta-Attribut „SequenceNumber“ für den Meta-Beziehungstyp „Defines-Path“ dokumentiert.[622, 623]

4.1.3.1.2.3 Festlegen von Äquivalenzen

Es ist vorstellbar, daß einander entsprechende Konzepte in Form von unterschiedlichen Instanzen vom Typ ComponentObjects in einem Modell an unterschiedlichen Stellen als Komponenten eingesetzt werden. Die Äquivalenz von

---

621 Strukturgleichheit wird hierbei dadurch bestimmt, daß die KomponentenObjekte jeweils dasselbe DefinitionsObjekt referenzieren und daher alle über dessen definierte Struktur aufweisen.

622 Die Adressierung erfolgt, indem vom Wurzelbestandteil ausgehend, Schritt für Schritt, Komponente für Komponente durchwandert wird, bis die Zielkomponente erreicht ist. Für jeden dieser Schritte wird eine Beziehung vom Typ „DefinesPath“ erzeugt, die die entsprechenden Instanzen von ReferencedElement und ComponentObject miteinander assoziiert, und die entsprechende SequenzNummer dafür vermerkt.

623 Im GegenstandsBereich „Datenflußmodellierung“ wird dieser Mechanismus auch für die Numerierung der unterschiedlichen Komponenten benutzt, indem beispielsweise für einen Speicher lediglich die lokale Identifikation („ContextIdentifier“) gespeichert wird. Die vollständige Identifikation ergibt sich dann daraus, daß entlang des Pfades zu dem ent-

Fortsetzung folgt auf der nächsten Seite.

zwei oder mehreren derartiger Bestandteile kann ausdrücklich vermerkt werden, indem eine Instanz vom Typ „EquivalenceSet“ über entsprechend viele Beziehungen vom Typ „HasMember“ mit den entsprechenden Instanzen vom Typ „ComponentObject“ assoziiert wird.[624]

Aufgrund der angegebenen Kardinalität[625] wird klar, daß eine Instanz vom Typ „ÄquivalenzMenge“ *mindestens* mit *zwei* Instanzen vom Typ „ComponentObject“ in Beziehung stehen muß.

4.1.3.1.3 Referenzierte Meta-Objekte

In diesem GegenstandsBereich werden zudem über die Vererbung entlang der Generalisierungshierarchie die Meta-Objekte „RootObject“[626], „RootEntity“[627], „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“[628] sowie „SemanticInformationObject“[629] benützt.

4.1.3.1.4 Auswertungen der Definitionen

Zunächst wird in diesem Abschnitt eine strukturelle Übersicht über das Meta-Modell gegeben, indem einfache Auswertungen tabellarisch aufbereitet werden. Daran anschließend werden sämtliche AttribuierbarenMetaObjekte, die ausdrücklich im Meta-Modell benutzt werden,[630] in der dem Meta-Modell entsprechenden Generalisierungshierarchie dargestellt, wobei MetaObjekte auf derselben Stufe alphabetisch sortiert sind. Abschließend werden jene Attribuierba-

---

sprechenden lokalen Speicher in der gewählten Reihenfolge die entsprechenden lokalen Identifikatoren gesammelt und miteinander kombiniert werden.

624 Wenn beispielsweise in einem Modell zwei der angeführten Speicher identisch sind, werden sie trotzdem als zwei selbständige Komponenten modelliert, die dieselbe Strukturdefinition referenzieren. Mit der ÄquivalenzMenge werden beide Instanzen gleichgesetzt.
Oder wenn beispielsweise ein Kunde oder Lieferant in einem Modell sowohl als ExternerAgent als auch als Speicher eingesetzt wird, kann mit Hilfe von ÄquivalenzMengen eine Gleichsetzung erfolgen.

625 Damit ist folgender Meta-Beziehungstyp angesprochen: „0:N **EquivalenceSet**.*HasMember*.ComponentObject 2:N“.

626 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.1 auf Seite 158 vorgestellt.

627 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.2 auf Seite 163 vorgestellt.

628 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.3 auf Seite 164 vorgestellt.

629 Die Definition wurde bereits weiter oben im Abschnitt 4.1.2.1.8 auf Seite 194 vorgestellt.

630 Es handelt sich also um Instanzen jener SammelbarenMetaObjekte, die in Instanzen des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ für die Instanz des entsprechenden GegenstandsBereiches enthalten sind.

renMetaObjekte in alphabetischer Reihenfolge summarisch angeführt, für die ausdrücklich[631] MetaAttribute definiert sind.

Sämtliche Auflistungen in diesem Abschnitt erfolgen anhand der voll qualifizierten Namen der AttribuierbarenMetaObjekte, wobei die Namen von Meta-Beziehungstypen immer kursiv gesetzt sind. Die Namen von Meta-Entitätstypen, deren Instanzen zwingend in Beziehungen auftreten müssen, werden immer fett hervorgehoben. Die Kardinalitäten der Meta-Beziehungstypen werden ausdrücklich angeführt, sowie die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“[632]. Sofern für AttribuierbareMetaObjekte die Mehrfachvererbung aufgrund der Definition vorgesehen ist, wird der gesamte vollqualifizierte Namen kursiv gesetzt. Sämtliche lokal definierten Meta-Attribute werden angeführt, wobei zusätzlich ihr EIA/CDIF-Datentyp in kleiner Schrift mitangegeben ist.[633]

4.1.3.1.4.1 Strukturelle Übersicht

Das Meta-Modell „Data Flow Modeling“ definiert insgesamt 84 SammelbareMetaObjekte, wobei die einzige direkte Instanz „RootObject“ des Meta-Meta-Entitätstyps „AttribuierbaresMetaObjekt“, die Meta-Entitätstypen „RootEntity“ und „SemanticInformationObject“ sowie der Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“, die über die Generalisierungshierarchie eingebunden werden, mitgezählt sind. Tabelle 4-65 stellt die Aufstellung dieser SammelbarenMetaObjekte tabellarisch dar.

631 Derartige Meta-Attribute werden in den EIA/CDIF-Standards auch als „lokal“ bezeichnet, um sie damit von über die Generalisierungshierarchie ererbten Meta-Attribute unterscheidbar zu machen.

632 Die Darstellung erfolgt in der vorgesehenen Verkodierung des EIA/CDIF-Datentyps „Identifier“, indem der Wert in Sterne eingefaßt wird (vgl. [CDIF94e], Seiten 11 und 20 unten, „IdentifierValue“).

633 Im Falle des aufzählbaren Datentyps werden auch die gültigen Werte der entsprechenden Wertemenge dargestellt.

Anzahl Instanzen vom Typ AMO[634]	1
Anzahl Instanzen vom Typ ME[635]	24
Anzahl Instanzen vom Typ MR[636]	9
Anzahl Instanzen vom Typ MA[637]	50
Anzahl (Summe) Instanzen vom Typ CMO	84

Tabelle 4-62: Verteilung der SammelbarenMetaObjekte des Meta-Modells „Data Flow Modeling“

Interessanterweise fallen in diesem GegenstandsBereich auf die Definition eines Meta-Beziehungstyps grob 2,5 Meta-Entitätstypen, ein Verhältnis, das für das Meta-Modell der Datenflußmodellierung charakteristisch ist. Dies erklärt sich aus der Tatsache, daß fast sämtliche Strukturen mit Hilfe des allgemeinen Strukturierungsmechanismus erstellt werden und darüber hinaus eigentlich nur ein spezifischer Meta-Beziehungstyp „ProducesOrConsumes“ mit den zwei Subtypen „Produces“ und „Consumes“ zur Verfügung stehen.

Werden die Meta-Entitätstypen und Meta-Beziehungstypen des allgemeinen Strukturierungsmechanismus nicht berücksichtigt, so werden spezifisch für die Datenflußmodellierung 18 Meta-Entitätstypen und drei Meta-Beziehungstypen definiert. Es entsteht dadurch ein bemerkenswertes Verhältnis von sechs Meta-Entitätstypen auf einen Meta-Beziehungstyp!

Tabelle 4-63 führt weitere Analyseergebnisse über die strukturellen Eigenschaften dieses Meta-Modells an, wobei die referenzierten AttribuierbarenMetaObjekte darin nicht mehr enthalten sind.

634 Hier wird die einzige direkte Instanz von AMO angeführt, nämlich „RootObject“.

635 Die beiden referenzierten Meta-Entitätstypen „RootEntity“ und „SemanticInformationObject“ sind in der Summe beinhaltet.

636 Der referenzierte Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ ist in der Summe eingerechnet.

637 Es sind insgesamt sieben Meta-Attribute mitberücksichtigt, die von den referenzierten AttribuierbarenMetaObjekten stammen. Es handelt sich hierbei um die Meta-Attribute von RootObject („CDIFIdentifier“, „DateCreated“, „DateUpdated“, „TimeCreated“, „TimeUpdated“) und von SemanticInformationObject („BriefDescription“, „FullDescription“).

Anzahl zwingend vorgeschriebener MA		2
Anzahl optionaler MA		41
Meta-Objekte mit/ohne Meta-Attribute	mit MA	ohne MA
Anzahl Instanzen vom Typ AMO	0	0
Anzahl Instanzen vom Typ ME	11	11
Anzahl Instanzen vom Typ MR	2	6
Anzahl von zwingend vorgeschriebenen MR[638]		2
Anzahl von ME, die an zwingend vorgeschriebenen MR teilhaben		2
Anzahl der Blätter (Instanzen vom Typ AMO) im Meta-Modellbaum[639]		23
Anzahl von AMOs mit Mehrfachvererbung[640]		1

Tabelle 4-63: Strukturelle Übersicht über das Meta-Modell für „Data Flow Modeling“[641]

In diesem GegenstandsBereich wird zum ersten Mal einer von insgesamt acht Meta-Beziehungstypen, nämlich „0:N **ReferencedElement**.*DefinesPath*.-ComponentObject 1:N“ mit einem Meta-Attribut versehen. Exakt die Hälfte der Meta-Entitätstypen und (erstmals) ein Drittel der Meta-Beziehungstypen weisen Meta-Attribute auf.

Von den insgesamt 43 im Gegenstandsbereich „Data Flow Modeling“ direkt definierten Meta-Attributen sind zwei zwingend vorgeschrieben, eines jeweils für einen Meta-Beziehungstyp, eines für einen Meta-Entitätstyp. Es handelt sich hierbei um die folgenden beiden Meta-Attribute, die voll qualifiziert werden:

– „Port.**IsFormal**“ und

638 Zwingend vorgeschriebene Meta-Beziehungstypen verfügen über einen Wert >=1 in mindestens einer der minimalen Kardinalitäten. Aufgrund der gegenüberliegenden Anschrift der Kardinalitäten im graphischen Modell bedeutet dies, daß ein Wert >=1 im Meta-Meta-Attribut „Min*Source*Card“ zur Folge hat, daß Instanzen des *Ziel*-Meta-Entitätstyps an Instanzen dieses Meta-Beziehungstyps teilnehmen müssen. Entsprechend müssen bei einem Wert >=1 im Meta-Meta-Attribut „Min*Dest*Card“ Instanzen des *Quell*-Meta-Entitätstyps an Instanzen des Meta-Beziehungstyps enthalten sein.

639 Blattypen, die einer Mehrfachvererbung unterworfen sind, werden lediglich einmal gezählt.

640 Hier werden nur jene Meta-Entitätstypen gezählt, für die die Mehrfachvererbung ausdrücklich festgelegt ist. Selbstverständlich sind aufgrund der Vererbung entlang der Generalisierungshierarchie auch sämtliche entsprechenden Subtypen der Mehrfachvererbung unterworfen.

641 In dieser Tabelle werden referenzierte Meta-Objekte nicht berücksichtigt.

– „**ReferencedElement**.*DefinesPath*.ComponentObject.**Sequence-Number**“.

Damit müssen Instanzen für den Meta-Entitätstyp „Port“ und den Meta-Beziehungstyp „DefinesPath“ in einem EIA/CDIF-Austausch Werte in den zwingend vorgeschriebenen Meta-Attributen aufweisen.

Wie im Gegenstandsbereich „Common“ wird auch hier von der Möglichkeit Gebrauch gemacht, Meta-Beziehungstypen so zu definieren, daß Instanzen von damit in Beziehung gesetzten Meta-Entitätstypen in jedem Fall in Instanzen des Meta-Beziehungstyps enthalten sein müssen. Es sind dies die beiden folgenden Meta-Beziehungstypen:

– „0:N **EquivalenceSet**.*HasMember*.ComponentObject **2**:N“: Instanzen vom Typ „EquivalenceSet“ müssen zwingend in *mindestens zwei* Beziehungen dieses Typs enthalten sein und somit mindestens zwei Instanzen vom Typ „ComponentObject“ gleichsetzen,

– „0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N“: Instanzen vom Typ „ReferencedElement“ müssen zwingend an *mindestens einer* Instanz dieses Meta-Beziehungstyps partizipieren.

Zusammenfassend müssen also Instanzen der Meta-Entitätstypen „EquivalenceSet“ und „ReferencedElement“ gleichzeitig auch an Instanzen der entsprechend zwingenden Meta-Beziehungstypen teilhaben.

Von den insgesamt 23 Blättern in diesem Meta-Modellbaum sind sieben Meta-Beziehungstypen und 16 Meta-Entitätstypen.

4.1.3.1.4.2 Aufgefundene Fehler

Die im Rahmen dieser Arbeit erfolgte Analyse der Definitionen in bezug auf die korrekte Nutzung der EIA/CDIF-Datentypen für die Meta-Meta-Attribute sowie mit dem Rahmen, den das Meta-Meta-Modell vorgibt, ergaben für das EIA/CDIF-Meta-Modell für die Datenflußmodellierung keinen Fehler, aber zwei Warnungen:

– Die Werte für das Meta-Meta-Attribut „Aliases“[642] für den Meta-Entitätstyp „Attribute“ sind nicht vom EIA/CDIF-Datentyp „Identifier“, da die Bezeichner Leerzeichen beinhalten. Es handelt sich hierbei um die Eintragung „Data Element, Instance Variable, Data Member, Column“, die korrekterweise „DataElement, InstanceVariable, DataMember, Column“ lauten müßte.

– Die Werte für das Meta-Meta-Attribut „Aliases“[643] für den Meta-Entitätstyp „DataFlowModel“ sind nicht vom EIA/CDIF-Datentyp „Identifier“, da die Bezeichner Leerzeichen beinhalten. Es handelt sich hierbei um die Eintragung „DFD Set, Process Model, IDEF0 Model“, die korrekterweise „DFDSet, ProcessModel, IDEF0Model“ lauten müßte.

4.1.3.1.4.3 Generalisierungshierarchie des Meta-Modells

Die folgende Abbildung 4-9 stellt die Generalisierungshierarchie dieses Meta-Modells dar, bei dem aufgrund der Nutzung der Meta-Objekte aus dem fundamentalen GegenstandsBereich „Foundation“ jeweils ein Teilbaum für die Meta-Entitätstypen und einer für die Meta-Beziehungstypen gebildet wird. Die maximale Höhe des Generalisierungsbaumes ist acht, wobei der Zweig für die Meta-Beziehungstypen eine Höhe von vier Ebenen ausmacht. Somit stellt sich der Teilbaum für Meta-Entitätstypen als spezialisierter dar als der für Meta-Beziehungstypen.

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*

642 Vgl. [CDIF94b], Seite 50 oben.

643 Vgl. ebenda.

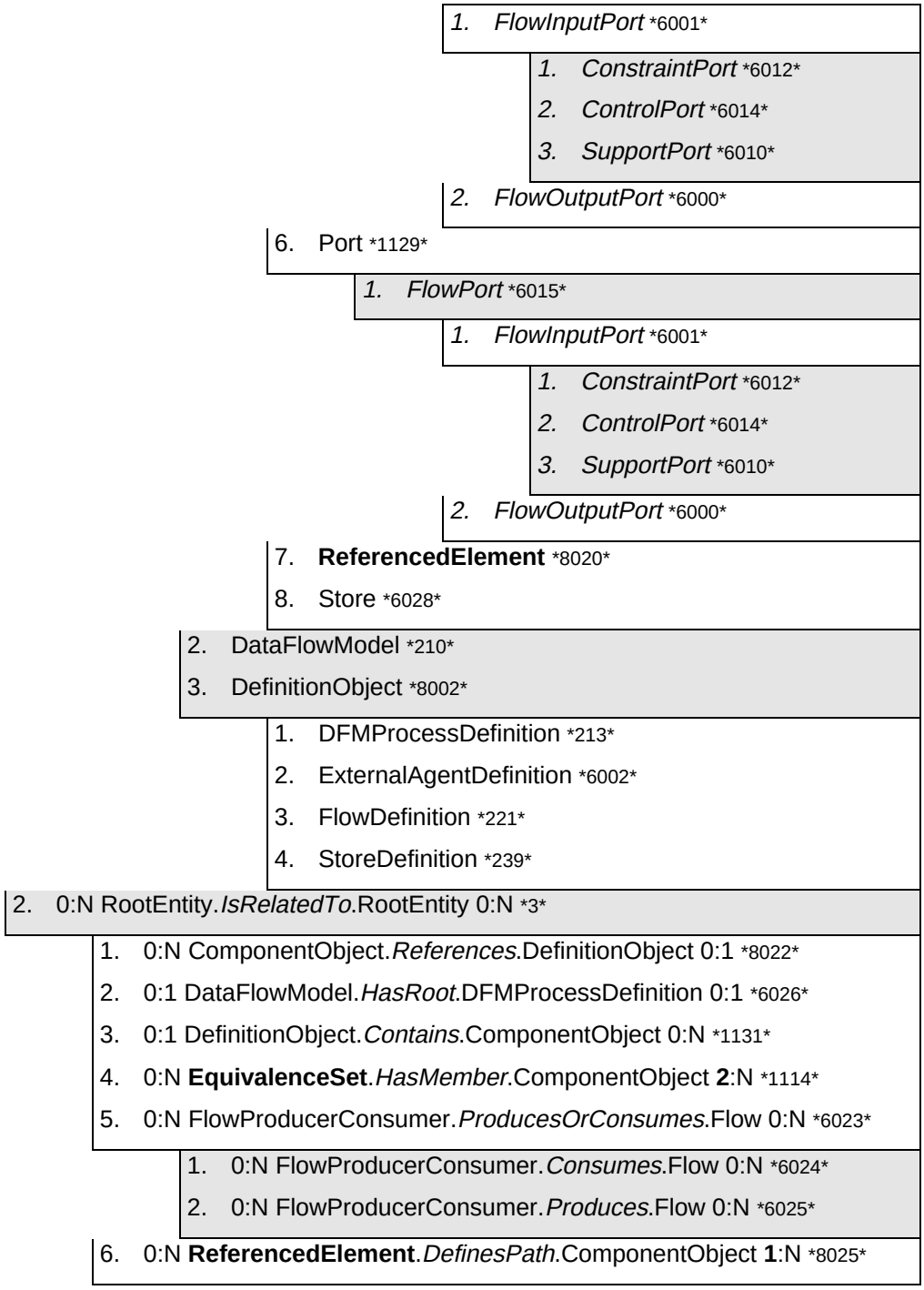


Abbildung 4-8: Hierarchie des Meta-Modells „Data Flow Modeling“[644]

4.1.3.1.4.4 Summarische Darstellung der AttribuierbarenMetaObjekte

Die weitere summarische Darstellung der Meta-Objekte für den Gegenstands-Bereich „Datenflußmodellierung“, kann aus dem Abschnitt 4.2, ‘**Das**

644 Entsprechend den EIA/CDIF-Konventionen weisen kursiv gesetzte Zeilen auf Typen mit Mehrfachvererbung hin.

„Integrierte EIA/CDIF-Meta-Modell“' auf Seite 267ff weiter unten, entnommen werden, in dem sämtliche Meta-Objekte der mit Anfang 1998 standardisierten EIA/CDIF-Meta-Modelle dargestellt sind.

Tabelle 4-64 führt aus Gründen der Übersichtlichkeit jene Meta-Objekte vollqualifiziert an, für die Meta-Attribute definiert sind. Wie in dieser Arbeit üblich, werden die Meta-Objekte mit den festgelegten Werten für das Meta-Meta-Attribut „CDIFMetaIdentifier“ in der EIA/CDIF „ENCODING.1“-Notation[645] für den EIA/CDIF-Datentyp „Identifier“ mit angeführt. Desgleichen werden gemeinsam mit den Meta-Beziehungstypen die entsprechenden Kardinalitäten angeführt.

Vollqualifizierter Name	**Anzahl MA[646]**
Attribute *17*	3/0
DFMProcess *6005*	3/0
DFMProcessDefinition *213*	6/0
DataFlowModel *210*	2/0
DefinitionObject *8002*	4/0
0:1 DefinitionObject.*Contains*.ComponentObject 0:N *1131*	1/0
ExternalAgent *6004*	3/0
Flow *227*	3/0
FlowDefinition *221*	4/0
Port *1129*	2/**1**
0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N *8025*	1/**1**
Store *6028*	3/0
StoreDefinition *239*	8/0

Tabelle 4-64: Meta-Objekte mit Meta-Attributen für den GegenstandsBereich „Datenflußmodellierung“

645 Vgl. in diesem Zusammenhang die Definitionen dafür in [CDIF94e].

646 Die erste Zahl gibt die Summe der Meta-Attribute für das entsprechende Meta-Objekt an, die zweite die Anzahl der darin enthaltenen, zwingend vorgeschriebenen Attribute.

4.1.3.2 DMOD – Das EIA/CDIF-Meta-Modell für den GegenstandsBereich „Datenmodellierung“

[CDIF96b] definiert in Form eines EIA/CDIF-Meta-Modells jene Konzepte, die grundlegend für die wichtigsten Varianten der konzeptionellen Datenmodellierung (englisch: „data modeling“[647]) mit Hilfe der Entity-Relationship-Modellierung sind. Somit soll es theoretisch möglich sein, daß sämtliche konzeptionellen Datenmodelle unabhängig von der konkreten Modellierungsvariante in einem EIA/CDIF-Austausch mit diesem Meta-Modell – und eventuellen Erweiterungen dazu – getauscht werden können.

Die Methode der Datenmodellierung mit Hilfe der Entity-Relationship-Modellierung wurde 1976 in [Chen76] eingeführt. Die Popularität dieser Methode ist ungebrochen und es existieren davon zahlreiche Varianten, die demselben Grundschema folgen, nämlich der Typisierung von Entitäten und der Beziehungen zwischen ihnen, erweitert unter anderem um Generalisierungshierarchien.[648]

In [CDIF96b] werden die grundlegenden Konzepte der Datenmodellierung mit der Entity-Relationship-Methode definiert. Zur Abbildung von (hierarchischen) Strukturen zur Bildung von Einheiten aus Teilen wird auf den Allgemeinen Strukturierungsmechanismus[649] zurückgegriffen. Grundsätzlich sind für die Bildung von DefinitionsObjekten für den Gegenstandsbereich der Datenmodellierung die Subtypen von „ComponentObject“, „Attribute“ und dessen Subtyp „ProjectedAttribute“ festgelegt. Für die Bildung von Strukturen sind die folgenden Subtypen von „DefinitionObject“ definiert:

– „Cluster“ (deutsch: „Haufen“): repräsentiert eine Gruppe von zusammengehörenden *Entitäten* (genauer: Entitätstypen), *Beziehungen* (genauer:

---

647 Dieser GegenstandsBereich wurde während seiner Entwicklung vom EIA/CDIF-Komitee mit dem Akronym „DMOD“ bezeichnet. Vgl. zu den Ausführungen in diesem Abschnitt auch [CDIF96b], das neben den Standardisierungsdefinitionen auch eine ausführliche Übersicht mit umfangreichen Beispielen gibt, so auf den Seiten 17 bis 65 („4. Subject Area Overview“) und 215 bis 230 („Appendix A, Example“).

648 Vgl. hierzu beispielsweise [BaCeNa92], [ElmNav89], [FerSin98], [Hars94], [Hans96], [StaHas97], [Vett90].

649 Vgl. Abschnitt 4.1.3.1.2, „Der Allgemeine Strukturierungsmechanismus“ auf Seite 229ff oben.

Beziehungstypen) und von weiteren *Clustern*,[650] die zusätzlich durch *Attribute* beschrieben werden können,

- „Entity“ (deutsch: „Entität“): repräsentiert eine *Entität*, genauer einen Entitätstyp, dessen Struktur durch *Attribute* gebildet werden kann.
- „Relationship“ (deutsch: „Beziehung“): repräsentiert eine Beziehung, genauer einen Beziehungstyp, dessen Struktur durch *Attribute* gebildet werden kann.
- „Role“ (deutsch: „Rolle“): repräsentiert eine Rolle, die ein oder mehrere *Entitäten* in bezug auf eine bestimmten *Beziehung* spielen, wobei die *Rolle* mit Hilfe von *Attributen* strukturiert werden kann.
- „RolePlayer“ (deutsch: „RollenSpieler“): repräsentiert ein Konzept, das eine bestimmte *Rolle* spielt, und *durch* Attribute strukturiert werden kann.

Obwohl auch der mächtige Allgemeine Strukturierungsmechanismus eingesetzt wird, sind in diesem GegenstandsBereich mit 34 Meta-Beziehungstypen zu 22 Meta-Entitätstypen, mehr Beziehungstypen als Entitätstypen definiert. Dies ist unter sämtlichen standardisierten EIA/CDIF-Meta-Modellen ein herausragendes Merkmal und weist darauf hin, daß es sich um einen komplexen GegenstandsBereich handelt beziehungsweise, daß komplexe Zusammenhänge damit ausgedrückt werden können.

Eine weitere Besonderheit dieses GegenstandsBereiches liegt darin, daß die Definition der Meta-Entitätstypen „Cluster“, „Entity“ und „Relationship“ mit Hilfe der Mehrfachvererbung erfolgt, indem alle drei Meta-Entitätstypen nicht nur direkte oder indirekte Subtypen von „DataModelObject“ sind, sondern zusätzlich auch Subtypen von „DefinitionObject“.[651]

---

650 Diese Festlegung folgt den Erläuterungen in [CDIF96b] auf Seite 54, nachdem in der detaillierten Definition für „Cluster“ ebenda auf Seite 94ff keine Informationen über die zulässigen KomponentenObjekte zu finden sind. Somit wird die Struktur eines Haufens nicht unbedingt durch Attribute beziehungsweise durch AbgeleiteteAttribute gebildet.

651 Im Unterschied dazu werden im GegenstandsBereich „DatenflußModellierung“ (vgl. Abschnitt 4.1.3.1 oben auf Seite 224ff) ausdrücklich Meta-Entitätstypen zur Definition von Strukturen definiert, beispielsweise „FlowDefinition“ oder etwa „StoreDefinition“.
Im GegenstandsBereich „DatenModellierung“ hingegen werden Meta-Entitätstypen aus einem Zweig der Generalisierungshierarchie einfach zu DefinitionsObjekten, das heißt mit dem allgemeinen Strukturierungsmechanismus strukturierbare Meta-Objekten, indem zusätzlich „DefinitionObject“ als direkter Supertyp aufgenommen wird.

Als einziges von allen per Anfang 1998 standardisierten EIA/CDIF-Meta-Modellen weisen die Definitionen einander ausschließende Meta-Beziehungstypen aus. Nachdem derartige Einschränkungen nicht mit Hilfe der Strukturen, die das EIA/CDIF-Meta-Meta-Modell zur Verfügung stellt,[652] dokumentierbar sind, ist man auf die Auswertung der graphischen Darstellungen der verschiedenen Teile des Meta-Modells angewiesen.[653] Folgende drei Gruppen von einander ausschließenden Meta-Beziehungstyppaaren[654] existieren:

- „0:N DataModelSubset.*Excludes*.Attribute 0:N“ und „0:N Attribute.*IsDiscriminatorFor*.SubtypeSetMembershipCriterion 0:N“,[655]
- „0:1 RolePlayer.*RefinesForSubtype*.DataModelObject 0:N“ und „0:1 DataModelObject.*ActsAs*.RolePlayer 0:N“[656] sowie
- „0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N“ und „0:N **ProjectionComponent**.*IsFullProjectionOf*.DefinitionObject **1**:1“.[657]

Für diesen GegenstandsBereich wurden schließlich keine Meta-Objekte definiert, die Meta-Attribute zwingend vorschreiben.[658]

4.1.3.2.1 Kurzcharakterisierung des EIA/CDIF-GegenstandsBereiches „Datenmodellierung“

In diesem Abschnitt werden die Konzepte kurz erläutert, die für den Austausch von Modelldaten aus dem GegenstandsBereich „Datenmodellierung“ vorgesehen sind. Die grundlegenden Konzepte sind:

---

652 Dies wurde bereits weiter oben im Abschnitt 3.4, „Zusammenfassende Diskussion des EIA/CDIF-Meta-Meta-Modells“ auf Seite 129ff, kritisiert.

653 Hierbei handelt es sich um die Diagramme in [CDIF96b], auf den Seiten 60 bis 65.

654 Es ist lediglich ein Zufall, daß es sich hier um Paare handelt. EIA/CDIF schreibt nicht vor, wieviele Meta-Beziehungstypen in einer einander ausschließenden Gruppe enthalten sein dürfen.

655 Vgl. [CDIF96b], Seite 61.

656 Vgl. aaO, Seite 63.

657 Vgl. aaO, Seite 65.

658 Das einzige, zwingend vorgeschriebene Meta-Attribut, ist „CDIFIdentifier“, das indirekt über die Generalisierungshierarchie vom Wurzel-Typ „RootObject“ ererbt wird. (Das Meta-Objekt „RootObject“ wird im GegenstandsBereich „Datenmodellierung“ nicht ausdrücklich benutzt, da es nicht an Instanzen vom Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ enthalten ist.)

– Ein *DatenModell* (englisch: „DataModel“) besteht aus *DatenModellObjekten* (englisch: „DataModelObject“), die entweder vererbbare *Entitäten*[659] (englisch: „Entity“) und *Beziehungen*[660] (englisch: „Relationship“) sind oder *Haufen* (englisch: „Cluster“) darstellen.

– *DatenModellObjekte*[661] spielen *Rollen*[662] (englisch: „Role“), die *Beziehungen* zugeordnet sind, die die *DatenModellObjekte* miteinander assoziieren. Es ist hierbei möglich, daß mehrere *DatenModellObjekte dieselbe Rolle* spielen. Zudem können *Rollen* so beschränkt werden, daß sie einander ausschließen.[663]

– *Cluster, Entitäten, Beziehungen, Rollen* und *RollenSpieler* (englisch: „RolePlayer“) können mit *Attributen*[664] beziehungsweise mit *AbgeleitetenAttributen* strukturiert werden, *Cluster* können darüber hinaus als Bestandteile weitere *Cluster*, *Entitäten* und *Beziehungen* aufweisen.

– *Beziehungen* können „einfach“ sein, indem zwei oder mehrere[665] Entitäten miteinander assoziiert werden, oder „komplex“, indem generell *DatenModellObjekte* miteinander in Beziehung gesetzt werden können.[666]

– *Entitäten* werden zumindest über einen (Primär-) *Schlüssel* (englisch: „*Key*“) identifiziert und können über mehrere *PrimärSchlüsselKandidaten*

---

659 Es handelt sich genauer um einen Meta-Entitäts*typ* mit der Bezeichnung „Entität“.

660 Es handelt sich genauer um einen Meta-Entitäts*typ* mit der Bezeichnung „Beziehung“.

661 In diesem Zusammenhang werden sie daher auch als „RollenSpieler“ bezeichnet.

662 Damit agieren derartige *DatenModellObjekte* als *RollenSpieler*.

663 Dies erfolgt mit Hilfe des Meta-Entitätstyps *RoleConstraint*, der über ein aufzählbares Meta-Attribut „Operator“ mit den möglichen Werten „AND“, „OR“ und „XOR“ aufweist, und die entsprechende Instanziierung des Meta-Beziehungstyps „0:N RoleConstraints.-*Incorporates.*RolePlayer 0:N“. Somit werden die einander ausschließenden *Rollen* mittelbar über die *RollenSpieler* festgelegt, wodurch es sogar möglich wird, Einschränkungen auch für eine Gruppe von *RollenSpielern* zu treffen, die *dieselbe Rolle* spielen.

664 Es handelt sich hierbei um dasselbe Konzept „Attribut“, das bereits im Zusammenhang mit dem GegenstandsBereich „Datenflußmodellierung“ definiert und eingesetzt wurde. Im GegenstandsBereich „Datenmodellierung“ wird darüber hinaus noch auf die Typisierung von Attributen flüchtig eingegangen, indem dafür auf den Entwurf des GegenstandsBereiches „Datendefinition“ verwiesen wird, vgl. hierzu [CDIF97a].

665 Obwohl EIA/CDIF aufgrund des EIA/CDIF-Meta-Meta-Modells nur mit Hilfe von binären Beziehungstypen EIA/CDIF-Meta-Modelle bilden kann, ist es mit diesem Gegenstands-Bereich möglich, für zu tauschende Modelldaten auch nicht-binäre Beziehungstypen zu modellieren.

666 Damit können beispielsweise auch Beziehungen direkt zwischen Beziehungen abgebildet werden, ohne daß es damit notwendig wäre, eigens assoziative Entitätstypen dafür zu definieren, wie dies beispielsweise für die EIA/CDIF-Meta-Modellierung selbst notwendig ist.

(englisch: „CandidateKey“) und über *FremdSchlüssel* (englisch: „Foreign-Key“) aufweisen, die *PrimärSchlüsselKandidaten* in anderen *Entitäten* referenzieren. *PrimärSchlüsselKandidaten* können grundsätzlich durch *Attribute* oder/und *FremdSchlüssel* oder/und *Beziehungen*[667] definiert werden.

– Für *Entitäten* können neben volumetrischen Informationen[668] auch Regeln angegeben werden, die aufgrund von vorhandenen Referenzierungen auf *Entitäten* über *FremdSchlüssel* im Zusammenhang mit Einfüge-, Änderungs- und Löschoperationen berücksichtigt werden müssen.[669]

– *ZugriffsPfade* (englisch: „AccessPath“) auf *Entitäten* können sich aus *Attributen und/*oder aus *Schlüsseln* ergeben.

– Im Zusammenhang mit den strukturellen Eigenschaften der miteinander in *Beziehung* stehenden *DatenModellObjekte* können zwischen äußeren (englisch: „*outer*“) und inneren (englisch: „*inner*“) Kardinalitäten (englisch: „Cardinality“) unterschieden werden. Damit kann exakt festgelegt werden, wieviele Instanzen von den assoziierten Entitätstypen sowie von dem betroffenen Beziehungstyp mindestens existieren müssen beziehungsweise maximal vorhanden sein dürfen.[670] Diese Detaillierungsmöglichkeit ist beispielsweise dann wichtig, wenn es mehrere *RollenSpieler* für eine *Rolle* gibt, wobei die *RollenSpieler* aber über unterschiedliche äußere Kardinalitäten aufweisen.[671]

---

667 Genauer durch einen *RollenSpieler*, das heißt ein *DatenModellObjekt*, das eine bestimmte *Rolle* spielt, die einer bestimmten *Beziehung* zugeordnet ist.

668 „Volumetrisch“ ist der Versuch, das englische Fachwort „volumetric“ einzudeutschen. Es handelt sich hierbei um statistische Angaben über unterschiedliche Ausprägungen eines bestimmten Entitätstyps beziehungsweise von bestimmten Attributen eines InheritableDataModelObject, beispielsweise die minimale/maximale/durchschnittliche Anzahl an Ausprägungen, die Anzahl an durchgeführten Lösch-, Einfüge-, Änderungs-, Leseoperationen, durchschnittliche Zeiten für die Durchführung der entsprechenden Operationen und ähnliches mehr.

669 In [CDIF96b] werden die Regeln hierzu auf den Seite 40 bis 43 und im Zusammenhang mit der detaillierten Definition des Meta-Entitätstyps „RolePlayer“ auf den Seiten 138 bis 155 erläutert. Grundsätzlich können für *Entitäten* mit dem *FremdSchlüssel* als *RollenSpieler* die Effekte „RESTRICTS“ und „CASCADES“ definiert werden. Für *Entitäten* mit dem *PrimärSchlüsselKandidaten* als *RollenSpieler* können für alle Operationen die Effekte „RESTRICTS“ und „CASCADES“ festgelegt werden, für die Änderungs- und Löschoperationen darüber hinaus noch „SETNULL“ und „SETDEFAULT“.

670 Vgl. in diesem Zusammenhang auch die Diskussion über Kardinalitäten in [CDIF96c], Seiten 24 bis 27 und die Erläuterungen im Anhang B auf den Seiten 231 bis 233.

671 Vgl. in diesem Zusammenhang auch [CDIF96b], Seiten 28 bis 29.

– *VererbbareDatenModellObjekte* (englisch: „InheritableDataModelObject“) können abstrakt oder konkret, also instanziierbar, sein, wobei die Subtypbildung für Mengen an Subtypen disjunkt[672] als auch überlappend[673] erfolgen kann.[674]

– *DatenModelle* können in *DatenModellTeilModelle* (englisch: „DataModel-SubSet“) zerlegt werden, *Cluster* hingegen setzen sich aus einer Teilmenge an *Entitäten, Beziehungen* und weiteren *Clustern* sowie *Attributen* zusammen.

Ausgangspunkt der Bildung eines Datenmodells kann der Meta-Entitätstyp *DataModel* sein, der beliebig viele *DatenModellObjekte* mit Hilfe des Meta-Beziehungstyps „0:N DataModel.*Collects*.DataModelObject 0:N“.

4.1.3.2.2 Referenzierte Meta-Objekte

In diesem GegenstandsBereich werden zudem über die Vererbung entlang der Generalisierungshierarchie die Meta-Objekte „RootObject“[675], „RootEntity“[676], „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“[677] sowie „0:N ComponentObject.-*References*.DefinitionObject 0:1“[678] benützt.

4.1.3.2.3 Auswertungen der Definitionen

Zunächst wird in diesem Abschnitt eine strukturelle Übersicht über das Meta-Modell gegeben, indem einfache Auswertungen tabellarisch aufbereitet werden. Daran anschließend werden sämtliche AttribuierbarenMetaObjekte, die aus-

672 Dies wird in [CDIF96b] auch als „exklusiv“ bezeichnet (vgl. ebenda, Seiten 45 bis 46) und durch den Wert „True“ für das Meta-Attribut „IsExclusive“ im Meta-Entitätstyp „SubtypeSet“ vermerkt.

673 Dies wird in [CDIF96b] auch als „orthogonal“ beziehungsweise als „parallel“ bezeichnet (vgl. ebenda, Seiten 46 bis 48) und durch den Wert „False“ für das Meta-Attribut „IsExclusive“ im Meta-Entitätstyp „SubtypeSet“ vermerkt.

674 Es ist zudem möglich, vererbte *Attribute* in ihrem Namen zu verändern, sowie ihren Datentyp einzuschränken. Vererbte *Beziehungen* können in ihren Kardinalitäten weiter eingeschränkt werden, wobei die Minimalwerte erhöht und die Maximalwerte verringert werden dürfen.

675 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.1 auf Seite 158 vorgestellt.

676 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.2 auf Seite 163 vorgestellt.

677 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.3 auf Seite 164 vorgestellt.

678 Dieser Meta-Beziehungstyp wurde weiter oben bereits im Abschnitt 4.1.3.1.2, „Der Allgemeine Strukturierungsmechanismus“, auf Seite 229 vorgestellt.

drücklich im Meta-Modell benutzt werden,[679] in der dem Meta-Modell entsprechenden Generalisierungshierarchie dargestellt, wobei MetaObjekte auf derselben Stufe alphabetisch sortiert sind. Abschließend werden jene AttribuierbarenMetaObjekte in alphabetischer Reihenfolge summarisch angeführt, für die ausdrücklich[680] MetaAttribute definiert sind.

Sämtliche Auflistungen in diesem Abschnitt erfolgen anhand der voll qualifizierten Namen der AttribuierbarenMetaObjekte, wobei die Namen von Meta-Beziehungstypen immer kursiv gesetzt sind. Die Namen von Meta-Entitätstypen, deren Instanzen zwingend in Beziehungen auftreten müssen, werden immer fett hervorgehoben. Die Kardinalitäten der Meta-Beziehungstypen werden ausdrücklich angeführt, sowie die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“[681]. Sofern für AttribuierbareMetaObjekte die Mehrfachvererbung aufgrund der Definition vorgesehen ist, wird der gesamte vollqualifizierte Namen kursiv gesetzt. Sämtliche lokal definierten Meta-Attribute werden gemeinsam mit den entsprechenden EIA/CDIF-Datentypen in kleiner Schrift angeführt.[682]

4.1.3.2.3.1 Strukturelle Übersicht

Das Meta-Modell „Data Modeling“ definiert insgesamt 146 SammelbareMeta-Objekte, wobei die einzige Instanz „RootObject“ des Meta-Meta-Entitätstyps AttribuierbaresMetaObjekt, der Meta-Entitätstyp „RootEntity“ sowie die Meta-Beziehungstypen „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ und „0:N ComponentObject.*References*.DefinitionObject 0:1“, die alle über die Generalisierungshierarchie eingebunden werden, mitgezählt sind. Tabelle 4-65 stellt die Aufstellung dieser SammelbarenMetaObjekte tabellarisch dar.

---

679 Es handelt sich also um Instanzen jener SammelbarenMetaObjekte, die in Instanzen des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ für die Instanz des entsprechenden GegenstandBereiches enthalten sind.

680 Derartige Meta-Attribute werden in den EIA/CDIF-Standards auch als „lokal“ bezeichnet, um sie damit von über die Generalisierungshierarchie ererbten Meta-Attribute unterscheidbar zu machen.

681 Die Darstellung erfolgt in der vorgesehenen Verkodierung des EIA/CDIF-Datentyps „Identifier“, indem der Wert in Sterne eingefaßt wird (vgl. [CDIF94e], Seiten 11 und 20 unten, „IdentifierValue“).

682 Im Falle des aufzählbaren Datentyps werden auch die gültigen Werte der entsprechenden Wertemenge dargestellt.

Anzahl Instanzen vom Typ AMO[683]	1
Anzahl Instanzen vom Typ ME[684]	23
Anzahl Instanzen vom Typ MR[685]	36
Anzahl Instanzen vom Typ MA[686]	86
Anzahl (Summe) Instanzen vom Typ CMO	146

Tabelle 4-65: Verteilung der sammelbaren Objekte des Meta-Modells „Data Modeling“

Wie bereits erwähnt, zeichnet sich der GegenstandsBereich „Datenmodellierung“ dadurch aus, daß in ihm wesentlich mehr Meta-Beziehungstypen als Meta-Entitätstypen definiert sind. Dies wurde darauf zurückgeführt, daß es sich hierbei um eine sehr entwickelte und damit komplexe Methode handelt. Hinzu kommt mit 86[687] eine signifikant hohe Zahl[688] an Meta-Attributen hinzu, die für Meta-Beziehungstypen und Meta-Entitätstypen festgelegt sind.

Tabelle 4-66 führt weitere Analyseergebnisse über die strukturellen Eigenschaften dieses Meta-Modells an, wobei die referenzierten AttribuierbarenMetaObjekte darin nicht mehr enthalten sind.

683 Hier wird die einzige, direkte Instanz von AMO angeführt, nämlich „RootObject“.

684 Die beiden referenzierten Meta-Entitätstypen „RootEntity“ *und* „SemanticInformationObject“ sind in der Summe beinhaltet.

685 Der referenzierte Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“ ist in der Summe eingerechnet.

686 Es sind insgesamt sieben Meta-Attribute mitberücksichtigt, die von den referenzierten AttribuierbarenMetaObjekten stammen. Es handelt sich hierbei um die Meta-Attribute von RootObject („CDIFIdentifier“, „DateCreated“, „DateUpdated“, „TimeCreated“, „TimeUpdated“) und von SemanticInformationObject („BriefDescription“, „FullDescription“).

687 Ohne die Meta-Attribute der referenzierten Meta-Objekte handelt es sich um 81 Meta-Attribute, die ausdrücklich für diesen GegenstandsBereich definiert sind.

688 In [CDIF96c] sind es beispielsweise 43, in [CDIF96d] 30, in [CDIF96a] 20 und in [CDIF94f] fünf Meta-Attribute.

Anzahl zwingend vorgeschriebener MA		0
Anzahl optionaler MA		81
Meta-Objekte mit/ohne Meta-Attribute	mit MA	ohne MA
Anzahl Instanzen vom Typ AMO	0	0
Anzahl Instanzen vom Typ ME	18	4
Anzahl Instanzen vom Typ MR	7	27
Anzahl von zwingend vorgeschriebenen MR[689]		14
Anzahl von ME, die an zwingend vorgeschriebenen MR teilhaben		10
Anzahl der Blätter (Instanzen vom Typ AMO) im Meta-Modellbaum[690]		48
Anzahl von AMOs mit Mehrfachvererbung[691]		3

Tabelle 4-66: Strukturelle Übersicht über das Meta-Modell für „Data Modeling“[692]

Es fällt auf, daß in diesem GegenstandsBereich zwar 81 Meta-Attribute definiert wurden, die aber *alle* optional sind. Immerhin sieben und damit fast ein Drittel aller Meta-Beziehungstypen verfügen über Meta-Attribute, sowie etwas mehr als 80 % aller Meta-Entitätstypen.

Wie im Gegenstandsbereich „Common“ wird auch hier von der Möglichkeit Gebrauch gemacht, Meta-Beziehungstypen so zu definieren, daß Instanzen von damit in Beziehung gesetzten Meta-Entitätstypen in jedem Fall in Instanzen des Meta-Beziehungstyps enthalten sein müssen. Es handelt sich um insgesamt 14 Meta-Beziehungstypen, die im folgenden kurz charakterisiert werden:

689 Zwingend vorgeschriebene“ Meta-Beziehungstypen verfügen über einen Wert >=1 in mindestens einer der minimalen Kardinalitäten. Aufgrund der gegenüberliegenden Anschrift der Kardinalitäten im graphischen Modell bedeutet dies, daß ein Wert >=1 im Meta-Meta-Attribut „Min*Source*Card“ zur Folge hat, daß Instanzen des *Ziel*-Meta-Entitätstyps an Instanzen dieses Meta-Beziehungstyps teilnehmen müssen. Entsprechend müssen bei einem Wert >=1 im Meta-Meta-Attribut „Min*Dest*Card“ Instanzen des *Quell*-Meta-Entitätstyp an Instanzen des Meta-Beziehungstyps enthalten sein.

690 Blattypen, die einer Mehrfachvererbung unterworfen sind, werden lediglich einmal gezählt.

691 Hier werden nur jene Meta-Entitätstypen gezählt, für die die Mehrfachvererbung ausdrücklich festgelegt ist. Selbstverständlich sind aufgrund der Vererbung entlang der Generalisierungshierarchie auch sämtliche Subtypen der Mehrfachvererbung unterworfen.

692 In dieser Tabelle werden referenzierte Meta-Objekte nicht berücksichtigt.

- „**1**:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N“: Instanzen vom Typ *RolePlayer* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps partizipieren.
- „0:N **DataModelSubset**.*IsSubsetOf*.DataModel **1**:1“: Instanzen vom Typ *DatenModellTeilMenge* müssen exakt einmal in einer Instanz dieses Meta-Beziehungstyps enthalten sein.
- „**1**:1 DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N“: Instanzen vom Typ *ProjectionComponent* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps teilhaben.
- „**1**:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N“: Instanzen vom Typ *ZugriffsPfad* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps partizipieren.
- „**1**:1 Entity.*IsIdentifiedBy*.**Key** 0:N“: Instanzen vom Typ *Key* müssen exakt einmal in einer Instanz dieses Meta-Beziehungstyps enthalten sein.
- „0:N **ForeignKey**.*References*.CandidateKey **1**:1“: Instanzen vom Typ *FremdSchlüssel* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps teilhaben.
- „**1**:N InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N“: Instanzen vom Typ *SubtypeSet* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps partizipieren.
- „**1**:1 InheritableDataModelObject.*IsSupertypeFor*.**SubtypeSet** 0:N“: Instanzen vom Typ *SubtypeSet* müssen exakt einmal in einer Instanz dieses Meta-Beziehungstyps enthalten sein.
- „0:N **ProjectionComponent**.*IsFullProjectionOf*.DefinitionObject **1**:1“: Instanzen vom Typ *ProjectionComponent* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps partizipieren.
- „0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N“: Instanzen vom Typ *ProjectionComponent* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps teilhaben.

- „**2**:N **Role**.*BelongsTo*.**Relationship 1**:1“: Instanzen vom Typ *Relationship* müssen *mindestens* in *zwei* Instanzen dieses Meta-Beziehungstyps enthalten sein. Gleichzeitig gilt, daß Instanzen vom Typ *Rolle* exakt einer Instanz von *Beziehung* zugeordnet sind.
- „**1**:N RolePlayer.*Plays*.**Role** 0:1“: Instanzen vom Typ *Role* müssen exakt einmal an einer Instanz dieses Meta-Beziehungstyps partizipieren.
- „**1**:1 SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion** 0:N“: Instanzen vom Typ *SubtypeSetMembershipCriterion* muß exakt einmal an einer Instanz dieses Meta-Beziehungstyps teilhaben.
- „0:N **SubtypeSetMembershipCriterion**.*Selects*.InheritableDataModelObject **1**:1“: Instanzen vom Typ *SubtypeSetMembership* müssen exakt einmal in einer Instanz dieses Meta-Beziehungstyps partizipieren.

Von den insgesamt 48 Blättern in diesem Meta-Modellbaum sind 33 Meta-Beziehungstypen und 15 Meta-Entitätstypen.

4.1.3.2.3.2 Aufgefundene Fehler

Die im Rahmen dieser Arbeit erfolgte Analyse der Definitionen in bezug auf die korrekte Nutzung der EIA/CDIF-Datentypen für die Meta-Meta-Attribute sowie mit dem Rahmen, den das Meta-Meta-Modell vorgibt, ergaben für das EIA/CDIF-Meta-Modell für die Datenmodellierung *drei* Fehler und fünf Warnungen:

- Fehler: Der Surrogatwert „1140“ für das Meta-Meta-Attribut „CDIFMetaIdentifier“ wurde *zweimal* vergeben, sodaß dadurch die wesentliche Grundeigenschaft der Eineindeutigkeit für Surrogate nicht mehr gegeben ist. Als Folge davon ist es anhand dieses Wertes nicht mehr möglich, das Meta-Attribut „Key.*SpecificationText*“ vom Meta-Attribut „DefinitionObject.*SpecificationText*“ zu unterscheiden, da beide Meta-Objekte den Wert „1140“ im Meta-Meta-Attribut „CDIFMetaIdentifier“ aufweisen.
- Fehler: Der Surrogatwert „1733“ für das Meta-Meta-Attribut „CDIFMetaIdentifier“ wurde *zweimal* vergeben, sodaß dadurch die wesentliche Grundeigenschaft der Eineindeutigkeit für Surrogate nicht mehr gegeben ist. Als Folge davon ist es anhand dieses Wertes nicht mehr möglich, den

Meta-Beziehungstyp „0:N Key.*Incorporates*.SemanticInformationObject 0:N“ vom Meta-Beziehungstyp „0:N Key.*Incorporates*.Attribute 0:N“ zu unterscheiden, da beide Meta-Objekte den Wert „1733“ im Meta-Meta-Attribut „CDIFMetaIdentifier“ aufweisen.

– Fehler: Der Surrogatwert „1736“ für das Meta-Meta-Attribut „CDIFMeta-Identifier“ wurde *zweimal* vergeben, sodaß dadurch die wesentliche Grundeigenschaft der Eineindeutigkeit für Surrogate nicht mehr gegeben ist. Als Folge davon ist es anhand dieses Wertes nicht mehr möglich, das Meta-Attribut „InheritableDataModelObject.*IsSubtypeIn***.SubtypeSet**.*Store-WithSupertype*“ vom Meta-Beziehungstyp „1:1 SubtypeSet.*Specifies*.**-SubtypeSetMembershipCriterion** 0:N“ zu unterscheiden, da beide Meta-Objekte den Wert „1736“ im Meta-Meta-Attribut „CDIFMetaIdentifier“ aufweisen.

– Warnung: In [CDIF96b] auf Seite 85 wird das Meta-Meta-Attribut des Meta-Meta-Objekts „SubjectArea“, „VersionNumber“, fälschlicherweise als „VersionNo“ angegeben.

– Warnung: Die Werteliste für das Meta-Meta-Attribut „Aliases“[693] für das Meta-Attribut „SemanticInformationObject.*FullDescription*“ wird mit einem Punkt abgeschlossen.

– Warnung: Die Werte für das Meta-Meta-Attribut „Aliases“[694] für den Meta-Entitätstyp „AccessPath“ sind nicht vom EIA/CDIF-Datentyp „Identifier“, da sie Leerzeichen beinhalten. Es handelt sich hierbei um die Eintragung „Alternate Key, Index, Multi-valued Key, Duplicate Key“, die korrekterweise „AlternateKey, Index, Multi-valuedKey, DuplicateKey“ lauten müßte.

– Warnung: Die Werte für das Meta-Meta-Attribut „Aliases“[695] für den Meta-Entitätstyp „Attribute“ sind nicht vom EIA/CDIF-Datentyp „Identifier“, da sie Leerzeichen beinhalten. Es handelt sich hierbei um die Eintragung „Data

693 Vgl. [CDIF94b], Seite 50 oben, Beschreibung von „Constraints“, in der angegeben wird, daß es sich um eine durch Kommata separierte Liste von Elementen vom Typ EIA/CDIF „Identifier“ handeln muß.

694 Vgl. [CDIF94b], Seite 50 oben.

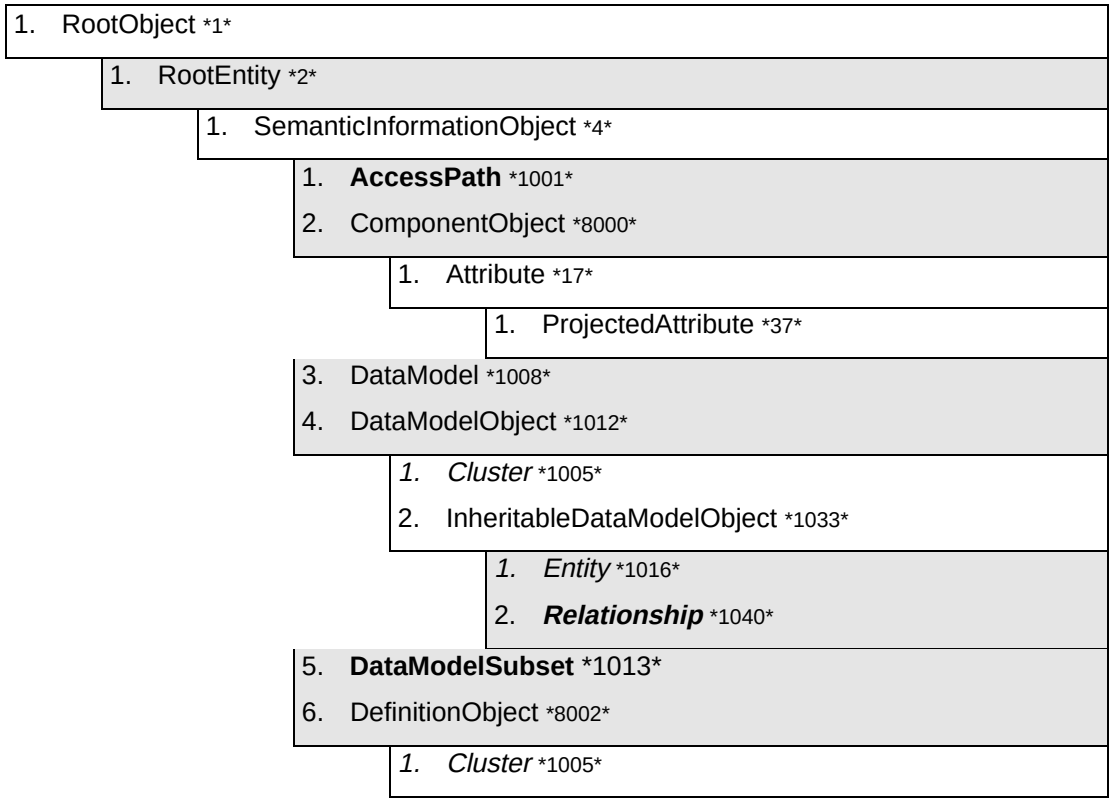
695 Vgl. [CDIF94b], Seite 50 oben.

Element, Instance Variable, Data Member, Column“, die korrekterweise „DataElement, InstanceVariable, DataMember, Column“ lauten müßte.

– Warnung: Die Werte für das Meta-Meta-Attribut „Aliases“[696] für den Meta-Entitätstyp „ProjectedAttribute“ sind nicht vom EIA/CDIF-Datentyp „Identifier“, da sie Leerzeichen beinhalten. Es handelt sich hierbei um die Eintragung „View Element, View Column, Derived Attribute“, die korrekterweise „ViewElement, ViewColumn, DerivedAttribute“ lauten müßte.

4.1.3.2.3.3 Generalisierungshierarchie des Meta-Modells

Die folgende Abbildung 4-9 stellt die Generalisierungshierarchie dieses Meta-Modells dar, bei dem aufgrund der Nutzung der Meta-Objekte aus dem fundamentalen GegenstandsBereich „Foundation“ jeweils ein Teilbaum für die Meta-Entitätstypen und einer für die Meta-Beziehungstypen gebildet wird. Die maximale Höhe des Generalisierungsbaumes ist acht, wobei der Zweig für die Meta-Beziehungstypen eine Höhe von vier Ebenen ausmacht. Somit stellt sich der Teilbaum für Meta-Entitätstypen als spezialisierter dar als der für Meta-Beziehungstypen.



696 Vgl. [CDIF94b], Seite 50 oben.

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*.SubtypeSetMembershipCriterion 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*.InheritableDataModel-Object **1**:1 *1737*

Abbildung 4-9: Generalisierungshierarchie des Meta-Modells „DataModeling“[697]

4.1.3.2.3.4 Summarische Darstellung der AttribuierbarenMetaObjekte

Die weitere summarische Darstellung der Meta-Objekte für den Gegenstands-Bereich „Datenmodellierung“, findet sich im Abschnitt 4.2, ‘Das „Integrierte EIA/CDIF-Meta-Modell“’ auf Seite 267ff, in dem sämtliche Meta-Objekte der bis Anfang 1998 standardisierten EIA/CDIF-Meta-Modelle enthalten sind.

Tabelle 4-67 führt aus Gründen der Übersichtlichkeit jene Meta-Objekte vollqualifiziert an, für die Meta-Attribute definiert sind. Wie in dieser Arbeit üblich, werden die Meta-Objekte mit den festgelegten Werten für das Meta-Meta-Attribut „CDIFMetaIdentifier“ in der EIA/CDIF-„ENCODING.1“-Notation[698] für den EIA/CDIF-Datentyp „Identifier“ mit angeführt. Desgleichen werden gemeinsam mit den Meta-Beziehungstypen die entsprechenden Kardinalitäten dargestellt.

Vollqualifizierter Name	**Anzahl MA[699]**
AccessPath *1001*	3/0
0:N AccessPath.*Incorporates.*Attribute 0:N *1716*	2/0

697 Entsprechend den EIA/CDIF-Konvention weisen kursiv gesetzte Zeilen auf Typen mit Mehrfachvererbung hin.

698 Vgl. in diesem Zusammenhang die Definitionen dafür in [CDIF94e].

699 Die erste Zahl gibt die Summe der Meta-Attribute des entsprechenden Meta-Objekts an, die zweite Zahl die Anzahl an zwingend vorgeschriebenen Attributen.

Vollqualifizierter Name	**Anzahl MA[699]**
Attribute *17*	3/0
CandidateKey *1003*	1/0
DataModel *1008*	2/0
DataModelSubset *1013*	1/0
DefinitionObject *8002*	4/0
0:1 DefinitionObject.*Contains*.ComponentObject 0:N *1131*	1/0
1:1 DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N *1709*	1/0
Entity *1016*	14/0
InheritableDataModelObject *1033*	1/0
1:N InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N *1715*	3/0
Key *1035*	3/0
0:N Key.*Incorporates*.Attribute 0:N *1733*	1/0
0:N Key.*Incorporates*.SemanticInformationObject 0:N *1733_Duplicate_2*	1/0
ProjectedAttribute *37*	2/0
ProjectionComponent *1036*	3/0
0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N *1722*	1/0
Relationship *1040*	1/0
Role *1042*	2/0
RoleConstraint *1045*	2/0
RolePlayer *1048*	21/0
SemanticInformationObject *4*	2/0
SubtypeSet *1070*	3/0
SubtypeSetMembershipCriterion *1074*	3/0

Tabelle 4-67: Meta-Objekte mit Meta-Attributen für den GegenstandsBereich „Datenmodellierung“

4.1.3.3 PLAC – Das EIA/CDIF-Meta-Modell „Presentation Location and Connectivity“

[CDIF96d] definiert in Form eines EIA/CDIF-Meta-Modells jene Konzepte, die notwendig sind, um auf Knoten und Kanten zurückführbare graphische Darstellungen mit Hilfe eines EIA/CDIF-Austausches zu transportieren. Die Bezeichnung „Presentation Location and Connectivity“ (deutsch: „Darstellungsorte und Verbindungen“) dieses GegenstandsBereiches deutet dies bereits an.[700]

Die Aufgabe von „PLAC“[701] liegt darin, die Knoten und Kanten von graphischen Repräsentationen von Modellen in ihrer Anordnung in einem dreidimensionalen Raum[702] aus Werkzeugen zu exportieren beziehungsweise in sie zu importieren. Hierbei wird ausdrücklich von der graphischen Repräsentation abstrahiert, sodaß keine Informationen bezüglich der konkreten Darstellung der Knoten und Kanten, also über die Form und Farbgebung oder der Erzeugung derselben gegeben wird. Die graphische Gestaltung von Knoten und Kanten hängen daher von den Werkzeugen ab.[703] Es ist darüber hinaus vorgesehen, daß die „GraphischenElemente“ Knoten und Kanten transformierbar sind, sowie mit absoluten Punkten oder in zueinander relativ gesetzten Punkten ihren Ort im dreidimensionalen EIA/CDIF-Koordinatenschema zugewiesen erhalten.

Dieser GegenstandsBereich weist zwei bemerkenswerte Besonderheiten auf:

---

700 Dieser GegenstandsBereich wurde während seiner Entwicklung vom EIA/CDIF-Komitee mit dem Akronym „PLAC“ bezeichnet. Vgl. zu den Ausführungen in diesem Abschnitt auch [CDIF96d], das neben den Standardisierungsdefinitionen auch eine ausführliche Übersicht mit umfangreichen Beispielen gibt, so auf den Seiten 15 bis 30.

701 Sowohl aufgrund der langen Bezeichnung dieses GegenstandsBereiches im Englischen, als auch im Deutschen, wird in diesem Abschnitt das Akronym „PLAC“ stattdessen benutzt.

702 Es handelt sich hierbei um den von EIA/CDIF-Definierten Raster aus den Achsen x, y und z. Der Nullpunkt, (0, 0, 0) beschreibt die linke, obere und vordere Ecke. Positive x erstrecken sich nach rechts, positive y nach unten und positive z nach hinten. Derartige dreidimensionale Koordinaten werden von EIA/CDIF mit Hilfe des EIA/CDIF-Datentyps „Point“ angegeben.

703 Somit ist es denkbar, daß beispielsweise Entitätstypen (Beziehungstypen) in einem Werkzeug in der Form von Rechtecken (als gerichtete Pfeile) mit spitzen Ecken und in einem anderen in Form von Rechtecken mit abgerundeten Ecken (als gerichtete Pfeile mit einem auf dem Kopf stehenden Trapez in der Mitte) repräsentiert werden. Die räumliche Anordnung der Knoten zueiander, in diesem Beispiel sind dies die Rechtecke, ist aufgrund von PLAC in beiden Werkzeugen dieselbe; auch die Information, welche Kanten die Knoten wie verbinden, kann von den Werkzeugen entsprechend extrahiert werden.

– die Modelldaten, die ausgetauscht werden, sind selbst in der Regel nicht mehr instanziierbar[704] und

– jedes „PresentationInformationObject“ kann mit Hilfe von Instanzen vom Typ „SemanticObjectReference“[705, 706] mit Instanzen vom Typ „SammelbareMetaObjekte“ über Instanzen des Meta-Beziehungstyps „0:N PresentationInformationObject.*Represents*.SemanticObjectReference 0:N“ in Beziehung gesetzt werden.

Der allgemeine Strukturierungsmechanismus wird für diesen GegenstandsBereich nicht verwendet, in dem 14 Meta-Entitätstypen und 11 Meta-Beziehungstypen ausdrücklich definiert sind. Es ist auffallend, daß sieben Meta-Beziehungstypen etwa 65 % aller Beziehungstypen, so definiert sind, daß zwingend Instanzen von insgesamt sechs (von insgesamt 14, etwa 43 %) Meta-Entitätstypen darin vorkommen müssen.

Eine weitere Besonderheit dieses GegenstandsBereiches liegt darin, daß insgesamt drei AttribuierbareMetaObjekte über zwingend vorgeschriebene Meta-Attribute aufweisen, mehr als jedes andere standardisierte EIA/CDIF-Meta-Modell.[707]

---

704 Im Gegensatz dazu repräsentieren die Modelldaten eines EIA/CDIF-Austausches mit Hilfe des Meta-Modells für den GegenstandsBereich „Datenmodellierung“ üblicherweise instanziierbare Konzepte. Vgl. hierzu auch Abschnitt 2.3.2, „Ein beispielhafter EIA/CDIF-Austausch“ auf Seite 40ff.

705 Dieser Meta-Entitätstyp weist zwei Meta-Attribute auf, „ReferencedMetaObjectInstance“ und „ReferencedMetaAttributeName“. Das zwingend vorgeschriebene Meta-Attribut „ReferencedMetaObjectInstance“ nimmt als Wert den Surrgatwert aus dem Meta-Attribut „CDIFIdentifier“ jenes AttribuierbarenMetaObjekts auf, das referenziert wird. Sofern stattdessen ein bestimmtes Meta-Attribut darin referenziert werden soll, muß der Name des entsprechenden Meta-Attributs im optionalen Meta-Attribut „ReferencedMetaAttributeName“ mit angegeben werden

706 Die Bezeichnung „SemanticObjectReference“ legt nahe, daß es sich hierbei um eine Referenz auf Instanzen vom Typ „SemanticInformationObject“ handelt, was nicht der Fall ist. Vielmehr können damit *alle* MetaObjekte über den im Meta-Attribut „CDIFIdentifier“ gespeicherten Surrogatwert referenziert werden (also alle Instanzen vom Typ „RootObject“).

707 Damit ist dieser GegenstandsBereich für etwa 38 % aller zwingend vorgeschriebener Meta-Attribute verantwortlich, obwohl in ihm im Vergleich zum Meta-Modell für die Datenmodellierung (vgl. Abschnitt 4.1.3.2, „DMOD – Das EIA/CDIF-Meta-Modell für den GegenstandsBereich „Datenmodellierung“ auf Seite 241ff oben) relativ wenige AttribuierbareMetaObjekte definiert werden. Vgl. hierzu auch die Ausführungen im Abschnitt 4.2, ‘Das „Integrierte EIA/CDIF-Meta-Modell“’ auf Seite 267ff unten.

4.1.3.3.1 Kurzcharakterisierung des EIA/CDIF-GegenstandsBereiches „PLAC“

In diesem Abschnitt werden die Konzepte kurz erläutert, die für den Austausch von Modelldaten aus dem GegenstandsBereich „PLAC“ vorgesehen sind. Die grundlegenden Konzepte sind:

- Ein *Diagramm* (englisch: „Diagram“) weist *GraphischeElemente* (englisch: „GraphicalElement“) auf, nämlich *Knoten* (englisch: „Node“) und *Kanten* (englisch: „Edge“), die selbst in bezug auf *Punkte* (englisch: „Point“) – *AbsolutePunkte* (englisch: „AbsolutePoint“) und *RelativePunkte* (englisch: „RelativePoint“) in bezug auf andere *Punkte* – in einem dreidimensionalen Raum angeordnet werden.
- *Kanten* verbinden *Knoten* miteinander, und können aus mehreren, zusammenhängenden *KantenElementen* (englisch: „EdgeElement“) bestehen.
- Instanzen vom Typ *Annotation*[708] (englisch: „Annotation“) können aus beliebigen *BeschreibungsArgumenten* (englisch: „AnnotationArgument“) zusammengesetzt sein und können entweder über den Beziehungstyp „0:N PresentationInformationObject.*Represents*.SemanticObjectReference 0:N“ Instanzen vom Typ „RootObject“ zugeordnet werden oder auf MIME-kodierte Daten[709] verweisen, die in den folgenden lokalen Meta-Attributen von *AnnotationArgument* vorgehalten werden: „DataBlock“, „MIME-Subtype“ und „MIMEType“.

4.1.3.3.2 Referenzierte Meta-Objekte

In diesem GegenstandsBereich werden zudem über die Vererbung entlang der Generalisierungshierarchie die fundamentalen Meta-Objekte „RootObject“[710], „RootEntity“[711] sowie „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“[712] benützt.

---

708 *Beschreibungen* können ebenso wie *Knoten* im dreidimensionalen Raum frei plaziert werden, da sie beide denselben Supertyp, nämlich „PositionedElement“ aufweisen.

709 Vgl. RFC 1521, „MIME“.

710 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.1 auf Seite 158 vorgestellt.

711 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.2 auf Seite 163 vorgestellt.

712 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.3 auf Seite 164 vorgestellt.

4.1.3.3.3 Auswertungen der Definitionen

Zunächst wird in diesem Abschnitt eine strukturelle Übersicht über das Meta-Modell gegeben, indem einfache Auswertungen tabellarisch aufbereitet werden. Daran anschließend werden sämtliche AttribuierbarenMetaObjekte, die ausdrücklich im Meta-Modell benutzt werden,[713] in der dem Meta-Modell entsprechenden Generalisierungshierarchie dargestellt, wobei MetaObjekte auf derselben Stufe alphabetisch sortiert sind. Abschließend werden jene AttribuierbarenMetaObjekte in alphabetischer Reihenfolge summarisch angeführt, für die ausdrücklich[714] MetaAttribute definiert sind.

Sämtliche Auflistungen in diesem Abschnitt erfolgen anhand der voll qualifizierten Namen der AttribuierbarenMetaObjekte, wobei die Namen von Meta-Beziehungstypen immer kursiv gesetzt sind. Die Namen von Meta-Entitätstypen, deren Instanzen zwingend in Beziehungen auftreten müssen, werden immer fett hervorgehoben. Die Kardinalitäten der Meta-Beziehungstypen werden ausdrücklich angeführt, sowie die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“[715]. Sofern für AttribuierbareMetaObjekte die Mehrfachvererbung aufgrund der Definition vorgesehen ist, wird der gesamte vollqualifizierte Namen kursiv gesetzt. Sämtliche lokal definierten Meta-Attribute werden angeführt, wobei zusätzlich ihr EIA/CDIF-Datentyp in kleiner Schrift mitangegeben ist.[716]

4.1.3.3.3.1 Strukturelle Übersicht

Das Meta-Modell „PLAC“ definiert insgesamt 61 SammelbareMetaObjekte, wobei die einzige Instanz „RootObject“ des Meta-Meta-Entitätstyps „AttribuierbaresMetaObjekt“, der Meta-Entitätstyp „RootEntity“ und der Meta-Beziehungstyp „RootEntity.*IsRelatedTo*.RootEntity“, die alle über die Generalisierungs-

---

713 Es handelt sich also um Instanzen jener SammelbarenMetaObjekte, die in Instanzen des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ für die Instanz des entsprechenden GegenstandBereiches enthalten sind.

714 Derartige Meta-Attribute werden in den EIA/CDIF-Standards auch als „lokal“ bezeichnet, um sie damit von über die Generalisierungshierarchie ererbten Meta-Attribute unterscheidbar zu machen.

715 Die Darstellung erfolgt in der vorgesehenen Verkodierung des EIA/CDIF-Datentyps „Identifier“, indem der Wert in Sterne eingefaßt wird (vgl. [CDIF94e], Seiten 11 und 20 unten, „IdentifierValue“).

716 Im Falle des aufzählbaren Datentyps werden auch die gültigen Werte der entsprechenden Wertemenge dargestellt.

hierarchie eingebunden werden, mitgezählt sind. Tabelle 4-68 stellt die Aufstellung dieser SammelbarenMetaObjekte tabellarisch dar.

Anzahl Instanzen vom Typ AMO[717]	1
Anzahl Instanzen vom Typ ME[718]	14
Anzahl Instanzen vom Typ MR[719]	11
Anzahl Instanzen vom Typ MA[720]	35
Anzahl (Summe) Instanzen vom Typ CMO	61

Tabelle 4-68: Verteilung der sammelbaren Objekte des Meta-Modells „PLAC“

Tabelle 4-69 führt weitere Analyseergebnisse über die strukturellen Eigenschaften dieses Meta-Modells an, wobei die referenzierten AttribuierbarenMetaObjekte darin nicht mehr enthalten sind.

717 Hier wird die einzige direkte Instanz von AMO angeführt, nämlich „RootObject“.

718 Der referenzierte Meta-Entitätstyp „RootEntity“ ist mitgerechnet.

719 Der referenzierte Meta-Beziehungstyp „RootEntity.IsRelatedTo.RootEntity“ ist in der Summe eingerechnet.

720 Es sind insgesamt fünf Meta-Attribute mitberücksichtigt, die von den referenzierten AttribuierbarenMetaObjekten stammen. Es handelt sich hierbei um die Meta-Attribute von RootObject („CDIFIdentifier“, „DateCreated“, „DateUpdated“, „TimeCreated“, „TimeUpdated“).

Anzahl zwingend vorgeschriebener MA		4
Anzahl optionaler MA		26
Meta-Objekte mit/ohne Meta-Attribute	mit MA	ohne MA
Anzahl Instanzen vom Typ AMO	0	0
Anzahl Instanzen vom Typ ME	7	6
Anzahl Instanzen vom Typ MR	2	8
Anzahl von zwingend vorgeschriebenen MR[721]		7
Anzahl von ME, die an zwingend vorgeschriebenen MR teilhaben		6
Anzahl der Blätter (Instanzen vom Typ AMO) im Meta-Modellbaum		19
Anzahl von AMOs mit Mehrfachvererbung		0

Tabelle 4-69: Strukturelle Übersicht über das Meta-Modell für „PLAC“[722]

Etwas mehr als die Hälfte aller Meta-Entitätstypen und ein Fünftel der Meta-Beziehungstypen weisen Meta-Attribute auf. Es fällt auf, daß vier Meta-Attribute zwingend vorgeschrieben sind.[723] Es handelt sich hierbei um die folgenden vier Meta-Attribute, die voll qualifiziert werden:

– AbsolutePoint.**Position**,

– **RelativePoint**.**DeltaY**,

– **RelativePoint**.**DeltaX** und

– SemanticObjectReference.**ReferencedMetaObjectInstance**.

Damit müssen Instanzen für die drei Meta-Entitätstypen „AbsolutePoint“, „RelativePoint“ und „SemanticObjectReference“ in einem EIA/CDIF-Austausch Werte in den zwingend vorgeschriebenen Meta-Attributen aufweisen.

---

721 Zwingend vorgeschriebene Meta-Beziehungstypen verfügen über einen Wert >=1 in mindestens einer der minimalen Kardinalitäten. Aufgrund der gegenüberliegenden Anschrift der Kardinalitäten im graphischen Modell bedeutet dies, daß ein Wert >=1 im Meta-Meta-Attribut „Min*Source*Card“ zur Folge hat, daß Instanzen des *Ziel*-Meta-Entitätstyps an Instanzen dieses Meta-Beziehungstyps teilnehmen müssen. Entsprechend müssen bei einem Wert >=1 im Meta-Meta-Attribut „Min*Dest*Card“ Instanzen des *Quell*-Meta-Entitätstyps an Instanzen des Meta-Beziehungstyps enthalten sein.

722 In dieser Tabelle werden referenzierte Meta-Objekte nicht berücksichtigt.

723 Dies entspricht fast der Hälfte sämtlicher zwingend vorgeschriebenen Meta-Attribute des Integrierten EIA/CDIF-Meta-Modells (vgl. auch Abschnitt 4.2, ’Das „Integrierte EIA/CDIF-Meta-Modell“’ auf Seite 267ff unten).

Wie im GegenstandsBereich „Common“ wird auch hier von der Möglichkeit Gebrauch gemacht, Meta-Beziehungstypen so zu definieren, daß Instanzen von damit in Beziehung gesetzten Meta-Entitätstypen in jedem Fall in Instanzen des Meta-Beziehungstyps enthalten sein müssen. Es handelt sich um insgesamt sieben Meta-Beziehungstypen, die im folgenden kurz charakterisiert werden:

- „**1**:1 Annotation.*Uses*.**AnnotationArgument** 0:N“: Instanzen vom Typ *AnnotationArgument* müssen exakt einmal in einer Beziehung dieses Typs enthalten sein.
- „**1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N“: Instanzen vom Typ *Edge-Element* müssen exakt einmal an einer Beziehung dieses Typs partizipieren.
- „0:N **EdgeElement**.*HasEnd*.Point **2**:2“: Instanzen vom Typ *EdgeElement* müssen exakt zweimal in einer Beziehung dieses Typs enthalten sein und geben damit die beiden Endpunkte einer Kante an.
- „0:N **GraphicalElement**.*AppearsOn*.Diagram **1**:1“: Instanzen vom Typ *GraphicalElement* müssen exakt einmal in einer Beziehung dieses Typs enthalten sein.
- „0:N **Point**.*IsLocatedOn*.Diagram **1**:1“: Instanzen vom Typ *Point* müssen exakt einmal an einer Beziehung dieses Typs teilhaben.
- „0:N **PositionedElement**.*HasCenter*.Point **1**:1 *“: Instanzen vom Typ *PositionedElement* müssen exakt einmal in einer Beziehung dieses Typs enthalten sein und geben den Mittelpunkt für *Knoten* und *Beschreibungen* an.
- „0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1“: Instanzen vom Typ *RelativePoint* müssen exakt einmal in einer Beziehung dieses Typs enthalten sein.

Von den insgesamt 19 Blättern in diesem Meta-Modellbaum sind zehn Meta-Beziehungstypen und neun Meta-Entitätstypen.

4.1.3.3.3.2 Aufgefundene Fehler

Die im Rahmen dieser Arbeit erfolgte Analyse der Definitionen in bezug auf die korrekte Nutzung der EIA/CDIF-Datentypen für die Meta-Meta-Attribute sowie

mit dem Rahmen, den das Meta-Meta-Modell vorgibt, ergaben für das EIA/CDIF-Meta-Modell für „PLAC“ *einen* Fehler und zehn Warnungen:

– Fehler: Der Wert „PresentationLocationAndConnectivity“ für das Meta-Meta-Attribut „Name“ des Meta-Meta-Objekts „SubjectArea“ stellt keinen gültigen EIA/CDIF Identifier-Wert[724] dar, da er mit 35 Zeichen um drei Buchstaben zu lang ist.

– Warnung: Das Meta-Meta-Attribut „Description“ ist für den Meta-Entitätstyp „Edge“ *zweimal* mit unterschiedlichem Text vorhanden.

– Warnung: Für die Definition des Meta-Entitätstyps „RelativePoint“ wird für das Meta-Attribut „Transform11“ ein ungültiger Bezeichner verwendet. Der Wert für das Meta-Meta-Attribut „Name“ wäre zwar vom EIA/CDIF-Datentyp „Identifier“[725], entspricht aber nicht gemäß [CDIF94e][726] einem gültigen „MetaObjectName“, der in [CDIF94d][727] festgelegt ist.
  Dieselben Warnungen gelten für die weiteren Meta-Attribute dieses Meta-Entitätstyps:

  - „Transform12“,
  - „Transform13“,
  - „Transform21“,
  - „Transform22“,
  - „Transform23“,
  - „Transform31“,
  - „Transform32“ und
  - „Transform33“.

---

724 Vgl. [CDIF94b], Abschnitt 3.5.7, „Identifier“ auf Seite 23 Mitte, wo unter anderem die maximale Anzahl an Zeichen für Werte vom EIA/CDIF-Datentyp „Identifier“ mit 32 angegeben ist.

725 Die Einstufung als Warnung erfolgt, da es sich bei den Namen aller betroffenen Meta-Attribute um gültige EIA/CDIF-Identifier handelt.

726 Vgl. [CDIF94e], Seite 21, Regel für die Bildung von „MetaObjectName“ („<UpperCase-AlphabeticChar> [<UpperOrLowerCaseAlphabeticChar>]...“).

727 Vgl. [CDIF94d], Seite 31, beispielsweise die Regeln für „<Meta[Meta]AttributeName>“, „<Meta[Meta]EntityName>“ oder für „<Meta[Meta]RelationshipName>“, die dafür einen Wert vom Typ „MetaObjectName“ vorsehen.

4.1.3.3.3.3 Generalisierungshierarchie des Meta-Modells

Die folgende Abbildung 4-10 stellt die Generalisierungshierarchie dieses Meta-Modells dar, bei dem aufgrund der Nutzung der Meta-Objekte aus dem fundamentalen GegenstandsBereich „Foundation“ jeweils ein Teilbaum für die Meta-Entitätstypen und einer für die Meta-Beziehungstypen gebildet wird. Die maximale Höhe des Generalisierungsbaumes ist sechs, wobei der Zweig für die Meta-Beziehungstypen eine Höhe von drei Ebenen ausmacht. Somit stellt sich der Teilbaum für Meta-Entitätstypen als spezialisierter dar als der für Meta-Beziehungstypen.

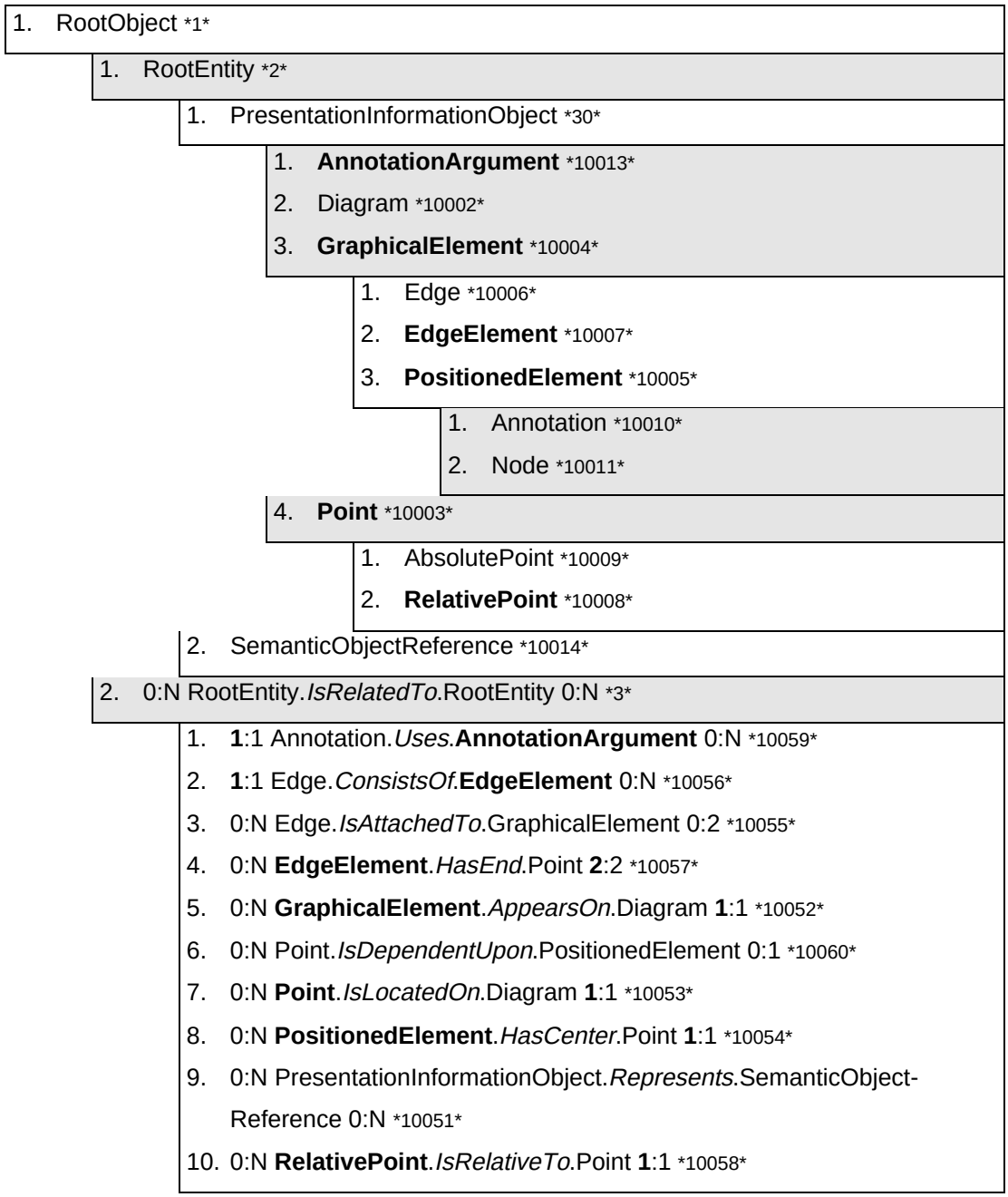


Abbildung 4-10: Generalisierungshierarchie des Meta-Modells „PLAC“[728]

4.1.3.3.3.4 Summarische Darstellung der AttribuierbarenMetaObjekte

Die weitere summarische Darstellung der Meta-Objekte für den Gegenstands-Bereich „Presentation Location and Connectivity“, kann aus dem Abschnitt 4.2, ‘Das „Integrierte EIA/CDIF-Meta-Modell“’ auf Seite 267ff, entnommen werden, in dem sämtliche Meta-Objekte der standardisierten EIA/CDIF-Meta-Modelle enthalten sind.

Tabelle 4-70 führt aus Gründen der Übersichtlichkeit jene Meta-Objekte voll-qualifiziert an, für die Meta-Attribute definiert sind. Wie in dieser Arbeit üblich, werden die Meta-Objekte mit den festgelegten Werten für das Meta-Meta-Attribut „CDIFMetaIdentifier“ in der EIA/CDIF „ENCODING.1“-Notation[729] für den EIA/CDIF-Datentyp „Identifier“ mit angeführt. Desgleichen werden gemeinsam mit den Meta-Beziehungstypen die entsprechenden Kardinalitäten dargestellt.

Vollqualifizierter Name	**Anzahl MA[730]**
AbsolutePoint *10009*	1/**1**
Annotation *10010*	2/0
1:1 Annotation.*Uses*.**AnnotationArgument** 0:N *10059*	1/0
AnnotationArgument *10013*	3/0
Diagram *10002*	5/0
1:1 Edge.*ConsistsOf*.**EdgeElement** 0:N *10056*	1/0
PositionedElement *10005*	3/0
RelativePoint *10008*	12/**2**
SemanticObjectReference *10014*	2/**1**

Tabelle 4-70: Meta-Objekte mit Meta-Attributen für den Gegenstandsbereich „PLAC“

728 Entsprechend den EIA/CDIF-Konventionen weisen kursiv gesetzte Zeilen auf Typen mit Mehrfachvererbung hin.

729 Vgl. in diesem Zusammenhang die Definitionen dafür in [CDIF94e].

730 Die erste Zahl gibt die Summe der Meta-Attribute des entsprechenden Meta-Objekts an, die zweite die Anzahl an zwingend vorgeschriebenen Attributen.

4.2 Das „Integrierte EIA/CDIF-Meta-Modell“

Das „Integrierte EIA/CDIF-Meta-Modell“ (abgekürzt: „IMM“) entsteht konzeptionell durch die Mengenvereinigung sämtlicher standardisierter Definitionen.[731] Die in dieser Arbeit für die erstmalige physische Erstellung des Integrierten EIA/CDIF-Meta-Modells berücksichtigten EIA/CDIF-Meta-Modelle umfassen sämtliche per 1. Jänner 1998 verabschiedeten EIA/CDIF-Standards. Es sind dies:

- „Integrated Meta-model, Foundation“, festgelegt in [CDIF94f],
- „Integrated Meta-model, Common“, festgelegt in [CDIF95],
- „Integrated Meta-model, Data Flow“, festgelegt in [CDIF96c],
- „Integrated Meta-model, Data Modeling“, festgelegt in [CDIF96b] und
- „Integrated Meta-model, Presentation Location and Connectivity“, festgelegt in [CDIF96d].

Die Anzahl der SammelbarenMetaObjekte des Integrierten Meta-Modells ist geringer als die Summe der SammelbarenMetaObjekte der Meta-Modelle für die einzelnen GegenstandsBereiche, nachdem Konzepte über alle Meta-Modelle hinweg wiederverwendet werden. Ein Beispiel dafür ist das fundamentale Meta-Modell, daß zwingend benutzt werden muß, sodaß die drei AttribuierbarenMetaObjekte „RootObject“, „RootEntity“ und „0:N RootEntity.*IsRelatedTo*.-RootEntity 0:N“ sich in allen Meta-Modellen wiederfinden[732], so auch im Integrierten Meta-Modell.

Tabelle 4-71 gibt eine Übersicht über sämtliche Meta-Objekte, die im Integrierten Meta-Modell enthalten sind.[733] Daraus folgt beispielsweise, daß es in etwa

---

731 Vgl. hierzu die Definition in den Glossaren zu den EIA/CDIF-Standards, beispielsweise in [CDIF94b], Eintrag „CDIF Integrated Meta-model“ auf Seite 88: „The description of the set of concepts and notations used to define a model. The CDIF Integrated Meta-model defines an Entity-Relationship-Attribute model that is used to construct and define models used in systems development“.

732 Dies gilt auch für Meta-Modelle, die diese fundamentalen Meta-Objekte nicht ausdrücklich mit Instanzen vom Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.-*IsUsedIn*.SubjectArea 1:N“ anführen, da in jedem Fall über die Generalisierungshierarchie diese drei Meta-Objekte eingebunden werden.

733 Nach Kenntnisstand des Autors gibt es derzeit kein Meta-Modell, das einem zuvor vereinbarten Meta-Meta-Modell entsprechend gebildet wird, das so umfassend viele Kon-

Fortsetzung folgt auf der nächsten Seite.

gleich viele Meta-Entitätstypen (62) und Meta-Beziehungstypen (60) gibt. Geht man davon aus, daß das Integrierte EIA/CDIF-Meta-Modell in seiner Struktur typisch für sämtliche EIA/CDIF-Meta-Modelle wäre, so könnten in bezug darauf Strukturabweichungen als Merkmale von individuellen Meta-Modellen und damit von ihren GegenstandsBereichen angesehen werden.

MO	**297**			
SA		5		
CMO		**292**		
MA			169	
AMO			**123**	[734] 1
MR				62
ME				60
Summe	**297**	**297**	**292**	**123**

Tabelle 4-71: Übersicht über die Meta-Objekte des Integrierten EIA/CDIF-Meta-Modells

Von den 62 Meta-Beziehungstypen sind 27, fast die Hälfte, zwingend vorgeschriebene Meta-Beziehungstypen, sodaß die Instanzen von insgesamt 21 Meta-Entitätstypen, fast ein Drittel aller Meta-Entitätstypen, auch in Instanzen von den entsprechenden Meta-Beziehungstypen enthalten sein müssen.

4.2.1 Kurzcharakterisierung des Integrierten EIA/CDIF-Meta-Modells

Das Integrierte EIA/CDIF-Meta-Modell beinhaltet sämtliche standardisierten Konzepte. Die Schwerpunktbildung und Interpretation von Teilen davon erfolgt mit Hilfe der Sichtweisen, für die die einzelnen EIA/CDF-Standards erzeugt wurden. Vom Meta-Meta-Modell ausgehend, wird in EIA/CDIF dieser Sichtme-

zepte beinhaltet wie das Integrierte EIA/CDIF-Meta-Modell. Hinzu kommt, daß weitere EIA/CDIF-Meta-Modelle in Vorbereitung sind, die zum Teil zahlreiche neue Konzepte in das Integrierte EIA/CDIF-Meta-Modell einbringen werden (vgl. z.B. BPM in [CDIF96e], OOAD in [CDIF96f], PMPS in [CDIF96g], DDEF in [CDIF96h], STEV in [CDIF96i] und CACSD in [CDIF98]).

[734] Es ist dies die einzige direkte Instanz des Meta-Meta-Entitätstyps „AttributableMeta-Object“, nämlich „RootObject“.

chanismus mit Instanzen vom Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“.

Wenn ein neues EIA/CDIF-Meta-Modell erstellt wird, soll nach den Vorstellungen des EIA/CDIF-Komitees soviele Konzepte aus dem Integrierten Meta-Modell wie möglich wiederbenutzt werden.[735]

4.2.2 Referenzierte Meta-Objekte

In diesem GegenstandsBereich werden *alle* Konzepte von EIA/CDIF-Meta-Modellen benutzt, sodaß es sich dabei – neben dem fundamentalen Meta-Modell – um das einzige Meta-Modell handelt, das keine MetaObjekte referenziert.

4.2.3 Auswertungen der Definitionen

Zunächst wird in diesem Abschnitt eine strukturelle Übersicht über das Integrierte EIA/CDIF-Meta-Modell gegeben, indem einfache Auswertungen tabellarisch aufbereitet werden. Daran anschließend werden sämtliche AttribuierbarenMetaObjekte, die ausdrücklich im Meta-Modell benutzt werden,[736] in der dem Meta-Modell entsprechenden Generalisierungshierarchie dargestellt, wobei MetaObjekte auf derselben Stufe alphabetisch sortiert sind. Abschließend werden *sämtliche* AttribuierbarenMetaObjekte in alphabetischer Reihenfolge summarisch angeführt, unabhängig davon, ob für sie ausdrücklich[737] MetaAttribute definiert sind. Somit stellt dieser Teil die vollständige Darstellung des Integrierten EIA/CDIF-Meta-Modell dar.

---

735 Dies stellt eine wesentliche Motivation für die Verfassung dieser Arbeit dar, da es bisher nach Kenntnis des Autors dieses Integrierte EIA/CDIF-Meta-Modell nur hypothetisch existierte. Es wird in diesem Abschnitt zum ersten Mal vollständig dokumentiert. Allerdings sind in dieser Arbeit, wie weiter oben bereits erwähnt, nur für die Meta-Modelle „Foundation“ und „Common“ die einzelnen AttribuierbareMetaObjekte im Detail beschrieben worden. Für eine detaillierte Beschreibung der Konzepte, die in anderen standardisierten EIA/CDIF-Meta-Modellen erfolgt, wird der Leser auf die entsprechenden, veröffentlichten Standards verwiesen.

736 Es handelt sich also um Instanzen jener SammelbarenMetaObjekte, die in Instanzen des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ für die Instanz des entsprechenden GegenstandBereiches enthalten sind.

737 Derartige Meta-Attribute werden in den EIA/CDIF-Standards auch als „lokal“ bezeichnet, um sie damit von den über die Generalisierungshierarchie ererbten Meta-Attribute unterscheidbar zu machen.

Sämtliche Auflistungen in diesem Abschnitt erfolgen anhand der voll qualifizierten Namen der AttribuierbarenMetaObjekte, wobei die Namen von Meta-Beziehungstypen immer kursiv gesetzt sind. Die Namen von Meta-Entitätstypen, deren Instanzen zwingend in Beziehungen auftreten müssen, werden immer fett hervorgehoben. Die Kardinalitäten der Meta-Beziehungstypen werden ausdrücklich angeführt, sowie die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“[738]. Sofern für AttribuierbareMetaObjekte die Mehrfachvererbung aufgrund der Definition vorgesehen ist, wird der gesamte vollqualifizierte Namen kursiv gesetzt. Sämtliche lokal definierten Meta-Attribute werden angeführt, wobei zusätzlich ihr EIA/CDIF-Datentyp in kleiner Schrift mitangegeben ist.[739]

4.2.3.1 Strukturelle Übersicht

Im Integrierten EIA/CDIF-Meta-Modell sind insgesamt 292 SammelbareMetaObjekte definiert. Tabelle 4-72 stellt die Aufstellung dieser SammelbarenMetaObjekte tabellarisch dar.

Anzahl Instanzen vom Typ AMO[740]	1
Anzahl Instanzen vom Typ ME	62
Anzahl Instanzen vom Typ MR	60
Anzahl Instanzen vom Typ MA	169
Anzahl (Summe) Instanzen vom Typ CMO	292

Tabelle 4-72: Verteilung der sammelbaren Objekte des Integrierten EIA/CDIF-Meta-Modells

Die folgende Tabelle 4-73 führt weitere Analyseergebnisse über die strukturellen Eigenschaften dieses Meta-Modells an.

738 Die Darstellung erfolgt in der vorgesehenen Verkodierung des EIA/CDIF-Datentyps „Identifier“, indem der Wert in Sterne eingefaßt wird (vgl. [CDIF94e], Seiten 11 und 20 unten, „IdentifierValue“).

739 Im Falle des aufzählbaren Datentyps werden auch die gültigen Werte der entsprechenden Wertemenge dargestellt.

740 Hier wird die einzige direkte Instanz von AMO angeführt, nämlich „RootObject“.

Anzahl zwingend vorgeschriebener MA		9
Anzahl optionaler MA		160
Meta-Objekte mit/ohne Meta-Attribute	mit MA	ohne MA
Anzahl Instanzen vom Typ AMO	1	0
Anzahl Instanzen vom Typ ME	41	21
Anzahl Instanzen vom Typ MR	10	50
Anzahl von zwingend vorgeschriebenen MR		27
Anzahl von ME, die an zwingend vorgeschriebenen MR teilhaben		21
Anzahl der Blätter (Instanzen vom Typ AMO) im Meta-Modellbaum		102
Anzahl von AMOs mit Mehrfachvererbung		4

Tabelle 4-73: Strukturelle Übersicht über das Integrierte EIA/CDIF-Meta-Modell

Etwa zwei Drittel aller Meta-Entitätstypen und ein Sechstel aller Meta-Beziehungstypen weisen Meta-Attribute auf. Insgesamt sind lediglich neun Meta-Attribute zwingend vorgeschrieben, die im folgenden mit ihren voll qualifizierten Namen aufgelistet werden:

- AbsolutePoint.**Position**
- AbstractionLevel.**Name**
- Port.**IsFormal**
- **ReferencedElement**.*DefinesPath*.ComponentObject.**SequenceNumber**
- **RelativePoint**.**DeltaX**
- **RelativePoint**.**DeltaY**
- RootObject.**CDIFIdentifier**
- SemanticObjectReference.**ReferencedMetaObjectInstance**
- ToolUser.**SystemName**

Damit müssen Instanzen der sieben Meta-Entitätstypen „AbsolutePoint“, „AbstractionLevel“, „Port“, „RelativePoint“, „RootObject“, „SemanticObject-Reference“ und „ToolUser“ sowie des Meta-Beziehungstyps „0:N **Referenced-**

Element.*DefinesPath*.ComponentObject 1:N“ in einem EIA/CDIF-Austausch Werte in den zwingend vorgeschriebenen Meta-Attributen aufweisen.

Wie im GegenstandsBereich „Common“ wird auch hier von der Möglichkeit Gebrauch gemacht, Meta-Beziehungstypen so zu definieren, daß Instanzen von damit in Beziehung gesetzten Meta-Entitätstypen in jedem Fall in Instanzen des Meta-Beziehungstyps enthalten sein müssen. Es handelt sich um insgesamt 27 Meta-Beziehungstypen, die bereits in den entsprechenden Abschnitten über die standardisierten EIA/CDIF-Meta-Modellen weiter oben kurz charakterisiert wurden und in der folgenden Tabelle 4-74[741] noch einmal angeführt sind.

1:1 Annotation.*Uses*.**AnnotationArgument** 0:N
1:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N
0:N **DataModelSubset**.*IsSubsetOf*.DataModel **1**:1
1:1 DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N
1:1 Edge.*ConsistsOf*.**EdgeElement** 0:N
0:N **EdgeElement**.*HasEnd*.Point **2**:2
1:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N
1:1 Entity.*IsIdentifiedBy*.**Key** 0:N
0:N **EquivalenceSet**.*HasMember*.ComponentObject **2**:N
0:N **ForeignKey**.*References*.CandidateKey **1**:1
0:N **GraphicalElement**.*AppearsOn*.Diagram **1**:1
1:N InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N
1:1 InheritableDataModelObject.*IsSupertypeFor*.**SubtypeSet** 0:N
0:N **Point**.*IsLocatedOn*.Diagram **1**:1
0:N **PositionedElement**.*HasCenter*.Point **1**:1
0:N **ProjectionComponent**.*IsFullProjectionOf*.DefinitionObject **1**:1
0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N
0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N

741 In der zusammenfassenden Darstellung in Abschnitt 4.3.1.1.3, „Zwingend vorgeschriebene Meta-Beziehungstypen“ auf Seite 344ff, werden die zwingend vorgeschriebenen Meta-Beziehungstypen nach GegenstandsBereichen getrennt sortiert dargestellt (vgl. Tabelle 4-79 auf Seite 345).

0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1
2:N **Role**.*BelongsTo*.**Relationship 1**:1
1:N RolePlayer.*Plays*.**Role** 0:1
1:1 RootEntity.*Has*.**AlternateName** 0:N
1:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N
1:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N
1:1 SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion** 0:N
0:N **SubtypeSetMembershipCriterion**.*Selects*.InheritableDataModelObject **1**:1
0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject 1:N

Tabelle 4-74: Liste der zwingend vorgeschriebenen Meta-Beziehungstypen

Von den insgesamt 102 Blättern in diesem Meta-Modellbaum sind 53 Meta-Beziehungstypen und 49 Meta-Entitätstypen, sodaß sie bemerkenswerterweise fast in einem Verhältnis von 1:1 zueinander existieren.

4.2.3.2 Aufgefundene Fehler

Für die Erstellung und die weitere Analyse des Integrierten EIA/CDIF-Meta-Modells wurden die Surrogatfehler aus dem Meta-Modell für den Gegenstands-Bereich „Datenmodellierung“[742] ausgeschaltet. Dies erfolgte einfach dadurch, daß jeweils einem der beiden doppelten Surrogatwerte ein „b“ angefügt wurde.[743]

Somit gelten für das Integrierte EIA/CDIF-Meta-Modell folgende sechs Festlegungen, von denen jene drei von den EIA/CDIF-Meta-Modell-Standards abweichen, die das kleine „b“ angefügt erhalten:

– Der Surrogatwert „1140“ wird dem Meta-Attribut „SpecificationText“ von dem Meta-Entitätstyp „DefinitionObject“ zugeordnet, der Wert „1140b“ dem Meta-Attribut „SpecificationText“ von dem Meta-Entitätstyp „Key“.

742 Vgl. Abschnitt 4.1.3.2.3.2, „Aufgefundene Fehler“ auf Seite 251ff weiter oben.

743 Somit kommt es bei einer Vorhaltung der EIA/CDIF-Definitionen für das Integrierte EIA/CDIF-Meta-Modell in einer relationalen Datenbank nicht mehr zu möglichen Verknüpfungsfehlern über die Spalte, die den Wert des Meta-Meta-Attributs „CDIFMetaIdentifier“ aufnimmt und für natürliche Verknüpfungen herangezogen wird.

– Der Surrogatwert „1733“ wird dem Meta-Beziehungstyp „0:N Key.-*Incorporates*.Attribute 0:N“ zugeordnet, der Wert „1733b“ dem Meta-Beziehungstyp „0:N Key.Incorporates.SemanticInformationObject 0:N“.

– Der Surrogatwert „1736“ wird dem Meta-Attribut „StoreWithSupertype“ von dem Meta-Beziehungstyp „1:N InheritableDataModelObject.*IsSubtypeIn*.**-SubtypeSet** 0:N“ zugeordnet, der Wert „1736b“ dem Meta-Beziehungstyp „1:1 SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion** 0:N“.

Sämtliche weiteren aufgefundenen Fehler und Warnungen sind für die Darstellung und Auswertung des Integrierten EIA/CDIF-Meta-Modells nicht relevant.

4.2.3.3 Generalisierungshierarchie des Integrierten EIA/CDIF-Meta-Modells

Die folgende **Abbildung 4-10** stellt die Generalisierungshierarchie dieses Meta-Modells dar, bei dem aufgrund der Nutzung der Meta-Objekte aus dem fundamentalen GegenstandsBereich „Foundation“ jeweils ein Teilbaum für die Meta-Entitätstypen und einer für die Meta-Beziehungstypen gebildet wird. Die maximale Höhe des Generalisierungsbaumes ist acht, wobei der Zweig für die Meta-Beziehungstypen eine Höhe von vier Ebenen ausmacht. Somit stellt sich der Teilbaum für Meta-Entitätstypen als spezialisierter dar als der für Meta-Beziehungstypen.

1. RootObject *1*
    1. RootEntity *2*
        1. AbstractionLevel *12*
        2. **AlternateName** *14*
        3. 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*

4. SemanticInformationObject *4*

1. **AccessPath** *1001*
2. ComponentObject *8000*

1. Attribute *17*

1. ProjectedAttribute *37*

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*

3. DataFlowModel *210*
4. DataModel *1008*
5. DataModelObject *1012*

1. *Cluster* *1005*
2. InheritableDataModelObject *1033*

1. *Entity* *1016*
2. ***Relationship*** *1040*

6. **DataModelSubset** *1013*
7. DataObject *22*
8. DefinitionObject *8002*

1. *Cluster* *1005*
2. DFMProcessDefinition *213*

3. *Entity* *1016*
4. ExternalAgentDefinition *6002*
5. FlowDefinition *221*
6. ***Relationship*** *1040*
7. **Role** *1042*
8. **RolePlayer** *1048*
9. StoreDefinition *239*

9. **Derivation** *26*
10. **Key** *1035*

1. CandidateKey *1003*
2. **ForeignKey** *1032*

11. ProcessObject *31*
12. **ProjectionComponent** *1036*
13. RoleConstraint *1045*
14. **SubtypeSet** *1070*
15. **SubtypeSetMembershipCriterion** *1074*

5. SemanticObjectReference *10014*
6. **TextualConstraint** *51*
7. ToolUser *56*

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. **1**:1 Annotation.*Uses*.**AnnotationArgument** 0:N *10059*
4. 0:N Attribute.*IsDiscriminatorFor*.SubtypeSetMembershipCriterion 0:N *1701*
5. 0:N Attribute.*IsInheritedFrom*.Attribute 0:1 *1702*
6. 0:N Cluster.*Collects*.DataModelObject 0:N *1703*
7. 0:N ComponentObject.*References*.DefinitionObject 0:1 *8022*
8. 0:1 DataFlowModel.*HasRoot*.DFMProcessDefinition 0:1 *6026*
9. 0:N DataModel.*Collects*.DataModelObject 0:N *1704*
10. **1**:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N *1705*
11. 0:N DataModelObject.*IsMemberOf*.DataModelSubset 0:N *1706*
12. 0:N DataModelSubset.*Excludes*.Attribute 0:N *1707*
13. 0:N **DataModelSubset**.*IsSubsetOf*.DataModel **1**:1 *1708*
14. 0:1 DefinitionObject.*Contains*.ComponentObject 0:N *1131*
15. **1**:1 DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N *1709*
16. **1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N *10056*
17. 0:N Edge.*IsAttachedTo*.GraphicalElement 0:2 *10055*
18. 0:N **EdgeElement**.*HasEnd*.Point **2**:2 *10057*

19. **1**:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N *1712*
20. **1**:1 Entity.*IsIdentifiedBy*.**Key** 0:N *1711*
21. 0:N **EquivalenceSet**.*HasMember*.ComponentObject **2**:N *1114*
22. 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*
23. 0:N **ForeignKey**.*References*.CandidateKey **1**:1 *1714*
24. 0:N **GraphicalElement**.*AppearsOn*.Diagram **1**:1 *10052*
25. **1**:N InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N *1715*
26. **1**:1 InheritableDataModelObject.*IsSupertypeFor*.**SubtypeSet** 0:N *1719*
27. 0:N Key.*Incorporates*.SemanticInformationObject 0:N *1733b*
    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*
28. 0:N Point.*IsDependentUpon*.PositionedElement 0:1 *10060*
29. 0:N **Point**.*IsLocatedOn*.Diagram **1**:1 *10053*
30. 0:N **PositionedElement**.*HasCenter*.Point **1**:1 *10054*
31. 0:N PresentationInformationObject.*Represents*.SemanticObjectReference 0:N *10051*
32. 0:N ProjectedAttribute.*IsProjectionOf*.Attribute 0:N *63*
33. 0:N **ProjectionComponent**.*IsFullProjectionOf*.DefinitionObject **1**:1 *1721*
34. 0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N *1722*
35. 0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N *8025*
36. 0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1 *10058*
37. **2**:N **Role**.*BelongsTo*.**Relationship 1**:1 *1724*
38. 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*
39. 0:N RolePlayer.*IsSupportedBy*.Key 0:1 *1729*
40. **1**:N RolePlayer.*Plays*.**Role** 0:1 *1730*
41. 0:N RolePlayer.*Refines*.RolePlayer 0:1 *1731*
42. 0:1 RolePlayer.*RefinesForSubtype*.DataModelObject 0:N *1732*
43. 0:N RootEntity.*CreatedBy*.ToolUser 0:1 *68*
44. **1**:1 RootEntity.*Has*.**AlternateName** 0:N *69*
45. 0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1 *70*
46. 0:N RootEntity.*Uses*.AlternateName 0:1 *71*
47. 0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N *72*
48. **1**:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N *73*

49. **1**:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N *74*
50. **1**:1 SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion** 0:N *1736b*
51. 0:N **SubtypeSetMembershipCriterion**.*Selects*.InheritableDataModel-Object **1**:1 *1737*
52. 0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject 1:N *80*

Abbildung 4-11: Generalisierungshierarchie des Integrierten EIA/CDIF-Meta-Modells[744]

4.2.3.4 Summarische Darstellung der AttribuierbarenMetaObjekte

Tabelle 4-75 führt aus Gründen der Übersichtlichkeit jene Meta-Objekte vollqualifiziert an, für die Meta-Attribute definiert sind. Wie in dieser Arbeit üblich werden die Meta-Objekte mit den festgelegten Werten für das Meta-Meta-Attribut „CDIFMetaIdentifier“ in der EIA/CDIF „ENCODING.1“-Notation[745] für den EIA/CDIF-Datentyp „Identifier“ mit angeführt. Desgleichen werden gemeinsam mit den Meta-Beziehungstypen die entsprechenden Kardinalitäten dargestellt.

Vollqualifizierter Name	**Anzahl MA[746]**
AbsolutePoint *10009*	1/**1**
AbstractionLevel *12*	1/**1**
AccessPath *1001*	3/0
0:N AccessPath.*Incorporates*.Attribute 0:N *1716*	2/0
AlternateName *14*	2/0
Annotation *10010*	2/0
1:1 Annotation.*Uses*.**AnnotationArgument** 0:N *10059*	1/0
AnnotationArgument *10013*	3/0
Attribute *17*	3/0

744 Entsprechend den EIA/CDIF-Konventionen weisen kursiv gesetzte Zeilen auf Typen mit Mehrfachvererbung hin.

745 Vgl. in diesem Zusammenhang die Definitionen dafür in [CDIF94e].

746 Die erste Zahl gibt die Summe der Meta-Attribute des entsprechenden Meta-Objekts an, die zweite Zahl die Anzahl zwingend vorgeschriebener Attribute.

Vollqualifizierter Name	Anzahl MA[746]
CandidateKey *1003*	1/0
DataFlowModel *210*	2/0
DataModel *1008*	2/0
DataModelSubset *1013*	1/0
DataObject *22*	1/0
DefinitionObject *8002*	4/0
0:1 DefinitionObject.*Contains*.ComponentObject 0:N *1131*	1/0
1:1 DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N *1709*	1/0
Derivation *26*	3/0
DFMProcess *6005*	3/0
DFMProcessDefinition *213*	6/0
Diagram *10002*	5/0
1:1 Edge.*ConsistsOf*.**EdgeElement** 0:N *10056*	1/0
Entity *1016*	14/0
ExternalAgent *6004*	3/0
Flow *227*	3/0
FlowDefinition *221*	4/0
InheritableDataModelObject *1033*	1/0
1:N InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N *1715*	3/0
Key *1035*	3/0
0:N Key.*Incorporates*.Attribute 0:N *1733*	1/0
0:N Key.*Incorporates*.SemanticInformationObject 0:N *1733_Duplicate_2*[747]	1/0
Port *1129*	2/**1**
PositionedElement *10005*	3/0
ProcessObject *31*	5/0
ProjectedAttribute *37*	2/0

747 Der Wert für das Meta-Meta-Attribut „CDIFMetaIdentifier“ wurde bereits für ein anderes Meta-Objekt vergeben, sodaß hier ein künstlicher Wert erzeugt wurde, der garantiert über das ganze Integrierte EIA/CDIF-Meta-Modell hinweg eindeutig ist. „`*1733_Duplicate_2*`“ wird in dieser Arbeit manchmal auch durch den Wert „`*1733b*`“ ersetzt.

Vollqualifizierter Name	Anzahl MA[746]
ProjectionComponent *1036*	3/0
0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N *1722*	1/0
0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N *8025*	1/**1**
Relationship *1040*	1/0
RelativePoint *10008*	12/**2**
Role *1042*	2/0
RoleConstraint *1045*	2/0
RolePlayer *1048*	21/0
RootObject *1*	5/**1**
SemanticInformationObject *4*	2/0
SemanticObjectReference *10014*	2/**1**
Store *6028*	3/0
StoreDefinition *239*	8/0
SubtypeSet *1070*	3/0
SubtypeSetMembershipCriterion *1074*	3/0
TextualConstraint *51*	4/0
ToolUser *56*	2/**1**

Tabelle 4-75: Meta-Objekte mit Meta-Attributen für das Integegrierte EIA/CDIF-Meta-Modell

In den weiteren Unterabschnitten erfolgt eine tabellarische Aufstellung *sämtlicher* AttribuierbarenMetaObjekte mit der Darstellung der ererbten und lokalen Meta-Attribute, wobei für letztere prägnant der entsprechende Datentyp dokumentiert wird. Die vollständige Darstellung macht das „Integrierte EIA/CDIF-Meta-Modell“ aus und kann auch als Referenz benutzt werden, die die wesentlichsten Definitionen beinhaltet. Vor einer jeden Tabelle wird in einem kurzen Satz angeführt, in welchen standardisierten EIA/CDIF-GegenstandsBereichen

beziehungsweise der entsprechenden EIA/CDIF-Meta-Modellen das nachfolgende MetaObjekt ausdrücklich[748] benutzt wird.

4.2.3.4.1.1 Summarische Definition der direkten Instanz vom Typ AMO

Tabelle IMM 4-1 stellt die einzige Instanz des Meta-Meta-Entitätstyp „AttributableMetaObject“ dar, die zugleich die Wurzel sämtlicher EIA/CDIF-konformen Meta-Modelle darstellt. „RootObject“ wird ausdrücklich im Gegenstandsbereich FND benutzt und ansonsten indirekt über die Generalisierungshierarchie ererbt. Es werden dafür insgesamt fünf lokale Meta-Attribute definiert, wovon eines zwingend vorgeschrieben ist.

RootObject *1*	
CDIFIdentifier *5* – Identifier	**zwingend** [lokal]
DateCreated *6* – Date	optional
DateUpdated *7* – Date	optional
TimeCreated *8* – Time	optional
TimeUpdated *9* – Time	optional

IMM 4-1: Summarische Darstellung von „RootObject“

4.2.3.4.1.2 Summarische Definition der direkten Instanzen vom Typ „MetaEntity“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AbsolutePoint“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es wird ein einziges lokales MetaAttribut definiert, das zugleich zwingend vorgeschrieben ist.

AbsolutePoint *10009*	
CDIFIdentifier *5*	***zwingend*** *von: RootObject* *1*
DateCreated *6*	*optional*
DateUpdated *7*	*optional*

748 „Ausdrücklich“ bedeutet in diesem Zusammenhang, daß mit Hilfe von Instanzen vom Typ „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ die MetaObjekte einem GegenstandsBereich zugeordnet sind.

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
Position *10101* – Point	**zwingend**	[lokal]

IMM 4-2: Meta-Entitätstyp „AbsolutePoint“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AbstractionLevel“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es wird ein einziges lokales MetaAttribut definiert, das zugleich zwingend vorgeschrieben ist.

AbstractionLevel *12*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
Name *13* – Enumerated {Conceptual, Logical, Physical}	**zwingend**	[lokal]

IMM 4-3: Meta-Entitätstyp „AbstractionLevel“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AccessPath“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

AccessPath *1001*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1081* – String(256)	optional	[lokal]

SpecificationLanguage *1082* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	
SpecificationText *1083* – Text	optional	

IMM 4-4: Meta-Entitätstyp „**AccessPath**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AlternateName“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

AlternateName *14*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
OtherLongName *15* – String(1024)	optional	[lokal]
OtherName *16* – String(256)	optional	

IMM 4-5: Meta-Entitätstyp „**AlternateName**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Annotation“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

Annotation *10010*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ExtentX *10121*	*optional*	*von: PositionedElement* *10005*
ExtentY *10122*	*optional*	
ExtentZ *10123*	*optional*	

DerivationLanguage *10102* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	[lokal]
DerivationText *10103* – Text	optional	

IMM 4-6: Meta-Entitätstyp „Annotation“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AnnotationArgument“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

AnnotationArgument *10013*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
DataBlock *10104* – Text	optional	[lokal]
MIMESubtype *10119* – String(32)	optional	
MIMEType *10118* – String(32)	optional	

IMM 4-7: Meta-Entitätstyp „**AnnotationArgument**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Attribute“ dar und wird ausdrücklich in den Gegenstandsbereichen DMOD und DFM benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

Attribute *17*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
DefaultValue *18* – String(1024)	optional	[lokal]

IsOptional *19* – Boolean	optional	
Name *21* – String(256)	optional	

IMM 4-8: Meta-Entitätstyp „Attribute“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „CandidateKey“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

CandidateKey *1003*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1077*	*optional*	*von: Key* *1035*
SpecificationLanguage *1078*	*optional*	
SpecificationText *1140b*	*optional*	
IsPrimary *1004* – Boolean	optional	[lokal]

IMM 4-9: Meta-Entitätstyp „CandidateKey“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Cluster“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

Cluster *1005*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*

FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	

IMM 4-10: Meta-Entitätstyp „Cluster“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ComponentObject“ dar und wird ausdrücklich in den Gegenstandsbereichen DMOD und DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

ComponentObject *8000*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	

IMM 4-11: Meta-Entitätstyp „ComponentObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ConstraintPort“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

ConstraintPort *6012*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*

FullDescription *45*	*optional*	
IsFormal *1139*	***zwingend***	*von: Port* *1129*
Name *1152*	*optional*	

IMM 4-12: Meta-Entitätstyp „ConstraintPort“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ControlPort“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

ControlPort *6014*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsFormal *1139*	***zwingend***	*von: Port* *1129*
Name *1152*	*optional*	

IMM 4-13: Meta-Entitätstyp „ControlPort“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DataFlowModel“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

DataFlowModel *210*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	

Name *211* – String(256)	optional	[lokal]
Type *212* – Enumerated {Current, Required}	optional	

IMM 4-14: Meta-Entitätstyp „DataFlowModel“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DataModel“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

DataModel *1008*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
ModelType *1010* – String(64)	optional	[lokal]
Name *1011* – String(256)	optional	

IMM 4-15: Meta-Entitätstyp „DataModel“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DataModelObject“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

DataModelObject *1012*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	

IMM 4-16: Meta-Entitätstyp „DataModelObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DataModelSubset“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

DataModelSubset *1013*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1015* – String(256)	optional	[lokal]

IMM 4-17: Meta-Entitätstyp „**DataModelSubset**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DataObject“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es wird ein einziges lokales MetaAttribut definiert.

DataObject *22*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *23* – String(256)	optional	[lokal]

IMM 4-18: Meta-Entitätstyp „DataObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DefinitionObject“ dar und wird ausdrücklich in den Gegenstandsbereichen DMOD und DFM benutzt. Es werden dafür insgesamt vier lokale Meta-Attribute definiert.

DefinitionObject *8002*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119* – String(256)	optional	[lokal]
Operator *1118* – Enumerated {AND, XOR, OR}	optional	
SpecificationLanguage *1141* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	
SpecificationText *1140* – Text	optional	

IMM 4-19: Meta-Entitätstyp „DefinitionObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Derivation“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

Derivation *26*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	

DerivationLanguage *29* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	[lokal]
DerivationText *27* – Text	optional	
IsRealizationOf *185* – Boolean	optional	

IMM 4-20: Meta-Entitätstyp „**Derivation**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DFMProcess“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

DFM|Process *6005*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
ContextDescription *1122* – Text	optional	[lokal]
ContextIdentifier *1123* – String(32)	optional	
Name *1150* – String(256)	optional	

IMM 4-21: Meta-Entitätstyp „DFMProcess“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DFMProcessDefinition“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt sechs lokale Meta-Attribute definiert.

DFM|ProcessDefinition *213*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	
FunctionalArea *214* – String(256)	optional	[lokal]
HasConcurrentChildren *215* – Boolean	optional	
HasRealTimeSemantics *1130* – Boolean	optional	
IsControl *216* – Boolean	optional	
IsData *217* – Boolean	optional	
IsMaterial *218* – Boolean	optional	

IMM 4-22: Meta-Entitätstyp „DFMProcessDefinition“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Diagram“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es werden dafür insgesamt fünf lokale Meta-Attribute definiert.

Diagram *10002*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
DotsPerUnit *10108* – Float	optional	[lokal]
ExtentX *10105* – Float	optional	
ExtentY *10106* – Float	optional	
ExtentZ *10107* – Float	optional	
Unit *10109* – Enumerated {PerMeter, PerCentimeter, PerMillimeter, PerInch, PerMicrometer}	optional	

IMM 4-23: Meta-Entitätstyp „Diagram“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Edge“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

Edge *10006*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-24: Meta-Entitätstyp „Edge“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „EdgeElement“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

EdgeElement *10007*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-25: Meta-Entitätstyp „**EdgeElement**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Entity“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt 14 lokale Meta-Attribute definiert.

Entity *1016*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	
IsAbstract *1034*	*optional*	*von: InheritableDataModelObject* *1033*
AvgNumberOfOccurrences *1017* – Float	optional	[lokal]
DeletionTimePeriod *1018* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
EntityType *1019* – Enumerated {Kernel, Characteristic, Associative}	optional	
InsertionTimePeriod *1020* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
MaxNumberOfOccurrences *1022* – Integer	optional	
MinNumberOfOccurrences *1023* – Integer	optional	
NormalizationState *1024* – Enumerated {UNF, 1NF, 2NF, 3NF, BCNF, 4NF, 5NF.}	optional	
NumberOfDeletions *1025* – Float	optional	
NumberOfInsertions *1026* – Float	optional	
NumberOfReads *1027* – Float	optional	
NumberOfUpdates *1028* – Float	optional	
ReadTimePeriod *1029* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
UpdateTimePeriod *1030* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
Usage *1031* – Text	optional	

IMM 4-26: Meta-Entitätstyp „Entity“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „EquivalenceSet“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

EquivalenceSet *1113*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	

IMM 4-27: Meta-Entitätstyp „**EquivalenceSet**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ExternalAgent“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

ExternalAgent *6004*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
ContextDescription *1124* – Text	optional	[lokal]
ContextIdentifier *1125* – String(32)	optional	
Name *1151* – String(256)	optional	

IMM 4-28: Meta-Entitätstyp „ExternalAgent“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ExternalAgentDefinition“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

ExternalAgentDefinition *6002*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	

IMM 4-29: Meta-Entitätstyp „ExternalAgentDefinition“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Flow“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

Flow *227*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
ContextDescription *228* – Text	optional	[lokal]

ContextIdentifier *6041* – String(32)	optional	
Name *231* – String(256)	optional	

IMM 4-30: Meta-Entitätstyp „Flow“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „FlowDefinition“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt vier lokale Meta-Attribute definiert.

FlowDefinition *221*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	
IsAnalog *1142* – Boolean	optional	[lokal]
IsControl *223* – Boolean	optional	
IsData *224* – Boolean	optional	
IsMaterial *225* – Boolean	optional	

IMM 4-31: Meta-Entitätstyp „FlowDefinition“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „FlowInputPort“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

FlowInputPort *6001*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	

DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsFormal *1139*	***zwingend***	*von: Port* *1129*
Name *1152*	*optional*	

IMM 4-32: Meta-Entitätstyp „FlowInputPort“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „FlowOutputPort“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

FlowOutputPort *6000*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsFormal *1139*	***zwingend***	*von: Port* *1129*
Name *1152*	*optional*	

IMM 4-33: Meta-Entitätstyp „FlowOutputPort“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „FlowPort“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

FlowPort *6015*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsFormal *1139*	***zwingend***	*von: Port* *1129*
Name *1152*	*optional*	

IMM 4-34: Meta-Entitätstyp „FlowPort“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „FlowProducerConsumer“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

FlowProducerConsumer *232*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	

IMM 4-35: Meta-Entitätstyp „FlowProducerConsumer“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ForeignKey“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

ForeignKey *1032*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*

FullDescription *45*	*optional*	
Name *1077*	*optional*	*von: Key* *1035*
SpecificationLanguage *1078*	*optional*	
SpecificationText *1140b*	*optional*	

IMM 4-36: Meta-Entitätstyp „**ForeignKey**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „GraphicalElement“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

GraphicalElement *10004*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-37: Meta-Entitätstyp „**GraphicalElement**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „InheritableDataModelObject“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

InheritableDataModelObject *1033*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsAbstract *1034* – Boolean	optional	[lokal]

IMM 4-38: Meta-Entitätstyp „InheritableDataModelObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Key“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

Key *1035*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1077* – String(256)	optional	[lokal]
SpecificationLanguage *1078* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	
SpecificationText *1140b* – Text	optional	

IMM 4-39: Meta-Entitätstyp „**Key**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Node“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

Node *10011*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ExtentX *10121*	*optional*	*von: PositionedElement* *10005*
ExtentY *10122*	*optional*	
ExtentZ *10123*	*optional*	

IMM 4-40: Meta-Entitätstyp „Node“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Point“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

Point *10003*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-41: Meta-Entitätstyp „**Point**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Port“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert, wovon eines zwingend vorgeschrieben ist.

Port *1129*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsFormal *1139* – Boolean	**zwingend**	[lokal]
Name *1152* – String(256)	optional	

IMM 4-42: Meta-Entitätstyp „Port“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „PositionedElement“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

PositionedElement *10005*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ExtentX *10121* – Float	optional	[lokal]
ExtentY *10122* – Float	optional	
ExtentZ *10123* – Float	optional	

IMM 4-43: Meta-Entitätstyp „**PositionedElement**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „PresentationInformationObject“ dar und wird ausdrücklich in den Gegenstandsbereichen CMMN und PLAC benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

PresentationInformationObject *30*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-44: Meta-Entitätstyp „PresentationInformationObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ProcessObject“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt fünf lokale Meta-Attribute definiert.

ProcessObject *31*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	

TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
ExecutionTimeInterval *34* – Float	optional	[lokal]
ExecutionTimeUnit *32* – Enumerated {Picosecond, Nanosecond, Microsecond, Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
Name *33* – String(256)	optional	
SpecificationLanguage *36* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	
SpecificationText *35* – Text	optional	

IMM 4-45: Meta-Entitätstyp „ProcessObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ProjectedAttribute“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

ProjectedAttribute *37*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
DefaultValue *18*	*optional*	*von: Attribute* *17*
IsOptional *19*	*optional*	
Name *21*	*optional*	

SpecificationLanguage *39* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	[lokal]
SpecificationText *38* – Text	optional	

IMM 4-46: Meta-Entitätstyp „ProjectedAttribute“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ProjectionComponent“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

ProjectionComponent *1036*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1037* – String(256)	optional	[lokal]
SpecificationLanguage *1038* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	
SpecificationText *1039* – Text	optional	

IMM 4-47: Meta-Entitätstyp „**ProjectionComponent**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ReferencedElement“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

ReferencedElement *8020*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	

TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	

IMM 4-48: Meta-Entitätstyp „**ReferencedElement**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Relationship“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

Relationship *1040*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	
IsAbstract *1034*	*optional*	*von: InheritableDataModelObject* *1033*
InverseName *1041* – String(256)	optional	[lokal]

IMM 4-49: Meta-Entitätstyp „**Relationship**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „RelativePoint“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es werden dafür insgesamt 12 lokale Meta-Attribute definiert, wovon zwei zwingend vorgeschrieben sind.

RelativePoint *10008*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*

DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
DeltaX *10124* – Float	**zwingend**	[lokal]
DeltaY *10125* – Float	**zwingend**	
DeltaZ *10126* – Float	optional	
Transform11 *10131* – Float	optional	
Transform12 *10132* – Float	optional	
Transform13 *10133* – Float	optional	
Transform21 *10134* – Float	optional	
Transform22 *10135* – Float	optional	
Transform23 *10136* – Float	optional	
Transform31 *10137* – Float	optional	
Transform32 *10138* – Float	optional	
Transform33 *10139* – Float	optional	

IMM 4-50: Meta-Entitätstyp „**RelativePoint**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Role“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

Role *1042*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*

Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	
IsMaster *1043* – Boolean	optional	[lokal]
IsSource *1044* – Boolean	optional	

IMM 4-51: Meta-Entitätstyp „**Role**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „RoleConstraint“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

RoleConstraint *1045*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1046* – String(256)	optional	[lokal]
Operator *1047* – Enumerated {AND, OR, XOR}	optional	

IMM 4-52: Meta-Entitätstyp „RoleConstraint“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „RolePlayer“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt 21 lokale Meta-Attribute definiert.

RolePlayer *1048*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	

TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	
AvgNumberOfOccurrences *1049* – Float	optional	[lokal]
DeleteEffect *1050* – Enumerated {RESTRICTS, CASCADES, SETNULL, SETDEFAULT}	optional	
DeletionTimePeriod *1051* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
InsertEffect *1052* – Enumerated {RESTRICTS, CASCADES, SETNULL, SETDEFAULT}	optional	
InsertionTimePeriod *1053* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
IsDeleteDeferrable *1054* – Boolean	optional	
IsInsertDeferrable *1055* – Boolean	optional	
IsUpdateDeferrable *1056* – Boolean	optional	
MaxInnerCardinality *1057* – String(10)	optional	
MaxNumberOfOccurrences *1058* – Integer	optional	
MaxOuterCardinality *1059* – String(10)	optional	
MinInnerCardinality *1060* – String(10)	optional	
MinNumberOfOccurrences *1061* – Integer	optional	
MinOuterCardinality *1062* – String(10)	optional	
NumberOfDeletions *1063* – Float	optional	
NumberOfInsertions *1064* – Float	optional	
NumberOfReads *1065* – Float	optional	

NumberOfUpdates *1066* – Float	optional	
ReadTimePeriod *1067* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year.}	optional	
UpdateEffect *1068* – Enumerated {RESTRICTS, CASCADES, SETNULL, SETDEFAULT}	optional	
UpdateTimePeriod *1069* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	optional	

IMM 4-53: Meta-Entitätstyp „**RolePlayer**“

Tabelle IMM 4-54 stellt die erste Instanz des Meta-Meta-Entitätstyp „MetaEntity“ dar und bildet zugleich den Wurzeltyp für sämtliche Meta-Entitätstypen von EA/CDIF-entsprechenden Meta-Modellen. „RootEntity“ wird ausdrücklich in den Gegenstandsbereichen FND und CMMN benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

RootEntity *2*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject *1**
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-54: Meta-Entitätstyp „RootEntity“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SemanticInformationObject“ dar und wird ausdrücklich in den Gegenstandsbereichen CMMN und DMOD benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

SemanticInformationObject *4*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	

TimeUpdated *9*	*optional*	
BriefDescription *44* – String(1024)	optional	[lokal]
FullDescription *45* – Text	optional	

IMM 4-55: Meta-Entitätstyp „SemanticInformationObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SemanticObjectReference“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert, wovon eines zwingend vorgeschrieben ist.

SemanticObjectReference *10014*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ReferencedMetaAttributeName *10111* – Identifier	optional	[lokal]
ReferencedMetaObjectInstance *10110* – Identifier	**zwingend**	

IMM 4-56: Meta-Entitätstyp „SemanticObjectReference“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „Store“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

Store *6028*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*

FullDescription *45*	*optional*	
ContextDescription *1120* – Text	optional	[lokal]
ContextIdentifier *1121* – String(32)	optional	
Name *1153* – String(256)	optional	

IMM 4-57: Meta-Entitätstyp „Store“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „StoreDefinition“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es werden dafür insgesamt acht lokale Meta-Attribute definiert.

StoreDefinition *239*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
Name *1119*	*optional*	*von: DefinitionObject* *8002*
Operator *1118*	*optional*	
SpecificationLanguage *1141*	*optional*	
SpecificationText *1140*	*optional*	
IsControl *240* – Boolean	optional	[lokal]
IsData *241* – Boolean	optional	
IsManual *242* – Boolean	optional	
IsMaterial *243* – Boolean	optional	
IsPermanent *244* – Boolean	optional	
IsReadOnly *245* – Boolean	optional	
Size *247* – Float	optional	
SizeUnits *248* – Enumerated {Bit, Byte, Kilobyte, Megabyte, Gigabyte, Terabyte}	optional	

IMM 4-58: Meta-Entitätstyp „StoreDefinition“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SubtypeSet“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

SubtypeSet *1070*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsExclusive *1071* – Boolean	optional	[lokal]
Name *1072* – String(256)	optional	
SubtypeListIsClosed *1073* – Boolean	optional	

IMM 4-59: Meta-Entitätstyp „**SubtypeSet**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SubtypeSetMembershipCriterion“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

SubtypeSetMembershipCriterion *1074*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
DiscriminatorValue *1080* – String(256)	optional	[lokal]

SpecificationLanguage *1075* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	
SpecificationText *1076* – Text	optional	

IMM 4-60: Meta-Entitätstyp „**SubtypeSetMembershipCriterion**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SupportPort“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Entitätstyp weist keine lokalen Meta-Attribute auf.

SupportPort *6010*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *44*	*optional*	*von: SemanticInformationObject* *4*
FullDescription *45*	*optional*	
IsFormal *1139*	***zwingend***	*von: Port* *1129*
Name *1152*	*optional*	

IMM 4-61: Meta-Entitätstyp „SupportPort“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „TextualConstraint“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt vier lokale Meta-Attribute definiert.

TextualConstraint *51*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
BriefDescription *52* – String(1024)	optional	[lokal]
ConstraintExpression *53* – Text	optional	

ConstraintLanguage *55* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	
FullDescription *54* – Text	optional	

IMM 4-62: Meta-Entitätstyp „**TextualConstraint**“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ToolUser“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert, wovon eines zwingend vorgeschrieben ist.

ToolUser *56*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
FullName *57* – String(256)	optional	[lokal]
SystemName *58* – String(32)	**zwingend**	

IMM 4-63: Meta-Entitätstyp „ToolUser“

4.2.3.4.1.3 Summarische Definition der direkten Instanzen vom Typ „MetaRelationship“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt zwei lokale Meta-Attribute definiert.

0:N AccessPath.*Incorporates.*Attribute 0:N *1716*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
IsAscending *1738* – Boolean	optional	[lokal]
SequenceNumber *1739* – Integer	optional	

IMM 4-64: Meta-Beziehungstyp „0:N AccessPath.*Incorporates.*Attribute 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Instantiates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:1 AccessPath.*Instantiates.*Key 0:1 *1700*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-65: Meta-Beziehungstyp „0:1 AccessPath.*Instantiates.*Key 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Uses“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es wird ein einziges lokales MetaAttribut definiert.

1:1 Annotation.*Uses*.**AnnotationArgument** 0:N *10059*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *10117* – Integer	optional	[lokal]

IMM 4-66: Meta-Beziehungstyp „**1**:1 Annotation.Uses.AnnotationArgument 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsDiscriminatorFor“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N Attribute.*IsDiscriminatorFor*.SubtypeSetMembershipCriterion 0:N *1701*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-67: Meta-Beziehungstyp „0:N Attribute.*IsDiscriminatorFor*.SubtypeSet-MembershipCriterion 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsInheritedFrom“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N Attribute.*IsInheritedFrom*.Attribute 0:1 *1702*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-68: Meta-Beziehungstyp „0:N Attribute.*IsInheritedFrom*.Attribute 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N CandidateKey.*Incorporates*.ForeignKey 0:N *1713*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *1735*	*optional*	*von: Key.Incorporates.-SemanticInformationObject* *1733b*

IMM 4-69: Meta-Beziehungstyp „0:N CandidateKey.*Incorporates*.ForeignKey 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Collects“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N Cluster.*Collects*.DataModelObject 0:N *1703*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-70: Meta-Beziehungstyp „0:N Cluster.*Collects*.DataModelObject 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „References“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N ComponentObject.*References*.DefinitionObject 0:1 *8022*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-71: Meta-Beziehungstyp „0:N ComponentObject.*References*.DefinitionObject 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „HasRoot“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:1 DataFlowModel.*HasRoot*.DFMProcessDefinition 0:1 *6026*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-72: Meta-Beziehungstyp „0:1 DataFlowModel.*HasRoot*.DFMProcessDefinition 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Collects“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N DataModel.*Collects*.DataModelObject 0:N *1704*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-73: Meta-Beziehungstyp „0:N DataModel.*Collects*.DataModelObject 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „ActsAs“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N *1705*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-74: Meta-Beziehungstyp „**1**:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsMemberOf“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N DataModelObject.*IsMemberOf*.DataModelSubset 0:N *1706*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-75: Meta-Beziehungstyp „0:N DataModelObject.*IsMemberOf*.DataModelSubset 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Excludes“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N DataModelSubset.*Excludes*.Attribute 0:N *1707*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-76: Meta-Beziehungstyp „0:N DataModelSubset.*Excludes*.Attribute 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsSubsetOf“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **DataModelSubset**.*IsSubsetOf*.DataModel **1**:1 *1708*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-77: Meta-Beziehungstyp „0:N **DataModelSubset**.*IsSubsetOf*.DataModel **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Contains“ dar und wird ausdrücklich in den Gegenstandsbereichen DMOD und DFM benutzt. Es wird ein einziges lokales MetaAttribut definiert.

0:1 DefinitionObject.*Contains*.ComponentObject 0:N *1131*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *8026* – Integer	optional	[lokal]

IMM 4-78: Meta-Beziehungstyp „0:1 DefinitionObject.*Contains*.ComponentObject 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsConstructedWith“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

1:1 DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N *1709*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *1710* – Integer	optional	[lokal]

IMM 4-79: Meta-Beziehungstyp „**1**:1 DefinitionObject.*IsConstructedWith*.**Projection-Component** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „ConsistsOf“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Es wird ein einziges lokales MetaAttribut definiert.

1:1 Edge.*ConsistsOf*.**EdgeElement** 0:N *10056*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *10114* – Integer	optional	[lokal]

IMM 4-80: Meta-Beziehungstyp „**1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsAttachedTo“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N Edge.*IsAttachedTo*.GraphicalElement 0:2 *10055*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-81: Meta-Beziehungstyp „0:N Edge.*IsAttachedTo*.GraphicalElement 0:2“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „HasEnd“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **EdgeElement**.*HasEnd*.Point **2**:2 *10057*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-82: Meta-Beziehungstyp „0:N **EdgeElement**.*HasEnd*.Point **2**:2“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsAccessedUsing“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N *1712*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-83: Meta-Beziehungstyp „**1**:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsIdentifiedBy“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:1 Entity.*IsIdentifiedBy*.**Key** 0:N *1711*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-84: Meta-Beziehungstyp „**1**:1 Entity.*IsIdentifiedBy*.**Key** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „HasMember“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **EquivalenceSet**.*HasMember*.ComponentObject **2**:N *1114*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-85: Meta-Beziehungstyp „0:N **EquivalenceSet**.*HasMember*.ComponentObject **2**:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Consumes“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N FlowProducerConsumer.*Consumes*.Flow 0:N *6024*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-86: Meta-Beziehungstyp „0:N FlowProducerConsumer.*Consumes*.Flow 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Produces“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N FlowProducerConsumer.*Produces*.Flow 0:N *6025*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-87: Meta-Beziehungstyp „0:N FlowProducerConsumer.*Produces*.Flow 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „ProducesOrConsumes“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N FlowProducerConsumer.*ProducesOrConsumes*.Flow 0:N *6023*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-88: Meta-Beziehungstyp „0:N FlowProducerConsumer.*ProducesOrConsumes*.-Flow 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:1 ForeignKey.*Incorporates*.RolePlayer 0:1 *1728*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *1735*	*optional*	*von: Key.Incorporates.Semantic-InformationObject* *1733b*

IMM 4-89: Meta-Beziehungstyp „0:1 ForeignKey.*Incorporates*.RolePlayer 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „References“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **ForeignKey**.*References*.CandidateKey **1**:1 *1714*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-90: Meta-Beziehungstyp „0:N **ForeignKey**.*References*.CandidateKey **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „AppearsOn“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **GraphicalElement**.*AppearsOn*.Diagram **1**:1 *10052*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-91: Meta-Beziehungstyp „0:N **GraphicalElement**.*AppearsOn*.Diagram **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsSubtypeIn“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es werden dafür insgesamt drei lokale Meta-Attribute definiert.

1:N InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N *1715*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SpecificationLanguage *1717* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, PASCAL, PL1, SQL, Other}	optional	[lokal]
SpecificationText *1718* – Text	optional	
StoreWithSupertype *1736* – Boolean	optional	

IMM 4-92: Meta-Beziehungstyp „**1**:N InheritableDataModelObject.*IsSubtypeIn*.-**SubtypeSet** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsSupertypeFor“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:1 InheritableDataModelObject.*IsSupertypeFor*.**SubtypeSet** 0:N *1719*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-93: Meta-Beziehungstyp „**1**:1 InheritableDataModelObject.*IsSupertypeFor*.-**SubtypeSet** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

0:N Key.*Incorporates*.Attribute 0:N *1733*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	

DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *1735*	*optional*	*von: Key.Incorporates.Semantic-InformationObject* *1733b*
IsAscending *1734* – Boolean	optional	[lokal]

IMM 4-94: Meta-Beziehungstyp „0:N Key.*Incorporates*.Attribute“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

0:N Key.*Incorporates*.SemanticInformationObject 0:N *1733b*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *1735* – Integer	optional	[lokal]

IMM 4-95: Meta-Beziehungstyp „0:N Key.*Incorporates*.SemanticInformationObject 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsDependentUpon“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N Point.*IsDependentUpon*.PositionedElement 0:1 *10060*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-96: Meta-Beziehungstyp „0:N Point.*IsDependentUpon*.PositionedElement 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsLocatedOn“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **Point**.*IsLocatedOn*.Diagram **1**:1 *10053*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-97: Meta-Beziehungstyp „0:N **Point**.*IsLocatedOn*.Diagram **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „HasCenter“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **PositionedElement**.*HasCenter*.Point **1**:1 *10054*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-98: Meta-Beziehungstyp „0:N **PositionedElement**.*HasCenter*.Point **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Represents“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N PresentationInformationObject.*Represents*.SemanticObjectReference 0:N *10051*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-99: Meta-Beziehungstyp „0:N PresentationInformationObject.*Represents*.-SemanticObjectReference 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsProjectionOf“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N ProjectedAttribute.*IsProjectionOf.*Attribute 0:N *63*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-100: Meta-Beziehungstyp „0:N ProjectedAttribute.*IsProjectionOf.*Attribute 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsFullProjectionOf“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **ProjectionComponent**.*IsFullProjectionOf.*DefinitionObject **1**:1 *1721*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-101: Meta-Beziehungstyp „0:N **ProjectionComponent**.*IsFullProjectionOf*.-DefinitionObject **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsProjectionOf“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Es wird ein einziges lokales MetaAttribut definiert.

0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N *1722*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *1723* – Integer	optional	[lokal]

IMM 4-102: Meta-Beziehungstyp „0:N **ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „DefinesPath“ dar und wird ausdrücklich im Gegenstandsbereich DFM benutzt. Es wird ein einziges lokales MetaAttribut definiert, das zugleich zwingend vorgeschrieben ist.

0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N *8025*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
SequenceNumber *6040* – Integer	**zwingend**	[lokal]

IMM 4-103: Meta-Beziehungstyp „0:N **ReferencedElement**.*DefinesPath*.Component-Object **1**:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsRelativeTo“ dar und wird ausdrücklich im Gegenstandsbereich PLAC benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1 *10058*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-104: Meta-Beziehungstyp „0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „BelongsTo“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

2:N **Role**.*BelongsTo*.**Relationship 1**:1 *1724*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-105: Meta-Beziehungstyp „**2**:N **Role**.*BelongsTo*.**Relationship 1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RoleConstraint.*Incorporates*.RoleConstraint 0:N *1725*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-106: Meta-Beziehungstyp „0:N RoleConstraint.*Incorporates*.RoleConstraint 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RoleConstraint.*Incorporates*.RolePlayer 0:N *1726*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-107: Meta-Beziehungstyp „0:N RoleConstraint.*Incorporates*.RolePlayer 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Incorporates“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RoleConstraint.*Incorporates*.SemanticInformationObject 0:N *1727*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-108: Meta-Beziehungstyp „0:N RoleConstraint.*Incorporates*.Semantic-InformationObject 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsSupportedBy“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RolePlayer.*IsSupportedBy*.Key 0:1 *1729*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-109: Meta-Beziehungstyp „0:N RolePlayer.*IsSupportedBy*.Key 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Plays“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:N RolePlayer.*Plays*.**Role** 0:1 *1730*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-110: Meta-Beziehungstyp „**1**:N RolePlayer.*Plays*.**Role** 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Refines“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RolePlayer.*Refines*.RolePlayer 0:1 *1731*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-111: Meta-Beziehungstyp „0:N RolePlayer.*Refines*.RolePlayer 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „RefinesForSubtype“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:1 RolePlayer.*RefinesForSubtype*.DataModelObject 0:N *1732*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-112: Meta-Beziehungstyp „0:1 RolePlayer.*RefinesForSubtype*.DataModelObject 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „CreatedBy“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RootEntity.*CreatedBy*.ToolUser 0:1 *68*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-113: Meta-Beziehungstyp „0:N RootEntity.*CreatedBy*.ToolUser 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Has“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:1 RootEntity.*Has*.**AlternateName** 0:N *69*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-114: Meta-Beziehungstyp „**1**:1 RootEntity.*Has*.**AlternateName** 0:N“

Tabelle IMM 4-115 stellt die erste Instanz des Meta-Meta-Entitätstyp „MetaRelationship“ dar und bildet zugleich den Wurzeltyp für sämtliche Meta-Beziehungstypen von EA/CDIF-entsprechenden Meta-Modellen. Der Meta-Beziehungstyp wird ausdrücklich im Gegenstandsbereich FND benutzt und weist keine lokalen Meta-Attribute auf.

0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *3*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject *1**
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-115: Meta-Beziehungstyp „0:N RootEntity.*IsRelatedTo*.RootEntity 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „LastUpdatedBy“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1 *70*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-116: Meta-Beziehungstyp „0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Uses“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N RootEntity.*Uses*.AlternateName 0:1 *71*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-117: Meta-Beziehungstyp „0:N RootEntity.*Uses*.AlternateName 0:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsCategorizedIn“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N *72*	
CDIFIdentifier *5*	***zwingend*** *von: RootObject* *1*
DateCreated *6*	*optional*
DateUpdated *7*	*optional*
TimeCreated *8*	*optional*
TimeUpdated *9*	*optional*

IMM 4-118: Meta-Beziehungstyp „0:N SemanticInformationObject.*IsCategorizedIn*.-AbstractionLevel 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „ProducedBy“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N *73*	
CDIFIdentifier *5*	***zwingend*** *von: RootObject* *1*
DateCreated *6*	*optional*
DateUpdated *7*	*optional*
TimeCreated *8*	*optional*
TimeUpdated *9*	*optional*

IMM 4-119: Meta-Beziehungstyp „**1**:N SemanticInformationObject.*ProducedBy*.-**Derivation** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „UsedIn“ dar und wird ausdrücklich im GegenstandsBereich CMMN benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N *74*	
CDIFIdentifier *5*	***zwingend*** *von: RootObject* *1*
DateCreated *6*	*optional*
DateUpdated *7*	*optional*

TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-120: Meta-Beziehungstyp „**1**:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Specifies“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

1:1 SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion** 0:N *1736b*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-121: Meta-Beziehungstyp „**1**:1 SubtypeSet.*Specifies*.**SubtypeSetMembership-Criterion** 0:N“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „Selects“ dar und wird ausdrücklich im Gegenstandsbereich DMOD benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **SubtypeSetMembershipCriterion**.*Selects*.InheritableDataModelObject **1**:1 *1737*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-122: Meta-Beziehungstyp „0:N **SubtypeSetMembershipCriterion**.*Selects*.-InheritableDataModelObject **1**:1“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Beziehungstyps „IsConstraintOn“ dar und wird ausdrücklich in den Gegenstandsbe-

reichen CMMN und benutzt. Der Meta-Beziehungstyp weist keine lokalen Meta-Attribute auf.

0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject **1**:N *80*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

IMM 4-123: Meta-Beziehungstyp „0:N **TextualConstraint**.*IsConstraintOn*.-SemanticInformationObject **1**:N“

4.3 Zusammenfassende Diskussion der standardisierten EIA/CDIF-Meta-Modelle

In diesem Abschnitt wird zunächst versucht, durch entsprechende zusammenfassende Darstellungen verschiedene Einsichten in die Strukturen der EIA/CDIF-Meta-Modelle zu geben. Daran anschließend wird eine prägnante, zusammenfassende Diskussion geführt, die auch auf jene strukturellen Einschränkungen[749] eingeht, die sinnvollerweise behoben werden sollten.

In den weiteren Unterabschnitten werden die standardisierten EIA/CDIF-Meta-Modelle unter jeweils einem Gesichtspunkt zusammengefaßt, diskutiert und entsprechende Schlußfolgerungen gezogen.

4.3.1 Verschiedene Auswertungen über SammelbareMetaObjekte

Die folgende Tabelle 4-76 stellt die Summe der SammelbarenMetaObjekte dar, die nach den entsprechenden EIA/CDIF-Meta-Modellen aufgeteilt sind. Die Prozentzahlen geben darüber Auskunft, wie hoch der Anteil der entsprechenden Meta-Modelldefinitionen an den SammelbarenMetaObjekten des Integrierten EIA/CDIF-Meta-Modells ist.

In den dargestellten standardisierten EIA/CDIF-Meta-Modellen werden nur jene AttribuierbarenMetaObjekte angeführt, die ausdrücklich im entsprechenden Meta-Modell benutzt werden.[750]

---

749 Es handelt sich demgemäß um Einschränkungen, die im Meta-Meta-Modell – also der M3-Schicht – selbst begründet sind.

750 Mit anderen Worten: die Zahlen beinhalten nicht die über die Generalisierungshierarchie mittels Vererbung referenzierten AttribuierbarenMetaObjekte.

Meta-Modell	AMO	AMO %	ME	ME %	MR	MR %	MA	MA%
IMM	**1**	**100,0%**	**60**	**100,0%**	**62**	**100,0%**	**169**	**100,0%**
FND	1	100,0%	1	1,7%	1	1,6%	5	3,0%
CMMN	-	-	10	16,7%	8	12,9%	20	11,8%
DMOD	-	-	22	36,7%	34	54,8%	81	47,9%
DFM	-	-	22	36,7%	8	12,9%	43	25,4%
PLAC	-	-	13	21,7%	10	16,1%	30	17,8%

Tabelle 4-76: Anteile der aufsummierten SammelbarenMetaObjekte der standardisierten EIA/CDIF-Meta-Modelle am Integrierten EIA/CDIF-Meta-Modell

Somit läßt sich beispielsweise aus Tabelle 4-76 ablesen, daß DMOD und DFM die gleiche Anzahl an Meta-Entitätstypen aus dem IMM benutzen, sich jedoch stark im Anteil an den Meta-Beziehungstypen unterscheiden. Daraus ließe sich beispielsweise der Schluß ziehen, daß DMOD aufgrund der höheren Anzahl an Meta-Beziehungstypen strukturell komplexer als DFM ist.[751]

Wenn die Anzahl der AttribuierbarenMetaObjekte zur Anzahl der Meta-Attribute ins Verhältnis gesetzt wird, so gilt für IMM ein Verhältnis von AMO zu MA von 1:1,37, für FND 1:1,67, für CMMN 1:1,11, für DMOD 1:1,45, für DFM 1:1,43 und für PLAC 1:1,30. Somit entfallen im Meta-Modell für den fundamentalen GegenstandsBereich „Foundation“ mit durchschnittlich 1,67 die meisten Meta-Attribute pro AttribuierbarenMetaObjekt, gefolgt vom Meta-Modell für den GegenstandsBereich „Datenmodellierung“ mit durchschnittlich 1,45 Meta-Attribute per AttribuierbarenMetaObjekt.

751 Hier wäre als Maßzahl das Verhältnis der Meta-Entitätstypen zu Meta-Beziehungstypen benutzt worden. Diese läge für das Integrierte EIA/CDIF-Meta-Modell bei etwa 1:1, für DFM bei 1:0,36 (etwa 3:1) und schließlich für DMOD bei 1:1.55 (etwa 2:3). CMMN (1:0.8, etwa 5:4) und PLAC (1:0.77, etwa 3:2) wären hier etwas weniger komplex als der „Durchschnitt“, wenn darunter das Integrierte EIA/CDIF-Meta-Modell verstanden würde.

4.3.1.1 AttribuierbareMetaObjekte

In diesem Abschnitt werden unterschiedliche AttribuierbareMetaObjekt-bezogene Eigenschaften der standardisierten EIA/CDIF-Meta-Modelle zusammenfassend dargestellt.

4.3.1.1.1 Wiederbenutzte AttribuierbareMetaObjekte

Sämtliche AttribuierbareMetaObjekte des GegenstandsBereiches „Foundation“ werden entweder direkt[752] in allen anderen Meta-Modellen benutzt oder werden infolge der Vererbung über die Generalisierungshierarchie eingesetzt.

Tabelle 4-77 stellt jene AttribuierbarenMetaObjekte dar, die in mehr als einem GegenstandsBereich ausdrücklich[753] benutzt werden, gemeinsam mit der Angabe der betroffenen GegenstandsBereiche selbst.[754]

Name des AttribuierbarenMetaObjekts	**benutzt in den Meta-Modellen**
RootEntity *2*	FND, CMMN
SemanticInformationObject *4*	CMMN, DMOD
Attribute *17*	DFM, DMOD
PresentationInformationObject *30*	CMMN, PLAC
DefinitionObject.*Contains*.ComponentObject *1131*	DFM, DMOD
ComponentObject *8000*	DFM, DMOD
DefinitionObject *8002*	DFM, DMOD

Tabelle 4-77: Liste der ausdrücklich mehrfach verwendeten AttribuierbarenMeta-Objekte

Die sieben aufgefundenen AttribuierbarenMetaObjekte stellen lediglich einen Anteil von 5,7 % von insgesamt 123 dar. Dieser geringe Anteil an Wiederverwendung ist daraus erklärbar, daß zunächst fundamental unterschiedliche Ge-

752 Dies ist dann der Fall, wenn die AttribuierbarenMetaObjekte über den Meta-Meta-Beziehungstyp „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ einem GegenstandsBereich ausdrücklich zugeordnet werden.

753 Es sind dies in allen aufgefundenen Fällen zwei GegenstandsBereiche.

754 Damit handelt es sich bei den dargestellten AttribuierbarenMetaObjekten um solche, die mindestens einmal wiederverwendet wurden.

genstandsBereiche in Form von standardisierten EIA/CDIF-Meta-Modellen erstellt werden, die miteinander relativ wenig gemein haben.[755]

Auf der anderen Seite kann ein hoher Grad an Wiederverwendung beispielsweise beim Vorschlag eines EIA/CDIF-Meta-Modells für die Geschäftsprozeßmodellierung von Boeing[756] aufgefunden werden. Darin finden sich sehr viele Konzepte aus dem EIA/CDIF-GegenstandsBereich „Datenflußmodellierung“ wieder.

4.3.1.1.2 AttribuierbareMetaObjekte mit Mehrfachvererbung

Von den insgesamt 123 AttribuierbarenMetaObjekten finden sich lediglich vier (Anteil von 3,3 %), in Tabelle 4-78 angeführte Meta-Entitätstypen, die das in EIA/CDIF zur Verfügung stehende Konzept der Mehrfachvererbung in Anspruch nehmen.

Meta-Modell	Name des AttribuierbarenMeta-Objekts (Subtyp)	Name des AttribuierbarenMeta-Objekts (Supertyp)
DMOD	Cluster *1005*	DataModelObject *1012* DefinitionObject *8002*
DMOD	Entity *1016*	DefinitionObject *8002* InheritableDataModelObject *1033*
DMOD	Relationship *1040*	DefinitionObject *8002* InheritableDataModelObject *1033*
DFM	FlowPort *6015*	FlowProducerConsumer *232* Port *1129*

Tabelle 4-78: AttribuierbareMetaObjekte mit Mehrfachvererbung

Auffallend ist hierbei, daß die drei Meta-Entitätstypen „Cluster“, „Entity“ und „Relationship“ des GegenstandBereiches „Datenmodellierung“ die Mehrfachvererbung dafür einsetzen, um auch die Eigenschaften des Meta-Entitätstyps

755 Dieses Faktum kann auch so interpretiert werden, daß die Mitglieder des EIA/CDIF-Komitees erfolgreich die GegenstandsBereiche so partitioniert haben, daß lediglich minimale Überschneidungen überhaupt möglich sind.

756 Vgl. in diesem Zusammenhang [CDIF96e] und die prägnante Darstellung der Auswertungen im Kapitel 5, „Zusammenfassung und Ausblick“ auf Seite 385ff weiter unten.

„DefinitionObject“ zu erlangen. Dadurch wird es für diese drei Meta-Entitätstypen möglich, den allgemeinen Strukturierungsmechanismus in Anspruch zu nehmen.

Im GegenstandsBereich „Datenflußmodellierung“ entsteht der Entitätstyp „FlowPort“ als Kreuzung zwischen „FlowProducerConsumer“ und „Port“, wodurch Meta-Entitätstypen erzeugt werden, die Öffnungen für ein- und ausgehende Flüsse darstellen.

4.3.1.1.3 Zwingend vorgeschriebene Meta-Beziehungstypen

Tabelle 4-79 stellt sämtliche zwingend vorgeschriebenen Meta-Beziehungstypen (in Summe 27) über alle standardisierten EIA/CDIF-Meta-Modelle hinweg dar, gemeinsam mit den entsprechenden GegenstandsBereichen.

	Name des zwingend vorgeschriebenen Meta-Beziehungstyps			**Meta-Modell**
1:1	RootEntity.*Has*.**AlternateName**	0:N	*69*	CMMN
1:N	SemanticInformationObject.*ProducedBy*.**Derivation**	0:N	*73*	CMMN
1:N	SemanticInformationObject.*UsedIn*.**Derivation**	0:N	*74*	CMMN
0:N	**TextualConstraint**.*IsConstraintOn*.SemanticInformation-Object	**1**:N	*80*	CMMN
1:1	DataModelObject.*ActsAs*.**RolePlayer**	0:N	*1705*	DMOD
0:N	**DataModelSubset**.*IsSubsetOf*.DataModel	**1**:1	*1708*	DMOD
1:1	DefinitionObject.*IsConstructedWith*.**ProjectionComponent**	0:N	*1709*	DMOD
1:1	Entity.*IsAccessedUsing*.**AccessPath**	0:N	*1712*	DMOD
1:1	Entity.*IsIdentifiedBy*.**Key**	0:N	*1711*	DMOD
0:N	**ForeignKey**.*References*.CandidateKey	**1**:1	*1714*	DMOD
1:N	InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet**	0:N	*1715*	DMOD
1:1	InheritableDataModelObject.*IsSupertypeFor*.**SubtypeSet**	0:N	*1719*	DMOD
0:N	**ProjectionComponent**.*IsFullProjectionOf*.DefinitionObject	**1**:1	*1721*	DMOD
0:N	**ProjectionComponent**.*IsProjectionOf*.Attribute	**1**:N	*1722*	DMOD
2:N	**Role**.*BelongsTo*.**Relationship**	**1**:1	*1724*	DMOD
1:N	RolePlayer.*Plays*.**Role**	0:1	*1730*	DMOD

	Name des zwingend vorgeschriebenen Meta-Beziehungstyps			Meta-Modell
1:1	SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion**	0:N	*1736b*[757]	DMOD
0:N	**SubtypeSetMembershipCriterion**.*Selects*.InheritableData-ModelObject	**1**:1	*1737*	DMOD
0:N	**EquivalenceSet**.*HasMember*.ComponentObject	**2**:N	*1114*	DFM
0:N	**ReferencedElement**.*DefinesPath*.ComponentObject	**1**:N	*8025*	DFM
1:1	Annotation.*Uses*.**AnnotationArgument**	0:N	*10059*	PLAC
1:1	Edge.*ConsistsOf*.**EdgeElement**	0:N	*10056*	PLAC
0:N	**EdgeElement**.*HasEnd*.Point	**2**:2	*10057*	PLAC
0:N	**GraphicalElement**.*AppearsOn*.Diagram	**1**:1	*10052*	PLAC
0:N	**Point**.*IsLocatedOn*.Diagram	**1**:1	*10053*	PLAC
0:N	**PositionedElement**.*HasCenter*.Point	**1**:1	*10054*	PLAC
0:N	**RelativePoint**.*IsRelativeTo*.Point	**1**:1	*10058*	PLAC

Tabelle 4-79: Zwingend vorgeschriebene Meta-Beziehungstypen, geordnet nach GegenstandsBereichen

Für die 27 zwingend vorgeschriebenen Meta-Beziehungstypen ergibt sich eine Verteilung auf die verschiedenen standardisierten EIA/CDIF-Meta-Modelle[758], wie sie in Tabelle 4-80 dargestellt ist.

757 Der angeführte Wert für das Meta-Meta-Attribut „CDIFMetaIdentifier“ entspricht dem für das Integrierte EIA/CDIF-Meta-Modell gewählten, nachdem der Originalwert aus einem Fehler heraus doppelt vergeben wurde (vgl. hierzu die Ausführungen in Abschnitt 4.1.3.2.3.2, Seite 251, sowie Abschnitt 4.2.3.2, Seite 273, weiter oben).

758 Es werden nur jene GegenstandsBereiche berücksichtigt, die zwingend vorgeschriebene Meta-Beziehungstypen aufweisen, sodaß als Folge davon in dieser Aufstellung das fundamentale Meta-Modell fehlt.

Meta-Modell	Meta-Beziehungstypen		
	alle	zwingend	Anteil
CMMN	8	4	50,0 %
DMOD	34	14	41,2 %
DFM	8	2	25,0 %
PLAC	10	7	70,0 %
Alle	**60**	**27**	**45,0 %**

Tabelle 4-80: Verteilung der zwingend vorgeschriebenen Meta-Beziehungstypen auf die verschiedenen, standardisierten EIA/CDIF-Meta-Modelle

4.3.1.1.4 Meta-Entitätstypen in zwingend vorgeschriebenen Meta-Beziehungstypen

Tabelle 4-81 stellt jene 21 Meta-Entitätstypen dar, die an zwingend vorgeschriebenen Meta-Beziehungstypen in bestimmten standardisierten EIA/CDIF-Meta-Modellen partizipieren müssen.

Name des Meta-Entitätstyps		Meta-Modell
AccessPath	*1001*	DMOD
AlternateName	*14*	CMMN
AnnotationArgument	*10013*	PLAC
DataModelSubset	*1013*	DMOD
Derivation	*26*	CMMN
EdgeElement	*10007*	PLAC
EquivalenceSet	*1113*	DFM
ForeignKey	*1032*	DMOD
GraphicalElement	*10004*	PLAC
Key	*1035*	DMOD
Point	*10003*	PLAC
PositionedElement	*10005*	PLAC
ProjectionComponent	*1036*	DMOD

Name des Meta-Entitätstyps		Meta-Modell
ReferencedElement	*8020*	DFM
Relationship	*1040*	DMOD
RelativePoint	*10008*	PLAC
Role	*1042*	DMOD
RolePlayer	*1048*	DMOD
SubtypeSet	*1070*	DMOD
SubtypeSetMembershipCriterion	*1074*	DMOD
TextualConstraint	*51*	CMMN

Tabelle 4-81: Meta-Entitätstypen, die zwingend an Meta-Beziehungstypen teilnehmen müssen, unter Angabe der entsprechenden GegenstandsBereiche

4.3.1.2 Auswertungen über Meta-Attribute

In diesem Abschnitt werden unterschiedliche Meta-Attribut-bezogene Eigenschaften der standardisierten EIA/CDIF-Meta-Modelle zusammenfassend dargestellt.

4.3.1.2.1 Verteilung der Meta-Attribute auf die AttribuierbarenMeta-Objekte

Tabelle 4-82 stellt das Ergebnis der Auszählung der AttribuierbarenMetaObjekte nach GegenstandsBereich getrennt dar, unter dem Gesichtspunkt, ob sie Meta-Attribute aufweisen oder nicht. Darüber hinaus wird gezeigt, wie hoch die entsprechende absolute Anzahl an zwingend vorgeschriebenen und optionalen Meta-Attributen ist.

Meta-Modell	Typ	∑ AMO	∑ AMO ohne MA	∑ AMO mit MA	Anteil in %	∑ zwingender MA	∑ optionaler MA	
FND	AMO	1	-	**1**	**100,0%**	1	4	∅ 4,0
	ME	1	1	-	-	-	-	
	MR	1	1	-	-	-	-	
CMMN	ME	10	2	**8**	**80,0%**	2	18	∅ 2,3
	MR	8	8	-	-	-	-	
DMOD	ME	22	4	**18**	**81,8%**	-	71	∅ 3,9
	MR	34	27	**7**	**20,6%**	-	10	∅ 1,4
DFM	ME	22	11	**11**	**50,0%**	1	40	∅ 3,5
	MR	8	6	**2**	**25,0%**	1	1	∅ 0,5
PLAC	ME	13	6	**7**	**53,8%**	4	24	∅ 3,4
	MR	10	8	**2**	**20,0%**	-	2	∅ 1,0

Tabelle 4-82: Auszählung der AttribuierbarenMetaObjekte nach GegenstandsBereich getrennt, mit und ohne Meta-Attribute und Zuordnung der absoluten Anzahl der zwingend vorgeschriebenen und optionalen Meta-Attribute

Es zeigt sich daher, daß im fundamentalen GegenstandsBereich im Unterschied zu allen anderen, auch der Meta-Entitätstyp über keine eigenen Meta-Attribute verfügt. Dies gilt für Meta-Beziehungstypen nur im GegenstandsBereich „Foundation“ und zusätzlich in „Common“.

Mindestens die Hälfte aller Meta-Entitätstypen in den entsprechenden GegenstandsBereichen verfügt über eigene Meta-Attribute, bei den Meta-Beziehungstypen ist dies bei nur einem Fünftel bis einem Viertel der Fall.

Die letzte Spalte in Tabelle 4-82 zeigt die durchschnittliche Anzahl an optionalen Meta-Attributen an, bezogen auf jene AttribuierbarenMetaObjekte, die überhaupt welche aufweisen. Tabelle 4-84 auf Seite 350 unten schließt die zwingend vorgeschriebenen Meta-Attribute in der entsprechenden Auswertung mit ein.

Die folgende Tabelle 4-83 zeigt die Anzahl der AttribuierbarenMetaObjekte geordnet nach den standardisierten EIA/CDIF-GegenstandsBereichen und der

Anzahl der definierten Meta-Attribute[759]. Beispielsweise existieren im GegenstandsBereich „Datenmodellierung“ vier Meta-Entitätstypen ohne jegliche Meta-Attribute und ein Meta-Entitätstyp[760] mit immerhin 21 (!) Meta-Attributen.

Meta-Modell	Typ	∑ AMO	Anzahl MA
FND	**AMO**	1	5
	ME	1	0
	MR	1	0
CMMN	**ME**	2	0
		2	1
		3	2
		1	3
		1	4
		1	5
	MR	8	0
DMOD	**ME**	4	0
		4	1
		5	2
		6	3
		1	4
		1	14
		1	21
	MR	27	0
		5	1
		1	2
		1	3
DFM	**ME**	11	0
		2	2
		5	3
		2	4
		1	6
		1	8
	MR	6	0
		2	1

759 Es wird in dieser Auswertung kein Unterschied zwischen zwingend vorgeschriebenen und optionalen Meta-Attributen gemacht.

760 Dabei handelt es sich um den Meta-Entitätstyp „RolePlayer“.

Meta-Modell	Typ	∑ AMO	Anzahl MA
PLAC	**ME**	6	0
		1	1
		2	2
		2	3
		1	5
		1	12
	MR	8	0
		2	1

Tabelle 4-83: Auszählung der Anzahl der AttribuierbarenMetaObjekte nach GegenstandsBereichen und Anzahl der Meta-Attribute geordnet

Tabelle 4-84 stellt die Daten aus Tabelle 4-83 in verdichteter Form dar, wobei hierfür nur AttribuierbareMetaObjekte berücksichtigt wurden, die über lokale Meta-Attribute[761] verfügen. Zudem wird bei dieser Auswertung auch die Standardabweichung berechnet. So ist es beispielsweise auffallend, daß die Standardabweichungen für die Anzahl der Meta-Attribute, die für Meta-Entitätstypen definiert sind, in den GegenstandsBereichen „Datenmodellierung“ und PLAC relativ hoch sind.

Meta-Modell	Typ	AMO mit MA	∑ MA	∅ MA	Standardabweichung	MIN MA	MAX MA
FND	AMO	1	5	5,00	0,00	5	5
CMMN	ME	8	20	2,50	1,41	1	5
DMOD	ME	18	71	3,94	5,15	1	21
	MR	7	10	1,43	0,79	1	3
DFM	ME	11	41	3,73	1,79	2	8
	MR	2	2	1,00	0,00	1	1
PLAC	ME	7	28	4,00	3,74	1	12
	MR	2	2	1,00	0,00	1	1

Tabelle 4-84: Einfache statistische Auswertung über die Verwendung von Meta-Attributen nach GegenstandsBereichen getrennt, bezogen auf AttribuierbareMeta-Objekte, für die Meta-Attribute definiert sind

761 Es wird in dieser Auswertung kein Unterschied zwischen zwingend vorgeschriebenen und optionalen Meta-Attributen gemacht.

Tabelle 4-85 listet jene acht AttribuierbarenMetaObjekte auf, in diesem Fall ausschließlich Meta-Entitätstypen, die überdurchschnittlich[762] viele lokale Meta-Attribute aufweisen.

Meta-Modell	Typ	Name des Attribuierbaren-MetaObjekts	Anzahl MA
FND	AMO	RootObject *1*	5
CMMN	ME	ProcessObject *31*	5
DMOD	ME	Entity *1016*	14
	ME	RolePlayer *1048*	21
DFM	ME	DFMProcessDefinition *213*	6
	ME	StoreDefinition *239*	8
PLAC	ME	Diagram *10002*	5
	ME	RelativePoint *10008*	12

Tabelle 4-85: AttribuierbareMetaObjekte mit überdurchschnittlich vielen (fünf oder mehr) Meta-Attributen, geordnet nach GegenstandsBereichen

4.3.1.2.2 Zwingend vorgeschriebene Meta-Attribute

Es fällt aufgrund der Auswertung sämtlicher, standardisierter EIA/CDIF-Meta-Modelle auf, daß von den insgesamt 169 Meta-Attributen nur neun Meta-Attribute zwingend vorgeschrieben sind. Anders ausgedrückt, *95 %* aller im Rahmen der EIA/CDIF-Standards definierten Meta-Attribute sind optional und können somit im Rahmen eines EIA/CDIF-Austausches entfallen.[763] Tabelle 4-86 dokumentiert sämtliche zwingend vorgeschriebenen Meta-Attribute und gibt

762 In diesem Zusammenhang werden fünf oder mehr Meta-Attribute als überdurchschnittlich viel angesehen.

763 Damit sollte die Benutzung von den standardisierten EIA/CDIF-Meta-Modellen im Rahmen von EIA/CDIF-Austäuschen sehr einfach sein. Dies deshalb, da in EIA/CDIF ausdrücklich optionale Meta-Attribute als optional angesehen werden und dafür auch keine Wertevorgaben definiert sind. Letzteres würde im übrigen bedeuten, daß Meta-Attribute zwingend wären, da im Falle ihrer Abwesenheit (beziehungsweise der Abwesenheit eines ausdrücklichen Wertes) ein Defaultwert eingesetzt werden muß. Dieser Lösungsansatz wurde nebenbei 1997 von der Firma Rational im Rahmen ihres ursprünglichen UML-Vorschlages für die OMG, EIA/CDIF zum Austausch von UML-Modellen zu verwenden, zugrundegelegt (vgl. in diesem Zusammenhang auch [UML97a]).

das AttribuierbareMetaObjekt an, für das es definiert wurde sowie den betroffenen GegenstandsBereich.

Meta-Modell	Name des AttribuierbarenMetaObjekts	Name des Meta-Attributs
FND	RootObject *1*	**CDIFIdentifier** *5*
CMMN	AbstractionLevel *12*	**Name** *13*
	ToolUser *56*	**SystemName** *58*
DFM	Port *1129*	**IsFormal** *1139*
	0:N **ReferencedElement**.*DefinesPath*.-ComponentObject **1**:N *8025*	**SequenceNumber** *6040*
PLAC	AbsolutePoint *10009*	**Position** *10101*
	RelativePoint *10008*	**DeltaX** *10124*
	RelativePoint *10008*	**DeltaY** *10125*
	SemanticObjectReference *10014*	**ReferencedMetaObjectInstance** *10110*

Tabelle 4-86: Die neun zwingend vorgeschriebenen Meta-Attribute der standardisierten EIA/CDIF-Meta-Modelle

4.3.2 Diskussion strukturbezogener Eigenschaften

In diesem Abschnitt wird auf grundlegende Einschränkungen eingegangen, die durch die standardisierten EIA/CDIF-Meta-Meta-Modelle gegeben sind. Es werden hierzu jene Beschränkungen diskutiert, deren Aufhebung im Rahmen der standardisierten EIA/CDIF-Meta-Modelle als wünschenswert erscheinen.

In Abschnitt 4.4, ‘Definition eines EIA/CDIF-konformen Meta-Modells „M2Level“’ Seite 356 unten, wird ein Vorschlag für ein eigenständiges EIA/CDIF-konformes Meta-Modell gemacht, mit dessen Hilfe ohne Erweiterung des minimalen[764] EIA/CDIF-Meta-Meta-Modells die kritisierten, strukturellen Beschränkungen aufgehoben werden können.

764 Das gegenwärtige EIA/CDIF-Meta-Meta-Modell (vgl. [CDIF94b]) ist minimal in der Hinsicht, daß es alle wesentlichen Konstrukte für die erweiterte konzeptionelle Modellierung aufweist und zur Verfügung stellt, wobei keines davon verzichtbar wäre. Es existieren hierzu, vor allem als Reaktion auf Rational’s UML beziehungsweise OMG’s MOF, Erwei-

Fortsetzung folgt auf der nächsten Seite.

4.3.2.1 Einander ausschließende Meta-Beziehungstypen

Bereits im Abschnitt über das EIA/CDIF-Meta-Meta-Modell[765] wurde das Problem angesprochen, daß für die Meta-Modellierung das Konzept der einander ausschließenden Meta-Beziehungstypen zur Verfügung steht. Allerdings ist die Dokumentation des Einsatzes dieser Technik nur graphisch möglich. Somit ist es Werkzeugen, die EIA/CDIF-Meta-Modelle auswerten, nicht möglich, diese Information automatisiert zu extrahieren und zu verarbeiten. Stattdessen müssen beispielsweise Werkzeuge zur Auswertung von Modelldaten aus dem GegenstandsBereich „Datenmodellierung“[766] speziell darauf zugeschnitten werden.

Es wäre wünschenswert, könnte diese für die Auswertung von darauf aufbauenden Modelldaten wesentliche Information aus EIA/CDIF-Transfers automatisiert entnommen und entsprechend weiterverarbeitet werden.

4.3.2.2 Klassifikation von Meta-Beziehungstypen

Bereits im Abschnitt über das EIA/CDIF-Meta-Meta-Modell[767] wurden Einschränkungen in bezug auf die Informationen, die für Meta-Beziehungstypen vorgesehen sind, thematisiert.

Derzeit besteht im Rahmen des EIA/CDIF-Meta-Meta-Modells keine Möglichkeit, ausdrücklich Meta-Beziehungstypen anzugeben, die eine Aggregationsbeziehung repräsentieren. Es wäre daher wünschenswert, könnte man Meta-Beziehungstypen, die aggregierenden Charakter aufweisen[768], als solches charakterisieren.

Zudem ist derzeit für assoziative und charakterisierende Meta-Entitätstypen nicht möglich, festzustellen, welche Meta-Beziehungstypen identitätsstiftenden

---

terungen, die zum Teil auch die in diesem Abschnitt dargestellten Einschränkungen aufheben. Vgl. hierzu auch die Diskussion in [CDIF97d].

765 Vgl. die Ausführungen über die Notation auf Seite 52 (vor Abbildung 3-1, „Das EIA/CDIF-Meta-Meta-Modell“ auf Seite 53).

766 Dieser GegenstandsBereich nutzt die Möglichkeit der Definition von einander ausschließenden Meta-Beziehungstypen. Sie werden unmittelbar vor dem Abschnitt 4.1.3.2.1 auf Seite 391 aufgelistet.

767 Vgl. die Ausführungen im Kapitel 3, „Das EIA/CDIF-Meta-Meta-Modell“ auf Seite 49ff.

768 Beispielsweise könnte man den Meta-Beziehungstyp „0:1 DefinitionObject.*Contains*.-ComponentObject 0:N“ als aggregierend ansehen, da dadurch KomponentenObjekte unzertrennliche Bestandteile von DefinitionsObjekten werden und nur mehr im Kontext der DefinitionsObjekte adressierbar sind.

Charakter aufweisen, also die Identität der entsprechenden Meta-Entitätstypen (mit) festlegen, und welche unabhängig davon Beziehungen zwischen Meta-Entitätstypen ermöglichen. Es wäre daher wünschenswert, könnte man in EIA/CDIF-konformen Meta-Modellen ausdrücklich angeben, welche Meta-Beziehungstypen identitätsstiftenden Charakter für einen Meta-Entitätstyp aufweisen.

4.3.2.3 Der allgemeine Strukturierungsmechanismus

Der im Zusammenhang mit der Definition des standardisierten EIA/CDIF-Meta-Modells für die Datenmodellierung und Datenflußmodellierung eingeführten allgemeinen Strukturierungsmechanismus, weist eine als wesentlich erachtete Einschränkung in bezug auf die Festlegung der KomponentenObjekte auf, aus denen ein DefinitionsObjekt zusammengesetzt werden darf: es kann nicht automatisiert festgestellt werden, aus welchen KomponentenObjekten ein DefinitionsObjekt zusammengesetzt werden darf.

Beispielsweise wird bei der Definition des standardisierten EIA/CDIF-Meta-Modells für den GegenstandsBereich „Datenflußmodellierung“ in Form einer Übersichtstabelle außerhalb der Definitionen der AttribuierbarenMetaObjekte angegeben, aus welchen KomponentenObjekten die DefinitionsObjekte zusammensetzbar sind.[769] In diesem GegenstandsBereich kommt es sogar zu einem Widerspruch im Tabelleninhalt, wenn man den Vergleich zu den textuellen Festlegungen zum Meta-Entitätstyp „StoreDefinition“[770] zieht, bei dem beispielsweise im Meta-Meta-Attribut „Usage“ das KomponentenObjekt „FlowPort“ nicht angegeben wird.

Im Meta-Modell für den GegenstandsBereich „Datenmodellierung“ finden sich hingegen weder eine Übersichtstabelle noch irgendwelche Hinweise auf die gültigen KomponentenObjekte in den Definitionen der Meta-Entitätstypen. Aufgrund der gegebenen Generalisierungshierarchie und unter Kenntnis des Allgemeinen Strukturierungsmechanismus[771] kann davon ausgegangen werden,

769 Vgl. [CDIF96c], Tabelle 5 in Abschnitt 4.1.7.3, Seite 20 oben.

770 Vgl. [CDIF96c], Seite 104 oben.

771 Vgl. die Ausführungen im Abschnitt 4.1.3.1.2, „Der Allgemeine Strukturierungsmechanismus“ auf Seite 229ff oben.

daß die DefinitionsObjekte der Datenmodellierung[772] nur aus dem KomponentenObjekt „Attribute“ beziehungsweise dessen Subtyp „ProjectedAttribute“ gebildet werden dürfen.

Die Schlußfolgerung – alle DefinitionsObjekte dürfen aus allen KomponentenObjekten gebildet werden – die für den GegenstandsBereich „Datenmodellierung“ gilt, ist für den GegenstandsBereich „Datenflußmodellierung“ nicht zulässig, da die entsprechenden DefinitionsObjekte in der Regel aus nur einer Teilmenge der im Meta-Modell definierten KomponentenObjekte konstruierbar sind.[773, 774]

Es wäre daher wünschenswert, könnte für jedes DefinitionsObjekt ausdrücklich im Rahmen von EIA/CDF angegeben werden, aus welchen KomponentenObjekten es sich zusammensetzen darf.

---

772 Es handelt sich hierbei um die Meta-Entitätstypen „Cluster“, „Entity“, „Relationship“, „Role“ und „RolePlayer“.

773 Dies wird anhand eines Vergleiches der DefinitionsObjekte „FlowDefinition“ (zulässige Komponenten: „Attribute“, „Flow“) und „DFMProcessDefinition“ (zulässige Komponenten: „Attribute“, „DFMProcess“, „ExternalAgent“, „Flow“, „FlowPort“ und„Store“) sofort klar.

774 Wenn unter Unkenntnis der einzelnen Meta-Modelle das Integrierte EIA/CDIF-Meta-Modell analysiert werden würde, so wäre es nicht möglich, daraus die Information abzuleiten, aus welchen KomponentenObjekten jene DefinitionsObjekte sich zusammensetzen dürfen, die ursprünglich für den GegenstandsBereich „Datenmodellierung“ festgelegt wurden!

4.4 Definition eines EIA/CDIF-konformen Meta-Modells „M2Level“

In diesem Abschnitt wird ein EIA/CDIF-konformes Meta-Modell namens „M2Level“[775] zur Diskussion gestellt, das im wesentlichen die in Abschnitt 4.3.2 auf Seite 352 oben angeführten Einschränkungen überwinden helfen soll.[776] Die Darstellung erfolgt mit der gleichen Gliederung wie sie in dieser Arbeit für die standardisierten EIA/CDIF Modelle benutzt wurde.

Dieses Meta-Modell stellt den Versuch dar, strukturbezogene Informationen, die eigentlich auf M3-Ebene vorgesehen werden sollten, auf M2-Ebene erfaßbar und damit auch automatisiert austauschbar und auswertbar zu machen. Insoferne unterscheidet es sich grundlegend von allen bisherigen standardisierten EIA/CDIF-Meta-Modellen und jenen, die sich in Planung befinden.[777] Gleichzeitig soll damit gezeigt werden, daß es prinzipiell möglich ist, fundamentale neue Konzepte in die M3-Schicht einzuführen und zu benutzen, ohne das Meta-Meta-Modell selbst ad hoc[778] ändern zu müssen. Sofern es als notwendig erachtet wird, auch Funktionen beziehungsweise Methoden ausdrücklich zu modellieren und M2-Konzepten (eventuell aggregierend) zuzuweisen, bräuchte nur das hier vorgestellte Meta-Modell entsprechend erweitert werden.[779]

Dieses Meta-Modell weist unter anderem die folgenden Besonderheiten auf:

– Es definiert einen Meta-Entitätstyp „M2LevelObject“, der als Wurzeltyp für sämtliche Meta-Entitätstypen dient, die strukturbezogene (M3-Schicht-)

---

775 Dementsprechend wird dafür die Abkürzung „M2L“ vorgeschlagen.

776 Dieses Meta-Modell stellt einen möglichen Vorschlag dar, wie man das EIA/CDIF-Meta-Meta-Modell auf M2-Ebene semantisch anreichern kann.

777 Keines der bisherigen EIA/CDIF-Meta-Modelle unternimmt den Versuch, Meta-Objekte selbst auf M2-Ebene zu *referenzieren*. In PLAC wurde im Gegensatz dazu das *Referenzieren von Meta-Objekt-Instanzen* erstmals in die EIA/CDIF-Meta-Modellierung eingeführt: der Meta-Entitätstyp „SemanticObjectReference“ weist ein Meta-Attribut „ReferencedMetaObjectInstance“ auf, das den Wert des Meta-Attributs „CDIFIdentifier“ des referenzierten Meta-Objekts aufnimmt.

778 Eine Änderung des Meta-Meta-Modells kann theoretisch von EIA oder ISO/IETC zu einem späteren Zeitpunkt jederzeit vorgenommen werden.

779 Sofern man dies als Vorteil ansehen möchte, wird durch die Nutzung dieses hier vorgeschlagenen Meta-Modells das EIA/CDIF-Meta-Meta-Modell unverändert belassen und bleibt minimal (beispielsweise im Vergleich zum OMG Meta-Meta-Modell MOF) und daher einfacher für den EIA/CDIF-Austausch handhabbar.

Informationen auf M2-Ebene repräsentieren sollen. Damit lassen sich sämtliche in diesem GegenstandsBereich definierten Konzepte von allen anderen standardisierten EIA/CDIF-Meta-Modellen unterscheiden.

– In diesem GegenstandsBereich werden Meta-Objekte ausdrücklich dadurch referenziert, indem ein Meta-Attribut[780] den Wert des entsprechenden Meta-Meta-Attributs „CDIFMetaIdentifier“ als Verweiswert aufnimmt.

Die folgende Abbildung gibt einen Überblick über dieses EIA/CDIF-konforme Meta-Modell.

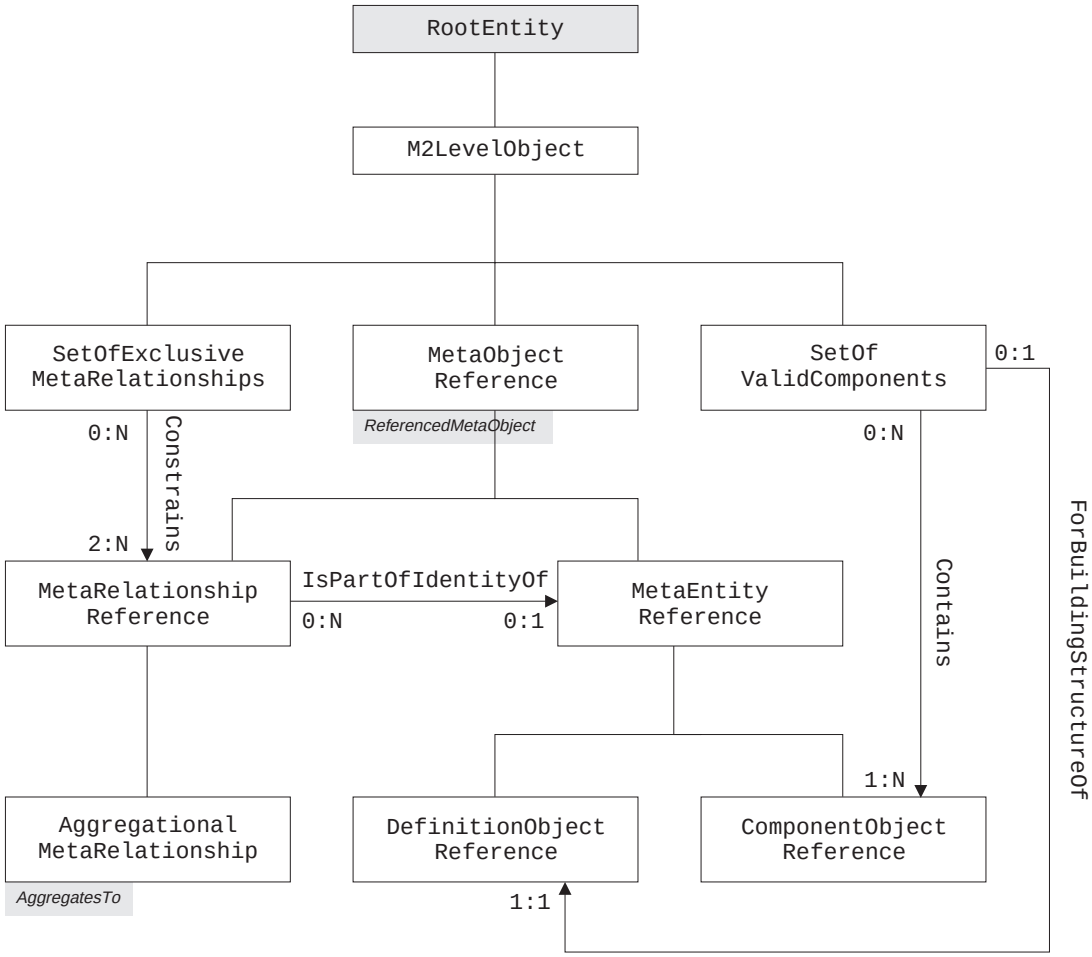


Abbildung 4-12: Das EIA/CDIF-konforme Meta-Modell „M2Level“

780 Es handelt sich konkret um das Meta-Attribut „ReferencedMetaObject“ des Meta-Entitätstyps „MetaObjectReference“ (vgl. Abschnitt 4.4.1.6 auf Seite 363ff unten), das als Wurzeltyp für sämtliche Typen mit Meta-Objekt-Referenzierungscharakter dient.

4.4.1 Definitionen der „M2L“-Konzepte

Die detaillierten Definitionen für die MetaObjekte des EIA/CDIF-konformen Meta-Modells „M2Level“ erfolgen in alphabetischer Reihenfolge, getrennt nach Meta-Entitätstypen und Meta-Beziehungstypen. Die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“ beginnen mit einem Großbuchstaben[781] und weisen dadurch darauf hin, daß es sich hierbei um keine offiziell standardisierten Definitionen handelt.

4.4.1.1 Meta-Entitätstyp „AggregationalMetaRelationship“

Der Meta-Entitätstyp „AggregationalMetaRelationship“ erlaubt die Festlegung, daß der referenzierte Meta-Beziehungstyp für das Aggregieren[782] benutzt wird. Tabelle 4-87 beschreibt den Meta-Entitätstyp „AggregationalMetaRelationship“.

Name des Attributs	Bedeutung
Name	***AggregationalMetaRelationship***
CDIFMetaIdentifier	***ME_05***
SubtypeOf	*MetaRelationshipReference*
SupertypeOf	–
Description	*Erlaubt die Festlegung, daß der referenzierte Meta-Beziehungstyp aggregierend ist.*
Usage	*Eine Instanz von diesem Typ sollte immer dann gebildet werden, wenn ein Meta-Beziehungstyp aggregierend wirkt.*
Aliases	–
Constraints	*Der Meta-Entitätstyp, zu dem hin über den referenzierten Meta-Beziehungstyp aggregiert wird, muß im Maximumwert der entsprechenden Kardinalität den Wert Eins aufweisen.*
Type	*Kernel*

Tabelle 4-87: Meta-Entitätstyp „AggregationalMetaRelationship“

781 Der Einfachheit halber werden dafür die Akronyme der entsprechenden Meta-Typen benutzt, gefolgt von einem Unterstrich und einer aufsteigenden Nummer.

782 Vgl. in diesem Zusammenhang beispielsweise [ElmNav89], Seite 438 unten, oder [BaCeNa92], Seite 19ff.

Der Meta-Entitätstyp „AggregationalMetaRelationship“ verfügt über folgendes, vererbtes Meta-Attribut:

– „ReferencedMetaObject“ (von „MetaObjectReference“).

Der Meta-Entitätstyp „AggregationalMetaRelationship“ verfügt darüber hinaus über ein einziges, zwingend vorgeschriebenes Meta-Attribut namens „AggregatesTo“.

4.4.1.1.1 Meta-Attribut „AggregatesTo“

Das zwingend vorgeschriebene Meta-Attribut „AggregatesTo“ vom EIA/CDIF-Datentyp „Enumerated“ weist eines der beiden aufzählbaren Werte „Source“ oder „Destination“ auf. Damit kann angegeben werden, zu welchem der beiden assoziierten Meta-Entitätstypen hin aggregiert wird. Tabelle 4-88 beschreibt das Meta-Attribut „AggregatesTo“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***AggregatesTo***
CDIFMetaIdentifier	***MA_02***
Description	*Die Position des Meta-Entitätsyps im vollqualifizierten Namen des Meta-Beziehungstyps, zu dem hin aggregiert wird.*
Usage	*Wenn der Quell-Meta-Entitätstyp das Aggregat darstellt, dann muß hier der Wert „Source“ eingetragen werden, „Destination“ sonst.*
Aliases	–
Constraints	*Der Maximumwert der Kardinalität des Aggregats beziehungsweise des Meta-Entitätstyps, zu dem hin aggregiert wird, muß Eins sein.*
DataType	*Enumerated*
Domain	*Source, Destination*
Length	–
IsOptional	*False*

Tabelle 4-88: Meta-Attribut „AggregatesTo“

4.4.1.2 Meta-Entitätstyp „ComponentObjectReference“

Der Meta-Entitätstyp „ComponentObjectReference“ referenziert einen Subtypen des Meta-Entitätstyps „ComponentObject“. Mit Hilfe des lokalen Meta-Beziehungstyps „0:N **SetOfValidComponents**.*Contains*.ComponentObject-Reference 1:N“ werden die referenzierten Subtypen vom Typ „Component-Object“ zusammengefaßt, aus denen der referenzierte Subtyp von „DefinitionObject“ aus dem Beziehungstyp „0:1 **SetOfValidComponents**.*For-BuildingStructureOf*.DefinitionObjectReference 1:1“ gebildet werden darf. Tabelle 4-89 beschreibt den Meta-Entitätstyp „ComponentObjectReference“.

Name des Attributs	Bedeutung
Name	***ComponentObjectReference***
CDIFMetaIdentifier	***ME_09***
SubtypeOf	*MetaEntityReference*
SupertypeOf	–
Description	*Eine Referenz auf einen Subtyp von „ComponentObject“.*
Usage	–
Aliases	–
Constraints	*Darf nur Meta-Entitätstypen referenzieren, die Subtypen von ComponentObject sind.*
Type	*Kernel*

Tabelle 4-89: Meta-Entitätstyp „ComponentObjectReference“

Der Meta-Entitätstyp „ComponentObjectReference“ verfügt über das folgende, vererbte Meta-Attribut:

– „ReferencedMetaObject“ (von „MetaObjectReference“).

4.4.1.3 Meta-Entitätstyp „DefinitionObjectReference“

Der Meta-Entitätstyp „DefinitionObjectReference“ referenziert einen Subtypen des Meta-Entitätstyps „DefinitionObject“. Mit Hilfe des lokalen Meta-Beziehungstyps „0:1 **SetOfValidComponents**.*ForBuildingStructureOf*.Definition-ObjectReference 1:1“ werden dem referenzierten Subtyp von „DefinitionObject“

die Referenzen auf die zulässigen KomponentenObjekte zugeordnet. Tabelle 4-90 beschreibt den Meta-Entitätstyp „DefinitionObjectReference“.

Name des Attributs	Bedeutung
Name	***DefinitionObjectReference***
CDIFMetaIdentifier	***ME_08***
SubtypeOf	*MetaEntityReference*
SupertypeOf	–
Description	*Eine Referenz auf einen Subtyp von „DefinitionObject“.*
Usage	–
Aliases	–
Constraints	*Darf nur Meta-Entitätstypen referenzieren, die Subtypen von DefinitionObject sind.*
Type	*Kernel*

Tabelle 4-90: Meta-Entitätstyp „DefinitionObjectReference“

Der Meta-Entitätstyp „DefinitionObjectReference“ verfügt über das folgende, vererbte Meta-Attribut:

– „ReferencedMetaObject“ (von „MetaObjectReference“).

4.4.1.4 Meta-Entitätstyp „M2LevelObject“

Der Meta-Entitätstyp „M2LevelObject“ stellt die Wurzel für sämtliche Meta-Objekte dar, die für den Zweck definiert sind, Strukturinformationen über die M2-Schicht vorzuhalten.[783]

Tabelle 4-91 beschreibt den Meta-Entitätstyp „M2LevelObject“.

783 Die Konzepte und Informationen dieses Meta-Modells könnten durch eine entsprechende Änderung des EIA/CDIF-Meta-Meta-Modells irgendwann einmal in der Zukunft obsolet werden.

Name des Attributs	Bedeutung
Name	***M2LevelObject***
CDIFMetaIdentifier	***ME_01***
SubtypeOf	*RootEntity*
SupertypeOf	*MetaObjectReference* *SetOfExclusiveMetaRelationships* *SetOfValidComponents*
Description	*Ein Objekt aus beziehungsweise zur Dokumentation der M2-Ebene.*
Usage	*Bildet den Wurzeltyp sämtlicher Meta-Objekte, die strukturbezogene Informationen über Meta-Objekte der M2-Ebene vorhalten, die eigentlich in der M3-Schicht vorgesehen werden sollten.*
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 4-91: Meta-Entitätstyp „M2LevelObject“

4.4.1.5 Meta-Entitätstyp „MetaEntityReference“

Der Meta-Entitätstyp „MetaEntityReference“ referenziert eine Instanz vom Meta-Meta-Entitätstyp „MetaEntity“[784]. Sofern für einen Meta-Entitätstyp eine Existenzabhängigkeit über einen Meta-Beziehungstyp besteht, wie dies für Meta-Entitätstypen vom Typ „Associative“ beziehungsweise „Characteristic“ der Fall ist, kann dies mit Hilfe des lokalen Meta-Beziehungstyps „0:N Meta-RelationshipReference.*IsPartOfIdentityOf*.MetaEntityReference 0:1“ ausgedrückt werden. Diese Existenzabhängigkeit wirkt umgekehrt auch als identitätsstiftend für die entsprechenden, referenzierten Meta-Entitätstypen.[785]

Tabelle 4-92 beschreibt den Meta-Entitätstyp „MetaEntityReference“.

784 Es handelt sich hierbei um den Meta-Entitätstyp „RootEntity“ beziehungsweise um einen seiner Subtypen.

785 Vgl. auch die entsprechenden Ausführungen in [Codd79], [Date85] oder auch die darauf bezogenen Diskussionen in [Flat90b].

Name des Attributs	Bedeutung
Name	***MetaEntityReference***
CDIFMetaIdentifier	***ME_04***
SubtypeOf	*MetaObjectReference*
SupertypeOf	*ComponentObjectReference* *DefinitionObejctReference*
Description	*Eine Referenz auf einen Meta-Entitätstyp, das heißt einer Instanz vom Meta-Meta-Entitätstyp „MetaEntity“.*
Usage	–
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 4-92: Meta-Entitätstyp „MetaEntityReference“

Der Meta-Entitätstyp „MetaEntityReference“ verfügt über das folgende, vererbte Meta-Attribut:

– „ReferencedMetaObject“ (von „MetaObjectReference“).

4.4.1.6 Meta-Entitätstyp „MetaObjectReference“

Der Meta-Entitätstyp „MetaObjectReference“ referenziert eine Instanz vom Meta-Meta-Entitätstyp „MetaObject“. Tabelle 4-93 beschreibt den Meta-Entitätstyp „MetaObjectReference“.

Name des Attributs	Bedeutung
Name	***MetaObjectReference***
CDIFMetaIdentifier	***ME_02***
SubtypeOf	*M2LevelObject*
SupertypeOf	*MetaEntityReference* *MetaRelationshipReference*
Description	*Eine Referenz auf eine Instanz vom Meta-Meta-Entitätstyp „MetaObject“.*
Usage	–

Name des Attributs	Bedeutung
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 4-93: Meta-Entitätstyp „MetaObjectReference“

Der Meta-Entitätstyp „MetaObjectReference“ verfügt über ein einziges, zwingend vorgeschriebenes Meta-Attribut namens „ReferencedMetaObject“.

4.4.1.6.1 Meta-Attribut „ReferencedMetaObject“

Das zwingend vorgeschriebene Meta-Attribut „ReferencedMetaObject“ ist vom EIA/CDIF-Datentyp „Identifier“ und beinhaltet den Wert des Meta-Meta-Attributs „CDIFMetaIdentifier“ von jenem MetaObjekt, das referenziert wird. Tabelle 4-94 beschreibt das Meta-Attribut „ReferencedMetaObject“.

Name des Attributs	Bedeutung
Meta-Attribute Name	***ReferencedMetaObject***
CDIFMetaIdentifier	***MA_01***
Description	*Enthält den Wert des Meta-Meta-Attributs „CDIFMetaIdentifier“ jenes MetaObjektes, das referenziert wird.*
Usage	–
Aliases	–
Constraints	*Der Wert dieses Meta-Attributs muß auch im Meta-Meta-Attribut „CDIFMetaIdentifier“ eines Meta-Objektes (des CDIF-Austausches enthalten) sein.*
DataType	*Identifier*
Domain	–
Length	–
IsOptional	*False*

Tabelle 4-94: Meta-Attribut „ReferencedMetaObject“

4.4.1.7 Meta-Entitätstyp „MetaRelationshipReference“

Der Meta-Entitätstyp „MetaRelationshipReference“ referenziert eine Instanz vom Meta-Meta-Entitätstyp „MetaRelationship“[786]. Somit können mit Hilfe des lokalen Meta-Beziehungstyps „0:N MetaRelationshipReference.*IsPartOfIdentity-Of*.MetaEntityReference 1:N“ die Meta-Beziehungstypen bestimmt werden, die assoziative beziehungsweise charakterisierende Meta-Entitätstypen in ihrer Existenz (mit)begründen.[787]

Zudem kann mit Hilfe des Meta-Beziehungstyps „0:N **SetOfExclusiveMeta-Relationships**.*Constrains*.MetaRelationshipReference 2:N“ angegeben werden, welche referenzierten Meta-Beziehungstypen gemeinsam einander ausschließen.[788]

Tabelle 4-95 beschreibt den Meta-Entitätstyp „MetaRelationshipReference“.

Name des Attributs	Bedeutung
Name	***MetaRelationshipReference***
CDIFMetaIdentifier	***ME_03***
SubtypeOf	*MetaObjectReference*
SupertypeOf	*AggregationalMetaRelationship*
Description	*Eine Referenz auf einen Meta-Beziehungstyp, das heißt einer Instanz vom Meta-Meta-Entitätstyp „MetaRelationship“.*
Usage	–
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 4-95: Meta-Entitätstyp „MetaRelationshipReference“

786 Es handelt sich hierbei um den Meta-Beziehungstyp „RootEntity.*IsRelatedTo*.RootEntity“ beziehungsweise um einen seiner Subtypen.

787 Bei charakterisierenden Meta-Entitätstypen wird *genau ein* einziger Meta-Beziehungstyp existenzbegründend wirken, bei assoziativen Meta-Entitätstypen *mindestens zwei* oder mehrere Meta-Beziehungstypen.

788 Eine weitere Variante auf Meta-Meta-Modellebene wäre der Einsatz des Rollenkonzepts, wie dies im GegenstandsBereich „Datenmodellierung“ (siehe Abschnitt 4.1.3.2, „DMOD – Das EIA/CDIF-Meta-Modell für den GegenstandsBereich „Datenmodellierung“ auf Seite 241ff oben) der Fall ist und in [CDIF97d] vorgeschlagen wird.

Der Meta-Entitätstyp „MetaRelationshipReference“ verfügt über das folgende, vererbte Meta-Attribut:

– „ReferencedMetaObject“ (von „MetaObjectReference“).

4.4.1.8 Meta-Entitätstyp „SetOfExclusiveMetaRelationships“

Der Meta-Entitätstyp „SetOfExclusiveMetaRelationships“ erlaubt es, festzulegen, welche referenzierten Meta-Beziehungstypen einander ausschließen. Dazu wird einfach der lokale Meta-Beziehungstyp „0:N **SetOfExclusiveMetaRelationships**.*Constrains*.MetaRelationshipReference 2:N“ entsprechend oft instanziiert.

Tabelle 4-96 beschreibt den Meta-Entitätstyp „SetOfExclusiveMetaRelationships“.

Name des Attributs	Bedeutung
Name	***SetOfExclusiveMetaRelationships***
CDIFMetaIdentifier	***ME_06***
SubtypeOf	*M2LevelObject*
SupertypeOf	–
Description	*Eine Instanz dieses Meta-Entitätstyps erlaubt es, über den lokalen Meta-Beziehungstyp „0:N **SetOfExclusiveMetaRelationships**.Constrains.MetaRelationshipReference 2:N“ jene Meta-Beziehungstypen über ihre Referenzen anzugeben, die einander ausschließen.*
Usage	–
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 4-96: Meta-Entitätstyp „SetOfExclusiveMetaRelationships“

4.4.1.9 Meta-Entitätstyp „SetOfValidComponents“

Der Meta-Entitätstyp „SetOfValidComponents“ erlaubt es, pro Instanz für einen referenzierten Subtypen vom Typ „DefinitionObject“ (über den lokalen Meta-

Beziehungstyp „0:1 **SetOfValidComponents**.*ForBuildingStructureOf*.DefinitionObjectReference 1:1“) jene Referenzen der Subtypen von „ComponentObject“ (über den lokalen Meta-Beziehungstyp „0:N **SetOfValidComponents**.*Contains*.ComponentObjectReference 1:N“) zuzuordnen, aus denen das referenzierte DefinitionsObjekt zusammengesetzt sein darf.

Die einzelnen referenzierten KomponentenObjekte dürfen so zusammengesetzt werden, wie es das Meta-Attribut „Operator“ des referenzierten DefinitionsObjektes dafür angibt.[789]

Tabelle 4-97 beschreibt den Meta-Entitätstyp „SetOfValidComponents“.

Name des Attributs	Bedeutung
Name	***SetOfValidComponents***
CDIFMetaIdentifier	***ME_07***
SubtypeOf	*M2LevelObject*
SupertypeOf	–
Description	*Eine Instanz dieses Meta-Entitätstyps erlaubt es, gemeinsam mit den Meta-Beziehungstypen „0:1* ***SetOfValidComponents****.ForBuildingStructureOf.DefinitionObjectReference 1:1“ und „0:N* ***SetOfValidComponents****.Contains.ComponentObjectReference 1:N“ anzugeben, welche referenzierten DefinitionsObjekte aus welchen referenzierten KomponentenObjekten aufgebaut sein dürfen.*
Usage	–
Aliases	–
Constraints	–
Type	*Kernel*

Tabelle 4-97: Meta-Entitätstyp „SetOfValidComponents“

789 Vgl. die entsprechenden Ausführungen im Abschnitt 4.1.3.1.2, „Der Allgemeine Strukturierungsmechanismus“ auf Seite 229ff oben. Es handelt sich um die aufzählbaren Werte „AND“, „OR“ und „XOR“.
(Würde man die Möglichkeit vorsehen, für die einzelnen erlaubten KomponentenObjekte noch die Anzahl ihres Auftretens genauer zu spezifizieren, könnte man damit grundsätzlich die Strukturierungsmöglichkeiten beispielsweise von SGML (XML) mit dem allgemeinen Strukturierungsmechanismus ausdrücken.)

4.4.1.10 Meta-Beziehungstyp „0:N Meta-RelationshipReference.-*IsPartOfIdentityOf*.MetaEntityReference 0:1“

Der binäre Meta-Beziehungstyp „IsPartOfIdentityOf“ mit dem vollqualifizierten Namen „0:N MetaRelationshipReference.*IsPartOfIdentityOf*.MetaEntityReference 0:1“ erlaubt die Zuordnung von Instanzen vom Typ „MetaRelationshipReference“ zu Instanzen vom Typ „MetaEntityReference“. Damit kann festgehalten werden, welche referenzierten Meta-Entitätstypen von welchen referenzierten Meta-Beziehungstypen einen Teil ihrer Identität erhalten und damit davon existenzabhängig sind.

Aufgrund der Kardinalität von „0:N“ zu „0:1“ *kann* eine Instanz vom Typ „MetaRelationshipReference“ mit maximal einer Instanz vom Typ „MetaEntityReference“ in Verbindung stehen. Umgekehrt kann eine Instanz vom Typ „MetaEntityReference“ mit ein oder beliebig vielen Instanzen vom Typ „MetaRelationshipReference“ assoziiert sein. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-98 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „IsPartOfIdentityOf“.

Name des Attributs	Bedeutung
Name	***IsPartOfIdentityOf***
CDIFMetaIdentifier	***MR_04***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*0*
MaxDestCard	*1*
Description	*Mit Hilfe von Instanzen dieses Meta-Beziehungstyps kann angegeben werden, welche referenzierten Meta-Beziehungstypen identitätsstiftend für welche referenzierten Meta-Entitätstypen sind.*
Usage	–

Name des Attributs	Bedeutung
Aliases	–
Constraints	*Für charakterisierende Meta-Entitätstypen muß genau eine Instanz von diesem Meta-Beziehungstyp gebildet werden, für assoziative Meta-Entitätstypen mindestens zwei Instanzen.*

Tabelle 4-98: Meta- Beziehungstyp „0:N MetaRelationshipReference.*IsPartOfIdentity-Of*.MetaEntityReference 0:1“

4.4.1.11 Meta-Beziehungstyp „0:N SetOfExclusiveMeta-Relationships.*Constrains*.MetaRelationshipReference 2:N“

Der binäre Meta-Beziehungstyp „Constrains“ mit dem vollqualifizierten Namen „0:N **SetOfExclusiveMetaRelationships**.*Constrains*.MetaRelationshipReference 2:N“ erlaubt die Zuordnung von Instanzen vom Typ „SetOfExclusiveMeta-Relationships“ zu Instanzen vom Typ „MetaRelationshipReference“. Damit kann festgehalten werden, welche referenzierten Meta-Beziehungstypen einander ausschließen.

Aufgrund der Kardinalität von „0:N“ zu „2:N“ *muß* eine Instanz vom Typ „SetOf-ExclusiveMetaRelationships“ mit *zwei* oder beliebig vielen weiteren Instanzen vom Typ „MetaRelationshipReference“ in Verbindung stehen. Umgekehrt kann eine Instanz vom Typ „MetaRelationshipReference“ mit ein oder beliebig vielen Instanzen vom Typ „SetOfExclusiveMetaRelationships“ assoziiert sein. Somit ist dieser Meta-Beziehungstyp für den Meta-Entitätstyp „SetOfExclusiveMeta-Relationships“ zwingend vorgeschrieben. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-99 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „Constrains“.

Name des Attributs	Bedeutung
Name	***Constrains***
CDIFMetaIdentifier	***MR_01***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*

MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*2*
MaxDestCard	*N*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität vom Typ „SetOf-ExclusiveMetaRelationships“ mit zwei oder mehr Instanzen vom Typ „MetaRelationshipReference“. Die referenzierten Meta-Beziehungstypen schließen einander aus.*
Usage	–
Aliases	–
Constraints	*Sämtliche referenzierten Meta-Beziehungstypen müssen entweder als Quelle oder Ziel denselben Meta-Entitätstyp aufweisen, für den sie einander ausschließen.*

Tabelle 4-99: Meta- Beziehungstyp „0:N **SetOfExclusiveMetaRelationships**.-*Constrains*.MetaRelationshipReference 2:N“

4.4.1.12 Meta-Beziehungstyp „0:N SetOfValidComponents.-*Contains*.ComponentObjectReference 1:N“

Der binäre Meta-Beziehungstyp „Contains“ mit dem vollqualifizierten Namen „0:N **SetOfValidComponents**.*Contains*.ComponentObjectReference 1:N“ erlaubt die Zuordnung von Instanzen vom Typ „SetOfValidComponents“ zu Instanzen vom Typ „ComponentObjectReference“. Damit kann festgehalten werden, welche referenzierten KomponentenObjekte für den Aufbau einer Struktur eines referenzierten DefinitionsObjektes zur Verfügung stehen.

Aufgrund der Kardinalität von „0:N“ zu „1:N“ *muß* eine Instanz vom Typ „SetOf-ValidComponents“ mit *ein* oder beliebig vielen weiteren Instanzen vom Typ „ComponentObjectReference“ in Verbindung stehen. Umgekehrt kann eine Instanz vom Typ „ComponentObjectReference“ mit ein oder beliebig vielen Instanzen vom Typ „SetOfValidComponents“ assoziiert sein. Somit ist dieser Meta-Beziehungstyp für den Meta-Entitätstyp „SetOfValidComponents“ zwingend vorgeschrieben. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-100 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „Contains“.

Name des Attributs	Bedeutung
Name	***Contains***
CDIFMetaIdentifier	***MR_02***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*N*
MinDestCard	*1*
MaxDestCard	*N*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität vom Typ „*SetOf-Vali*dComponents“ mit ein oder mehreren Instanzen vom Typ „ComponentObjectReference“.* *Die referenzierten KomponentenObjekte stehen für die Bildung der Struktur des im Meta-Beziehungstyp „0:1* ***SetOfValidComponents****.-ForBuildingStructureOf.DefinitionObjectReference 1:1“ referenzierten DefinitionsObjekt zur Verfügung.* *Die Art der Verwendung der referenzierten KomponentenObjekte wird durch den Wert im Meta-Attribut „Operator“ des referenzierten DefinitionsObjektes bestimmt.*
Usage	–
Aliases	–
Constraints	–

Tabelle 4-100: Meta- Beziehungstyp „0:N **SetOfValidComponents**.*Contains*.-ComponentObjectReference 1:N“

4.4.1.13 Meta-Beziehungstyp „0:1 SetOfValidComponents.*ForBuildingStructureOf*.DefinitionObjectReference 1:1“

Der binäre Meta-Beziehungstyp „ForBuildingStructureOf“ mit dem vollqualifizierten Namen „0:1 **SetOfValidComponents**.*ForBuildingStructureOf*.DefinitionObjectReference 1:1“ erlaubt die Zuordnung von Instanzen vom Typ „SetOfValidComponents“ zu Instanzen vom Typ „DefinitionObjectReference“. Damit

kann festgehalten werden, für welches referenzierte DefinitionsObjekt eine MengeGültigerKomponenten definiert wird.

Aufgrund der Kardinalität von „0:1“ zu „1:1“ *muß genau* eine Instanz vom Typ „SetOfValidComponents“ mit *einer* Instanz vom Typ „DefinitionObject-Reference“ in Verbindung stehen. Umgekehrt kann eine Instanz vom Typ „DefinitionObjectReference“ mit maximal einer Instanz vom Typ „SetOfValid-Components“ assoziiert sein. Somit ist dieser Meta-Beziehungstyp für den Meta-Entitätstyp „SetOfValidComponents“ zwingend vorgeschrieben. Für diesen Meta-Beziehungstyp werden keine eigenen Meta-Attribute definiert.

Die folgende Tabelle 4-101 beinhaltet sämtliche Definitionen für den Meta-Beziehungstyp „ForBuildingStructureOf“.

Name des Attributs	**Bedeutung**
Name	***ForBuildingStructureOf***
CDIFMetaIdentifier	***MR_03***
SubtypeOf	*RootEntity.IsRelatedTo.RootEntity*
MinSourceCard	*0*
MaxSourceCard	*1*
MinDestCard	*1*
MaxDestCard	*1*
Description	*Dieser Meta-Beziehungstyp assoziiert eine Entität vom Typ „SetOf-ValidComponents“ mit genau einer Instanz vom Typ „Definition-ObjectReference“.* *Mit Hilfe des Meta-Beziehungstyps „0:N* ***SetOfValidComponents****.-Contains.ComponentObjectReference 1:N“ können jene KomponentenObjekte festgelegt werden, aus denen das referenzierte DefinitionsObjekt bestehen darf.*
Usage	–
Aliases	–
Constraints	–

Tabelle 4-101: Meta- Beziehungstyp „0:1 **SetOfValidComponents**.*ForBuilding-StructureOf.*DefinitionObjectReference 1:1“

4.4.2 Referenzierte Meta-Objekte

In diesem GegenstandsBereich werden zudem über die Vererbung entlang der Generalisierungshierarchie die Meta-Objekte „RootObject“[790], „RootEntity“[791] sowie „RootEntity.*IsRelatedTo*.RootEntity“[792] benützt.

4.4.3 Auswertungen der Definitionen

Zunächst wird in diesem Abschnitt eine strukturelle Übersicht über das Meta-Modell gegeben, indem einfache Auswertungen tabellarisch aufbereitet werden. Daran anschließend werden sämtliche AttribuierbarenMetaObjekte, die ausdrücklich im Meta-Modell benutzt werden,[793] in der dem Meta-Modell entsprechenden Generalisierungshierarchie dargestellt, wobei MetaObjekte auf derselben Stufe alphabetisch sortiert sind. Abschließend werden sämtliche AttribuierbarenMetaObjekte in alphabetischer Reihenfolge summarisch angeführt.

Sämtliche Auflistungen in diesem Abschnitt erfolgen anhand der voll qualifizierten Namen der AttribuierbarenMetaObjekte, wobei die Namen von Meta-Beziehungstypen immer kursiv gesetzt sind. Die Namen von Meta-Entitätstypen, deren Instanzen zwingend in Beziehungen auftreten müssen, werden immer fett hervorgehoben. Die Kardinalitäten der Meta-Beziehungstypen werden ausdrücklich angeführt, sowie die Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“[794]. Sofern für AttribuierbareMetaObjekte die Mehrfachvererbung aufgrund der Definition vorgesehen ist, wird der gesamte vollqualifizierte Name kursiv gesetzt. Sämtliche lokal definierten Meta-Attribute werden angeführt, wobei zusätzlich ihr EIA/CDIF-Datentyp in kleiner Schrift mitangegeben ist.[795]

---

790 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.1 auf Seite 158 vorgestellt.

791 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.2 auf Seite 163 vorgestellt.

792 Die Definition wurde bereits weiter oben im Abschnitt 4.1.1.1.3 auf Seite 164 vorgestellt.

793 Es handelt sich also um Instanzen jener SammelbarenMetaObjekte, die in Instanzen des Meta-Meta-Beziehungstyps „0:N **CollectableMetaObject**.*IsUsedIn*.SubjectArea 1:N“ für die Instanz des entsprechenden GegenstandBereiches enthalten sind.

794 Die Darstellung erfolgt in der vorgesehenen Verkodierung des EIA/CDIF-Datentyps „Identifier“, indem der Wert in Sterne eingefaßt wird (vgl. [CDIF94e], Seiten 11 und 20 unten, „IdentifierValue“).

795 Im Falle des aufzählbaren Datentyps werden auch die gültigen Werte der entsprechenden Wertemenge dargestellt.

4.4.3.1 Strukturelle Übersicht

Das Meta-Modell „M2Level“ definiert insgesamt 23 SammelbareMetaObjekte, wobei die einzige Instanz „RootObject“ des Meta-Meta-Entitätstyps AttribuierbaresMetaObjekt, der Meta-Entitätstyp „RootEntity“ und der Meta-Beziehungstyp „RootEntity.*IsRelatedTo*.RootEntity“, die alle über die Generalisierungshierarchie eingebunden werden, mitgezählt sind. Tabelle 4-102 stellt die Aufstellung dieser SammelbarenMetaObjekte tabellarisch dar.

Anzahl Instanzen vom Typ AMO[796]	1
Anzahl Instanzen vom Typ ME	10
Anzahl Instanzen vom Typ MR	5
Anzahl Instanzen vom Typ MA	7
Anzahl (Summe) Instanzen vom Typ CMO	23

Tabelle 4-102: Verteilung der sammelbaren Objekte des Meta-Modells „M2Level“

Tabelle 4-103 führt weitere Analyseergebnisse über die strukturellen Eigenschaften dieses Meta-Modells an, wobei die referenzierten AttribuierbarenMetaObjekte darin nicht mehr enthalten sind.

796 Hier wird die einzige direkte Instanz von AMO angeführt, nämlich „RootObject“.

Anzahl zwingend vorgeschriebener MA		2
Anzahl optionaler MA		0
Meta-Objekte mit/ohne Meta-Attribute	mit MA	ohne MA
Anzahl Instanzen vom Typ AMO	0	0
Anzahl Instanzen vom Typ ME	2	7
Anzahl Instanzen vom Typ MR	0	4
Anzahl von zwingend vorgeschriebenen MR[797]		3
Anzahl von ME, die an zwingend vorgeschriebenen MR teilhaben		2
Anzahl der Blätter (Instanzen vom Typ AMO) im Meta-Modellbaum		9
Anzahl von AMOs mit Mehrfachvererbung		0

Tabelle 4-103: Strukturelle Übersicht über das Meta-Modell für „M2Level“ [798]

Daraus läßt sich erkennen, daß für diesen GegenstandsBereich die Meta-Entitätstypen im Verhältnis von 2:1 zu den Meta-Beziehungstypen stehen. Des weiteren ist es interessant, festzustellen, daß für Meta-Beziehungstypen überhaupt keine Meta-Attribute vorgesehen sind. Hingegen wurden für zwei von neun Meta-Entitätstypen insgesamt zwei Meta-Attribute vorgesehen, die beide zwingend sind. Es handelt sich hierbei um die folgenden beiden Meta-Attribute, die voll qualifiziert werden:

– „AggregationalMetaRelationship.**AggregatesTo**“ und

– „MetaObjectReference.**ReferencedMetaObject**“.

Damit müssen Instanzen vom Meta-Entitätstyp „AggregationalMeta-Relationship“ sowie von „MetaObjectReference“ in einem EIA/CDIF-Austausch Werte in den zwingend vorgeschriebenen Meta-Attributen aufweisen.

---

797 Zwingend vorgeschriebene Meta-Beziehungstypen verfügen über einen Wert >=1 in mindestens einer der minimalen Kardinalitäten. Aufgrund der gegenüberliegenden Anschrift der Kardinalitäten im graphischen Modell bedeutet dies, daß ein Wert >=1 im Meta-Meta-Attribut „Min*Source*Card“ zur Folge hat, daß Instanzen des *Ziel*-Meta-Entitätstyps an Instanzen dieses Meta-Beziehungstyps teilnehmen müssen. Entsprechend müssen bei einem Wert >=1 im Meta-Meta-Attribut „Min*Dest*Card“ Instanzen des *Quell*-Meta-Entitätstyps an Instanzen des Meta-Beziehungstyps enthalten sein.

798 In dieser Tabelle werden referenzierte Meta-Objekte nicht berücksichtigt.

In diesem GegenstandsBereich „M2Level“ wird in drei von vier Fällen auch von der Möglichkeit Gebrauch gemacht, Meta-Beziehungstypen so zu definieren, daß Instanzen von damit in Beziehung gesetzten Meta-Entitätstypen in jedem Fall an Instanzen des Meta-Beziehungstyps teilhaben müssen. Es sind dies die drei folgenden Meta-Beziehungstypen:

- „0:N **SetOfExclusiveMetaRelationships**.*Constrains*.MetaRelationship-Reference **2**:N“: Instanzen vom Typ „SetOfExclusiveMetaRelationships“ müssen mindestens in zwei Instanzen dieses Meta-Beziehungstyps enthalten sein,
- „0:N **SetOfValidComponents**.*Contains*.ComponentObjectReference **1**:N“: Instanzen vom Typ „SetOfValidComponents“ müssen zumindest an einer Instanz dieses Meta-Beziehungstyps teilhaben, und
- „0:1 **SetOfValidComponents**.*ForBuildingStructureOf*.DefinitionObject-Reference **1**:1“: Instanzen vom Typ „SetOfValidComponents“ müssen genau an einer Instanz dieses Meta-Beziehungstyps partizipieren.

Zusammenfassend müssen also Instanzen der beiden Meta-Entitätstypen „SetOfExclusiveMetaRelationships“ und „SetOfValidComponents“ gleichzeitig auch an Instanzen von Meta-Beziehungstypen zwingend teilhaben.

Die Meta-Objekte, die Blätter im Baum dieses Meta-Modells darstellen, sind defacto gleichmäßig auf Meta-Entitätstypen und Meta-Beziehungstypen im Verhältnis von fünf zu vier aufgeteilt.

4.4.3.2 Aufgefundene Fehler

Die Namen „M2Level“ („SubjectArea“) und „M2LevelObject“ („MetaEntity“) sind vom EIA/CDIF-Datentyp „Identifier“, allerdings verletzen sie die zusätzlich definierten Regeln für Meta-Objekt-Namen, die eigentlich keine Ziffern aufweisen dürfen.[799] Da im GegenstandsBereich „PLAC“ trotz dieser Regel[800] sinnvollerweise für die Bezeichnung der einzelnen Elemente der Transformationsmatrix

---

799 Vgl. in diesem Zusammenhang auch Abschnitt 4.1.3.3,’PLAC – EIA/CDIF-Meta-Modell für „Presentation Location and Connectivity“’ auf Seite 257ff.

800 Es kann daher angenommen werden, daß diese Einschränkung für Meta-Objekt-Namen, nämlich keine Ziffern aufweisen zu dürfen, bei einer Überarbeitung der Regeln aufgehoben wird.

die Namen den Index in Form von Zahlen aufweisen, wurde in dieser Arbeit aus Gründen der Prägnanz die Verletzung dieser Regel in Kauf genommen.

4.4.3.3 Generalisierungshierarchie des Meta-Modells

Die folgende Abbildung 4-13 stellt die Generalisierungshierarchie dieses Meta-Modells dar, bei dem aufgrund der Nutzung der Meta-Objekte aus dem fundamentalen GegenstandsBereich „Foundation“ jeweils ein Teilbaum für die Meta-Entitätstypen und einer für die Meta-Beziehungstypen gebildet wird. Die maximale Höhe des Generalisierungsbaumes ist sechs, wobei der Zweig für die Meta-Beziehungstypen eine Höhe von drei Ebenen ausmacht. Somit stellt sich der Teilbaum für Meta-Entitätstypen als wesentlich spezialisierter dar als der für Meta-Beziehungstypen.

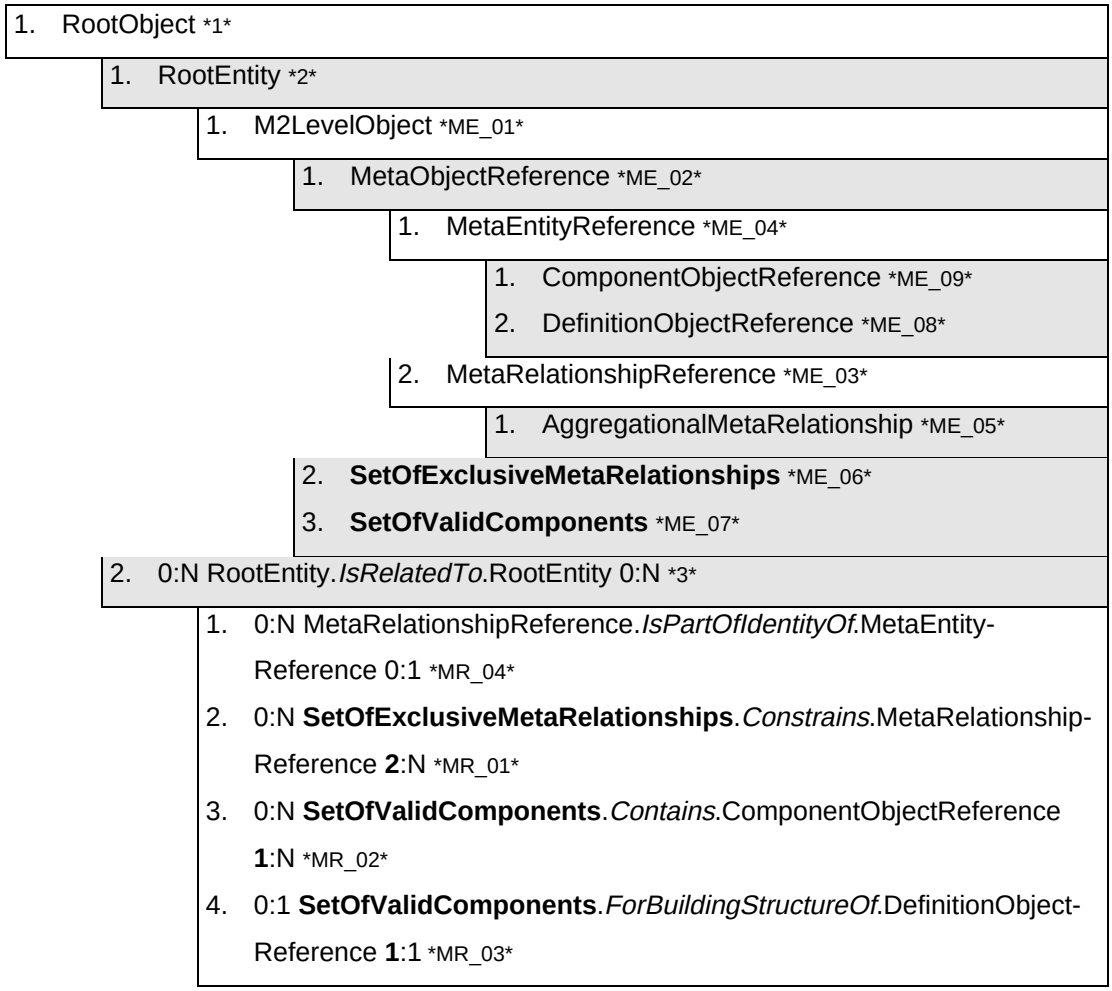


Abbildung 4-13: Generalisierungshierarchie des Meta-Modells „M2Level“

4.4.3.4 Summarische Darstellung der AttribuierbarenMetaObjekte

Die summarische Darstellung der Meta-Entitätstypdefinitionen entspricht dem Kapitel 4.4, „MetaEntity Summary“ eines jeden EIA/CDIF-Standards, die der Meta-Beziehungstypdefinitionen dem Kapitel 4.5, „MetaRelationship Summary“. Hierbei erfolgt die Reihung in der alphabetisch sortierten Reihenfolge der qualifizierten Namen der AttribuierbarenMetaObjekte. In dieser Darstellung wird summarisch gezeigt, welche Meta-Attribute über die Generalisierungshierarchie ererbt und welche, wenn überhaupt, lokal definiert wurden. Ererbte Meta-Attribute werden hierbei – wie in den EIA/CDIF-Standards üblich – kursiv dargestellt, lokale mit normaler Schriftauszeichnung. Zwingend vorgeschriebene Meta-Attribute werden zusätzlich fett hervorgehoben.

Im Unterschied zu den EIA/CDIF-Standards ist in der Aufstellung auch angegeben, von welchen übergeordneten Typen die ererbten Meta-Attribute stammen und welche Werte[801] für das entsprechende Meta-Meta-Attribut „CDIFMeta-Identifier“ vorgesehen sind. Des weiteren werden die EIA/CDIF-Datentypen für die lokal definierten Meta-Attribute angeführt, im Falle von zwingend vorgeschriebene Meta-Attributen werden diese fett gesetzt, um darauf aufmerksam zu machen.

Sämtliche Meta-Beziehungstypen werden gemeinsam mit den für sie definierten Kardinalitäten dargestellt. Sofern in einem Meta-Beziehungstyp Meta-Entitätstypen zwingend[802] teilhaben müssen, werden die betroffenen Meta-Entitätstypen in allen Darstellungen fett dargestellt.

Somit erlauben es die folgenden Tabellen, auf einen Blick die wesentlichsten Definitionen zu erkennen.

4.4.3.4.1 Darstellung der Meta-Entitätstypen

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „AggregationalMetaRelationship“ dar und wird ausdrücklich im Gegenstands-

801 Diese Surrogatwerte werden in kleiner Schriftgröße angegeben, wobei die Werte entsprechend [CDIF94e], „ENCODING.1“, in Sternen eingeschlossen sind.

802 Der Minimalwert der gegenüberliegenden Kardinalität ist in einem solchen Fall größer Null.

Bereich „M2L“ benutzt. Es wird ein einziges lokales MetaAttribut definiert, das zugleich zwingend vorgeschrieben ist.

AggregationalMetaRelationship *ME_05*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ReferencedMetaObject *MA_01*	***zwingend***	*von: MetaObjectReference* *ME_02*
AggregatesTo *MA_02* – Enumerated {Source, Destination}	**zwingend**	[lokal]

Tabelle 4-104: Summarische Darstellung des Meta-Entitätstyps „AggregationalMeta-Relationship“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „ComponentObjectReference“ dar und wird ausdrücklich im Gegenstands-Bereich „M2L“ benutzt. Er verfügt über kein lokales MetaAttribut.

ComponentObjectReference *ME_09*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ReferencedMetaObject *MA_01*	***zwingend***	*von: MetaObjectReference* *ME_02*

Tabelle 4-105: Summarische Darstellung des Meta-Entitätstyps „ComponentObject-Reference“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „DefinitionObjectReference“ dar und wird ausdrücklich im GegenstandsBereich „M2L“ benutzt. Er verfügt über kein lokales MetaAttribut.

DefinitionObjectReference *ME_08*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ReferencedMetaObject *MA_01*	***zwingend***	*von: MetaObjectReference* *ME_02*

Tabelle 4-106: Summarische Darstellung des Meta-Entitätstyps „DefinitionObject-Reference“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „M2LevelObject“ dar und wird ausdrücklich im GegenstandsBereich „M2L“ benutzt. Er verfügt über kein lokales MetaAttribut.

M2LevelObject *ME_01*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

Tabelle 4-107: Summarische Darstellung des Meta-Entitätstyps „M2LevelObject“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „MetaEntityReference“ dar und wird ausdrücklich im GegenstandsBereich „M2L“ benutzt. Er verfügt über kein lokales MetaAttribut.

MetaEntityReference *ME_04*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	

TimeUpdated *9*	*optional*	
ReferencedMetaObject *MA_01*	***zwingend***	*von: MetaObjectReference* *ME_02*

Tabelle 4-108: Summarische Darstellung des Meta-Entitätstyps „MetaEntityReference“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „MetaObjectReference“ dar und wird ausdrücklich im GegenstandsBereich „M2L“ benutzt. Es wird ein einziges lokales MetaAttribut definiert, das zugleich zwingend vorgeschrieben ist.

MetaObjectReference *ME_02*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ReferencedMetaObject *MA_01* – Identifier	**zwingend**	[lokal]

Tabelle 4-109: Summarische Darstellung des Meta-Entitätstyps „MetaObject-Reference“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „MetaRelationshipReference“ dar und wird ausdrücklich im GegenstandsBereich „M2L“ benutzt. Er verfügt über kein lokales MetaAttribut.

MetaRelationshipReference *ME_03*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	
ReferencedMetaObject *MA_01*	***zwingend***	*von: MetaObjectReference* *ME_02*

Tabelle 4-110: Summarische Darstellung des Meta-Entitätstyps „MetaRelationship-Reference“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SetOfExclusiveMetaRelationships“ dar und wird ausdrücklich im GegenstandsBereich „M2L“ benutzt. Er verfügt über kein lokales MetaAttribut.

SetOfExclusiveMetaRelationships *ME_06*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

Tabelle 4-111: Summarische Darstellung des Meta-Entitätstyps „SetOfExclusiveMeta-Relationships“

Die folgende Tabelle stellt eine summarische Darstellung des Meta-Entitätstyps „SetOfValidComponents“ dar und wird ausdrücklich im GegenstandsBereich „M2L“ benutzt. Er verfügt über kein lokales MetaAttribut.

SetOfValidComponents *ME_07*		
CDIFIdentifier *5*	***zwingend***	*von: RootObject* *1*
DateCreated *6*	*optional*	
DateUpdated *7*	*optional*	
TimeCreated *8*	*optional*	
TimeUpdated *9*	*optional*	

Tabelle 4-112: Summarische Darstellung des Meta-Entitätstyps „SetOfValid-Components“

4.4.3.4.2 Darstellung der Meta-Beziehungstypen

Da sämtliche Meta-Beziehungstypen dieses GegenstandsBereiches über keine eigenständigen Meta-Attribute aufweisen und sie gleichzeitig alle direkte Subtypen des fundamentalen Meta-Beziehungstyps „0:N RootEntity.*IsRelatedTo*.-RootEntity 0:N“ sind, erscheint eine summarische Darstellung über die Vererbung der Meta-Attribute an dieser Stelle nicht für notwendig.

Stattdessen wird zusammenfassend in der folgenden Tabelle 4-113 eine Aufstellung der definierten Meta-Beziehungstypen gegeben, gemeinsam mit den entsprechenden Kardinalitäten[803].

0:N	MetaRelationshipReference.*IsPartOfIdentityOf*.MetaEntityReference *MR_04*	0:1
0:N	**SetOfExclusiveMetaRelationships**.*Constrains*.MetaRelationshipReference *MR_01*	**2**:N
0:N	**SetOfValidComponents**.*Contains*.ComponentObjectReference *MR_02*	**1**:N
0:1	**SetOfValidComponents**.*ForBuildingStructureOf*.DefinitionObjectReference *MR_03*	**1**:1

Tabelle 4-113: Summarische Darstellung der Meta-Beziehungstypen, gemeinsam mit den entsprechenden Kardinalitäten, GegenstandsBereich „M2Level“

803 Wenn der Quell- (MinSourceCard > 0) oder/und der Ziel-Meta-Entitätstyp (MinDestCard > 0) am Beziehungstyp partizipieren müssen, dann wird diese Tatsache hervorgehoben: der Meta-Entitätstyp, der zwingenderweise in Instanzen des entsprechenden Meta-Beziehungstyps enthalten sein muß, wird im vollqualifizierten Namen in der Tabelle fett dargestellt.

5 Zusammenfassung und Ausblick

Zunächst wird in diesem Abschnitt eine kurze Zusammenfassung über den Diskurs über das EIA/CDIF-Meta-Meta-Modell und die EIA/CDIF-Meta-Modelle gegeben und noch einmal die Genese des Meta-Modells „M2Level“ dargestellt, die ausschließlich durch die aufgefundenen Einschränkungen motiviert ist.

Daran anschließend wird kurz auf weitere, mögliche Entwicklungen im Rahmen dieser Diskussionen eingegangen.

5.1 Zusammenfassung

Die im Rahmen dieser Arbeit erfolgte Auseinandersetzung mit den EIA/CDIF-Standards konzentrierte sich zunächst auf die systematische Darstellung des EIA/CDIF-Meta-Meta-Modells. Die Modellierungssprache, die damit definiert wird, entspricht der konzeptionellen Datenmodellierung mit der erweiterten Entity-Relationship-Attribute-Methode, etwa nach [BaCeNa92] oder [ElmNav89]. Nachdem diese beiden Bücher seit Jahren im Bereich der konzeptionellen Datenmodellierung beziehungsweise im Design von Datenbanken als weltweite Standardliteratur angesehen werden kann, ist es wahrscheinlich, daß damit Vertraute keinerlei Probleme mit dem Erarbeiten des reflexiven EIA/CDIF-Meta-Meta-Modells selbst haben. Desgleichen wird erwartet, daß die damit erstellten EIA/CDIF konformen Meta-Modelle ebenso leicht oder schwer erarbeitet werden können.

Aufgrund der OMG IDL-Definitionen für das EIA/CDIF-Meta-Meta-Modell können Meta-Modelle verteilt werden, wobei das Meta-Meta-Modell selbst instanziiert wird und für die Typaufsuchung zur Verfügung steht. Für die Abbildung des EIA/CDIF-Meta-Meta-Modells in relationale Datenbanken wurde eine entsprechende Spezifikation in Form von SQL-Anweisungen festgelegt.[804] Desgleichen ist eine Abbildung in eine objektorientierte Programmiersprache, in dieser Arbeit

---

804 Eine Implementierung aufgrund dieser Spezifikationen erfolgt im Anhang im Abschnitt 6.2.1, „Implementierung des EIA/CDIF-Meta-Meta-Modells in ORACLE“ auf Seite 424ff.

ist dies die interpretierte Sprache Object Rexx, mit relativ einfachen Mitteln möglich.[805]

Die zusammenfassende Diskussion des EIA/CDIF-Meta-Meta-Modells charakterisiert seine grundlegenden Eigenschaften und zeigt auch Einschränkungen auf, die durch die beabsichtigte Minimalität des EIA/CDIF-Meta-Meta-Modells gegeben sind. Als größte Einschränkung wird der Mangel angesehen, einander ausschließende Meta-Beziehungstypen explizit innerhalb des Meta-Meta-Modells ausdrücken zu können. Sämtliche aufgefundenen Einschränkungen können mit Hilfe des für diese Arbeit erstellten Meta-Modells „M2Level“ behoben werden.[806]

Ein weiterer Hauptteil der Arbeit stellt in einer systematischen und zusammenhängenden Form sämtliche per 1. Jänner 1998 standardisierten EIA/CDIF-Meta-Modelle vor. Es werden dafür unterschiedliche Auswertungen vorgenommen, die das Ziel verfolgen, einerseits über strukturelle Eigenschaften Einblick zu geben, andererseits die wichtigsten Informationen kompakt und zusammenhängend so darzustellen, daß sie gleichzeitig auch Referenzcharakter erhalten.

Aufgrund dieser Definitionen wird in weiterer Folge erstmals, das „Integrierte EIA/CDIF-Meta-Modell“ erstellt und so dokumentiert, daß dieser Teil jederzeit selbst als Referenzbuch[807] über die standardisierten EIA/CDIF-Meta-Objekte dienen kann. Im Zuge der Überprüfungen der Definitionen wurden drei schwerwiegende Surrogatfehler gefunden, die dazu führen könnten, daß der Austausch von Modelldaten unter bestimmten Umständen nicht erfolgreich für den Gegenstandsbereich „Datenmodellierung“ erfolgen kann.[808]

---

805 Ein Implementierungsbeispiel aufgrund der Object Rexx-Spezifikationen findet sich im Anhang im Abschnitt 6.2.2, „Implementierung des EIA/CDIF-Meta-Meta-Modells in Object Rexx“ auf Seite 449ff.

806 Diese Einschränkungen könnten auch dadurch behoben werden, indem das EIA/CDIF-Meta-Meta-Modell selbst komplexer strukturiert wird. Nachdem eine derartige Zunahme in der Komplexität auch für die Erstellung des ISO/CDIF-Standards als nicht annehmbar erschien, ist es nicht zu erwarten, daß dieser Fall eintritt.

807 Es handelt sich um insgesamt 123 Tabellen, die auch die Vererbungsstruktur und die ererbten Meta-Attribute für jedes AttribuierbareMetaObjekt darstellen, wobei für die Meta-Attribute die Datentypen angegeben sind, sowie die Wertemenge im Fall des Datentyps „aufzählbare Werte“.

808 Diese aufgefundenen Fehler werden für den entsprechenden ISO/CDIF Standard gleichermaßen behoben, und werden im Rahmen von EIA/CDIF durch Übernahme der ISO/CDIF-Standards ausgemerzt werden.

Die daran anschließende zusammenfassende Diskussion nimmt zunächst eine Auswertung über alle SammelbarenMetaObjekte vor. In weiterer Folge werden übersichtsmäßig alle AttribuierbarenMetaObjekte gezeigt, die in mehr als einem Meta-Modell benutzt werden, sowie eine Liste von AttribuierbarenMetaObjekten mit Mehrfachvererbung dargestellt. Über weitere derartige Auswertungen wird auch eine Analyse der Verteilung von AttribuierbarenMetaObjekten mit beziehungsweise ohne Meta-Attributen angegeben.

In weiterer Folge erfolgte eine pägnante kritische Diskussion der strukturellen Eigenschaften, in der auf der Ebene der Meta-Modelle auf Einschränkungen in bezug auf den allgemeinen Strukturierungsmechanismus eingegangen wird. Es ist damit derzeit nicht möglich, ausdrücklich festzulegen, welches DefinitionsObjekt aus welchen KomponentenObjekten zusammengesetzt sein darf. Zudem werden die beiden strukturellen Einschränkungen angeführt, die bereits während der Diskussion des EIA/CDIF-Meta-Meta-Modells diskutiert wurden: der Mangel, einander ausschließende Meta-Beziehungstypen ausdrücklich dokumentieren zu können, sowie Meta-Beziehungstypen selbst zu klassifizieren.

Aufgrund der aufgefundenen Einschränkungen wird in einem eigenen Abschnitt ein neues Meta-Modells „M2Level“ konzipiert und zur Diskussion gestellt. Damit wird versucht, die aufgefundenen und diskutierten Mängel so zu beheben, daß es nicht notwendig wird, die Struktur des EIA/CDIF-Meta-Meta-Modells selbst zu ändern. Zudem wird damit auch die Möglichkeit geschaffen, identitätsstiftende und aggregierende Meta-Beziehungstypen ausdrücklich als solche zu kennzeichnen.

5.2 Ausblick

Die in aufwendiger Arbeit in einem Zeitraum von mehr als zehn Jahren verfaßten Standards von EIA/CDIF, beginnend mit der systematischen Entwicklung eines Meta-Meta-Modells, das die Grundlage für die Erstellung von Meta-Modellen bildet, bis hin zu Standards zur Verteilung von Meta-Modellen und Modellen mit Hilfe von CORBA IDL, werden im Rahmen der ISO/IETC adaptiert und zu einem internationalen Standard erhoben. Parallel dazu werden die Arbeitsergebnisse von EIA/CDIF im Rahmen der OMG-Arbeiten für die Definition von objektorientierten Analyse- und Modellierungsmethoden von Informations-

systemen herangezogen, wenn es darum geht, erstellte Modelldaten austauschfähig zu machen. Dieser Abschnitt schließt mit einem Boeing-Entwurf für die Modellierung des GegenstandsBereiches „Geschäftsprozeßmodellierung“, der gleichzeitig einen Ausgangspunkt für die Diskussion von entsprechenden Modellierungsmethoden bilden kann.

5.2.1 ISO/CDIF

EIA/CDIF dient als Ausgangsbasis für den ISO/CDIF-Standard, der im November 1998 endgültig verabschiedet werden soll.

Im ISO/CDIF-Meta-Meta-Modell ergeben sich im Vergleich zu EIA/CDIF voraussichtlich zwei Änderungen:

- ein Meta-Meta-Attribut „IsAbstract“ wird dem Meta-Meta-Entitätstyp „AttributabelMetaObject“ hinzugefügt und gibt darüber Auskunft, ob von einem Meta-Entitätstyp oder Meta-Beziehungstyp direkt Instanzen gebildet werden dürfen;
- ein neuer Meta-Meta-Beziehungstyp namens „0:N **CollectableMetaObject**.*IsDefinedIn*.SubjectArea 1:1“ wird eingeführt, mit dessen Hilfe eindeutig angegeben werden kann, in welchem Meta-Modell ein SammelbaresMetaObjekt (erstmalig) definiert wird.

Im ISO/CDIF-Meta-Modell „Common“ werden im Vergleich zu der EIA/CDIF-Version von „Common“ einerseits bisher nicht in Anspruch genommene Meta-Entitätstypen[809] entfernt und andererseits der Allgemeine Strukturierungsmechanismus eingeführt, der zum Teil mit Meta-Entitätstypen und Meta-Beziehungstypen erweitert wird, die in [Ernst98], Seiten 82 bis 126, diskutiert werden. Die angesprochenen Erweiterungen zum Allgemeinen Strukturierungsmechanismus sind:

- der Meta-Entitätsyp „Model“, der gemeinsam mit dem Meta-Beziehungstyp „0:1 Model.*HasRoot*.DefinitionObject 0:1“ die Wurzel eines beliebigen Modells unabhängig von einem bestimmten GegenstandsBereich festlegt,

809 Es handelt sich um die EIA/CDIF-Meta-Entitätstypen „DataObject“ und „ProcessObject“.

- der Meta-Beziehungstyp „0:1 DefinitionObject.*ContainsAsFormal*.ComponentObject 0:N“, der es erlaubt, formale KomponentenObjekte festzulegen,
- der Meta-Beziehungstyp „0:N ComponentObject.*IsActualFor*.ComponentObject 0:N“, der es erlaubt, aktuelle KomponentenObjekte in Beziehung zu formalen zu setzen.

5.2.2 EIA/CDIF

Es ist zu erwarten, daß in weiterer Folge das EIA/CDIF-Komitee, das an der Weiterentwicklung in Form von ISO/CDIF mitgearbeitet hat, diese Standards als Update übernimmt.[810]

Es werden folgende im Rahmen von EIA/CDIF erstellten Entwürfe für weitere Meta-Modelle weiterentwickelt werden:

- „Business Process Modeling Subject Area“[811] (EIA/CDIF-Abkürzung: „BPM“) in [CDIF96e] frei verfügbar,
- „Computer Aided Control Systems Design Subject Area“ (EIA/CDIF-Abkürzung: „CACSD“) in [CDIF98a] frei verfügbar,
- „Data Definition Subject Area“ (EIA/CDIF-Abkürzung: „DDEF“) in [CDIF96h] frei verfügbar,
- „Object-oriented Analysis and Design Core Subject Area“[812] (EIA/CDIF-Abkürzung: „OOAD“) in [CDIF96f] frei verfügbar,
- „Project Management Planning And Scheduling Subject Area“ (EIA/CDIF-Abkürzung: „PMPS“) in [CDIF96g] frei verfügbar, und

---

810 Der Amerikaner Woody Pidcock von der Firma Boeing ist mit Stand Mitte 1998 der „Conveyor“ (Vorsitzende) von ISO/IEC JTC 1/SC 7 und sämtlicher seiner Arbeitsgruppen, zugleich ist er Präsident des EIA/CDIF-Komitees. Zudem ist er aktiv an der Ausarbeitung und Evaluierung von OMG’s „SMIF“-RFP beteiligt.

811 Für diesen für die Wirtschaftsinformatik wichtigen GegenstandsBereich wird am Ende dieses Kapitels eine summarische Übersicht der erfolgten Auswertungen gegeben.

812 Die Arbeiten an diesem Meta-Modell wurden vorübergehend eingestellt, da zunächst abgewartet werden sollte, was OMG’s UML abdeckt. Es ist derzeit (1998-06-02) noch nicht klar, ob dieser weit gediehene Entwurf weiter verfolgt wird.

– „State/Event Model Subject Area“ (EIA/CDIF-Abkürzung: „STEV“) in [CDIF96a] frei verfügbar.

5.2.3 OMG’s SMIF

Von Sommer 1998 an bis zum Frühjahr 1999 werden die aufgrund von OMG’s „Request For Proposal“ (abgekürzt: „RFP“) für ein „Stream-based Model Interchange Format“[813] (abgekürzt: „SMIF“) bis Anfang Juli 1998 eingereichten Vorschläge bearbeitet werden und zu einem OMG-Standard für den Austausch von Modelldaten führen.

Eine wesentliche Bedingung dafür ist die Unterstützung von CDIF und STEP[814], sodaß die entsprechenden, eingereichten Lösungsvorschläge in jedem Fall CDIF unterstützen müssen.

Es ist zu erwarten, daß ein Konsortium von unterschiedlichen Firmen, die bereits jetzt EIA/CDIF und künftig auch ISO/CDIF unterstützen, einen rein CDIF-basierenden Vorschlag für den Modelldatenaustausch einreichen. Hierbei wird eine Untermenge des OMG Meta-Meta-Modells „Meta Object Facility“[815] (abgekürzt: „MOF“) benutzt, die direkt mit den Konzepten des EIA/CDIF- beziehungsweise ISO/CDIF-Meta-Meta-Modells korrespondieren.[816] Somit werden dadurch sämtliche standardisierten und in Entwurf befindlichen Meta-Modelle für den Modelldatenaustausch zugänglich gemacht. In jedem Fall ist zu erwarten, daß im Zusammenhang mit der Einreichung eines auf CDIF-basierenden

---

813 Vgl. hierzu [OMG97] vom Dezember 1997.

814 Mit Mitte 1998 ist eine Abbildung von EIA/CDIF-Meta-Modellen auf STEP – ähnlich jener auf ECMA/PCTE (vgl. Abschnitt 1.3.2.1, „EIA/CDIF, ECMA-149 und ECMA-270“ auf Seite 17ff oben) – vom „British Standardisation Institute“ in Arbeit und soll über ein „Fast-Track“-Verfahren zu einem ISO/CDIF-Standard werden (vgl. hierzu auch die Ausführungen in Abschnitt 1.3.2.3, „EIA/CDIF und STEP/EXPRESS (ISO/IEC TC 184/SC 4)“ auf Seite 21ff).

815 Vgl. [MOF97a], [MOF97b] und Abschnitt 1.3.2.4, „EIA/CDIF und OMG (UML und SMIF)“ auf Seite 22ff oben.

816 Dazu werden die „statischen“ Konzepte aus MOF herangezogen. Die „dynamischen“ Teile von MOF könnten unter anderem auch durch die Erweiterung des in dieser Arbeit vorgestellten EIA/CDIF-konformen Meta-Modells „M2Level“ (vgl. Abschnitt 4.4, „Definition eines EIA/CDIF-konformen Meta-Modells „M2Level“‘ auf Seite 356ff) abgebildet werden, soferne dies für sinnvoll angesehen wird.

SMIF-Vorschlages gezeigt werden wird, wie UML-Modelle mit Hilfe eines CDIF-Meta-Modells für UML ohne Informationsverlust getauscht werden können.[817]

Als „Medium“ für den Modelldatenaustausch können Ströme von Zeichen dienen, die EIA/CDIF-konform mit der SYNTAX.1-Syntax und der ENCODING.1-Kodierung erstellt werden. Mitte 1998 wurde von EIA/CDIF ein erster Entwurf für ein XML-DTD[818] vorgestellt, mit dessen Hilfe CDIF-Meta-Modell- und CDIF-Modelldaten austauschbar sind.[819] EIA/CDIF verfügt zudem über einen weiteren, OMG CORBA-konformen Weg, Modelldaten zu tauschen, nämlich in Form von OMG IDL-definierten Schnittstellen, die in verteilten Umgebungen für den Modelldatenaustausch benutzt werden können.

5.2.4 Entwurf eines EIA/CDIF-Meta-Modells für die „Geschäftsprozeßmodellierung“

Unter [CDIF96e] steht ein über das Internet frei beziehbarer Entwurf eines Meta-Modells für den GegenstandsBereich „Geschäftsprozeßmodellierung“ zur Verfügung. Es handelt sich hierbei um einen Entwurf, der von der Firma Boeing stammt, die dieses Meta-Modell für die eigene Geschäftsprozeßmodellierung einsetzt.

Der Leser wird auf diesen Entwurf verwiesen, der wie ein typischer EIA/CDIF-Standard aufgebaut ist. Nach erfolgreicher Erarbeitung der in dieser Arbeit vermittelten Inhalte, sollte es problemlos lesbar und verständlich sein.[820] Die folgende Auswertung und Übersicht mögen eine Hilfestellung beim Durcharbeiten von [CDIF96e] geben. Das Meta-Modell „BPM“ definiert insgesamt 110 SammelbareMetaObjekte, die sich entsprechend der Tabelle 5-1 auf die unterschiedlichen Meta-Meta-Entitätstypen verteilen.

---

817 Davon hängt auch die weitere Entwicklung des EIA/CDIF-Entwurfes „Object-oriented Analysis and Design Core Subject Area“ ab, vgl. hierzu [CDIF96f]. Findet sich kein zufriedenstellender Entwurf für ein EIA/CDIF-Meta-Modell für UML, so ist zu erwarten, daß die Arbeit an CDIF-OOAD weitergeführt wird.

818 „XML“ stellt das Akronym für „Extensible Markup Language“ und „DTD“ das Akronym für „Document Type Definition“ dar. Vgl. den Entwurf in [CDIF98b].

819 Nachdem XML breit in der Industrie angenommen wurde, ist es in Zukunft möglich, mit entsprechenden XML-Parsern und XML-Anwendungen Dokumenteninstanzen, die entsprechend der CDIF XML-DTD strukturiert sind, auszuwerten und weiterzuverarbeiten.

820 Das Verständnis davon und die Auseinandersetzung damit sollte nach dem Studium dieser Arbeit ohne fremde Hilfe möglich sein. Ist dies gelungen, ist ein wesentliches Ziel und Anliegen des Autors damit erfüllt worden.

Anzahl Instanzen vom Typ AMO	1
Anzahl Instanzen vom Typ ME	37
Anzahl Instanzen vom Typ MR	19
Anzahl Instanzen vom Typ MA	53
Anzahl (Summe) Instanzen vom Typ CMO	110

Tabelle 5-1: Verteilung der sammelbaren Objekte des Meta-Modellentwurfs „Geschäftsprozeßmodellierung“

Die folgende Abbildung 5-1 stellt die Generalisierungshierarchie dieses Meta-Modells dar. Die maximale Höhe des Generalisierungsbaumes ist acht, wobei der Zweig für die Meta-Beziehungstypen eine Höhe von vier Ebenen ausmacht. Somit stellt sich der Teilbaum für Meta-Entitätstypen als wesentlich spezialisierter dar als der für Meta-Beziehungstypen.[821]

1. RootObject *1*
    1. RootEntity *Missing_CDIF_7*
        1. AbstractionLevel *12*
        2. **AlternateName** *14*
        3. DataModelObject *Missing_CDIF_8*
        4. InheritableDataModelObject *Missing_CDIF_9*
            1. *ProductDefinition* *BPM1001_Duplicate_3*
                1. *Hardware* *BPM1005*
                2. *Information* *BPM1013*
                3. *Service* *BPM8002_Duplicate_4*
            2. ***Relationship*** *1040*
                1. *BusinessRelationship* *BPM8002*
        5. SemanticInformationObject *4*
            1. ComponentObject *8000*
                1. Attribute *17*
                2. BusinessProcess *BPM17*
                3. ComputingSystem *BPM1016*
                4. **EquivalenceSet** *1113*
                5. FlowProducerConsumer *232*
                    1. Flow *227*

821 Die angeführten Surrogatwerte zeigen an, ob bei der Analyse dieses Entwurfes Fehler gefunden wurden und wenn ja, welcher Natur sie sind.

2. *FlowPort* *6015*
    1. *FlowInputPort* *6001*
        1. *SupportPort* *6010*
    2. *FlowOutputPort* *6000*
6. Port *1129*
    1. *FlowPort* *6015*
        1. *FlowInputPort* *6001*
            1. *SupportPort* *6010*
        2. *FlowOutputPort* *6000*
7. Product *BPM1001*
8. **ReferencedElement** *8020*

2. DefinitionObject *Missing_CDIF_2*
    1. BusinessProcessDefinition *BPM18*
    2. FlowDefinition *221*
    3. *ProductDefinition* *BPM1001_Duplicate_3*
        1. *Hardware* *BPM1005*
        2. *Information* *BPM1013*
        3. *Service* *BPM8002_Duplicate_4*
    4. ***Relationship*** *1040*
        1. *usinessRelationship* *BPM8002*
    5. **Role** *1042*
    6. **RolePlayer** *1048*
3. **Derivation** *26*
4. KeyElementDescription *BPM8002_Duplicate_2*
5. RoleConstraint *1045*
6. SubtypeSet *1070*

6. **TextualConstraint** *51*
7. ToolUser *56*

2. 0:N RootEntity.*IsRelatedTo*.RootEntity 0:N *Missing_CDIF_6*
    1. 0:N ComponentObject.*References*.DefinitionObject 0:1 *8022*
    2. **1**:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N *1705*
    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*
    7. **2**:N **Role**.*BelongsTo*.**Relationship 1**:1 *1724*

8. **1**:N RolePlayer.*Plays*.**Role** 0:1 *1730*
9. 0:N RootEntity.*CreatedBy*.ToolUser 0:1 *68*
10. **1**:1 RootEntity.*Has*.**AlternateName** 0:N *69*
11. 0:N RootEntity.*UpdatedBy*.ToolUser 0:1 *BPM70*
12. 0:N RootEntity.*Uses*.AlternateName 0:1 *71*
13. 0:N SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N *72*
14. **1**:N SemanticInformationObject.*ProducedBy*.**Derivation** 0:N *73*
15. **1**:N SemanticInformationObject.*UsedIn*.**Derivation** 0:N *74*
16. 0:N **TextualConstraint**.*IsConstraintOn*.SemanticInformationObject **1**:N *80*

Abbildung 5-1: Generalisierungshierarchie des Meta-Modellentwurfs „Geschäfts-prozeßmodellierung“[822]

Es fällt auf, daß dieses Meta-Modell sehr stark auf die Definitionen des GegenstandsBereiches „Datenflußmodellierung“[823] zurückgreift und auch Definitionen aus dem GegenstandsBereich „Datenmodellierung“[824] wiederverwendet.

822 Entsprechend den EIA/CDIF-Konvention weisen kursiv gesetzte Zeilen auf Typen mit Mehrfachvererbung hin.

823 Vgl. Abschnitt 4.1.3.1, „DFM – EIA/CDIF-Meta-Modell für die Datenflußmodellierung“ auf Seite 224ff oben.

824 Vgl. Abschnitt 4.1.3.2, „DMOD – EIA/CDIF-Meta-Modell für die Datenmodellierung“ auf Seite 241ff oben.

6 Anhang

6.1 Querverweistabellen für das Integrierte EIA/CDIF-Meta-Modell

Die in diesem Abschnitt dargestellten Meta-Objekte sind ausschließlich dem Integrierten EIA/CDIF-Meta-Modell entnommen und beziehen sich im Rahmen dieser Arbeit gleichermaßen auf die bis Anfang 1998 standardisierten EIA/CDIF-Meta-Modelle.

Eine strukturelle Darstellung aller standardisierter AttribuierbarenMetaObjekte des Integrierten EIA/CDIF-Meta-Modells unter Berücksichtigung der Vererbung entlang der Generalisierungshierarchie findet sich für die Meta-Entitätstypen im Abschnitt 4.2.3.4.1.2, ‘Summarische Definition der direkten Instanzen vom Typ „MetaEntity“’ auf Seite 281ff und für die Meta-Beziehungstypen im Abschnitt 4.2.3.4.1.3“, ‘Summarische Definition der direkten Instanzen vom Typ „MetaRelationship“’ auf Seite 316ff. Zusammenfassende Darstellungen, Auswertungen und Diskussionen über die Strukturen der Meta-Modelle finden sich zum einen bei der Darstellung der entsprechenden Meta-Modelle und über alle Meta-Modelle hinweg im Abschnitt 4.3, „Zusammenfassende Diskussion der standardisierten EIA/CDIF-Meta-Modelle“ auf Seite 340ff.

6.1.1 Querverweistabelle der standardisierten Meta-Objekte, sortiert nach dem Meta-Meta-Attribut „CDIFMetaIdentifier“

Die folgende Tabelle 6-1 listet sämtliche standardisierten Meta-Objekte des Integrierten EIA/CDIF-Meta-Modells, sortiert nach dem Meta-Meta-Attribut „CDIFMetaIdentifier“. In der ersten Spalte findet sich der Surrogatwert, in der zweiten Spalte wird der Name des Meta-Objekts gemeinsam mit der Abkürzung seines Metatyps angeführt. Für Meta-Attribute wird in der dritten Spalte in Klammern eingeschlossen das AttribuierbareMetaObjekt gezeigt, für das es definiert wurde. Für Meta-Beziehungstypen wird in der dritten Spalte der vollqualifizierte Namen angegeben, wobei auch die Kardinalitäten zur Verbesserung des Referenzcharakters der Tabelle mit angeführt sind.

Wie im Rahmen dieser Arbeit üblich werden Meta-Entitätstypen, die zwingend in Meta-Beziehungstypen des Integrierten EIA/CDIF-Meta-Modells enthalten sein müssen, fett dargestellt. Ebenso wird im entsprechenden Meta-Beziehungstyp zusätzlich auch der entsprechende minimale Kardinalitätswert fett hervorgehoben. Desgleichen werden die Namen von zwingend vorgeschriebenen Meta-Attributen fett gesetzt.

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
1	RootObject (AMO)	
2	RootEntity (ME)	
3	IsRelatedTo (MR)	0:N RootEntity.*IsRelatedTo*.RootEntity 0:N
4	SemanticInformationObject (ME)	
5	**CDIFIdentifier** (MA)	(AMO: RootObject)
6	DateCreated (MA)	(AMO: RootObject)
7	DateUpdated (MA)	(AMO: RootObject)
8	TimeCreated (MA)	(AMO: RootObject)
9	TimeUpdated (MA)	(AMO: RootObject)
10	Foundation (SA)	
11	Common (SA)	
12	AbstractionLevel (ME)	
13	**Name** (MA)	(ME: AbstractionLevel)
14	**AlternateName** (ME)	
15	OtherLongName (MA)	(ME: **AlternateName**)
16	OtherName (MA)	(ME: **AlternateName**)
17	Attribute (ME)	
18	DefaultValue (MA)	(ME: Attribute)
19	IsOptional (MA)	(ME: Attribute)
21	Name (MA)	(ME: Attribute)
22	DataObject (ME)	
23	Name (MA)	(ME: DataObject)
26	**Derivation** (ME)	
27	DerivationText (MA)	(ME: **Derivation**)
29	DerivationLanguage (MA)	(ME: **Derivation**)
30	PresentationInformationObject (ME)	
31	ProcessObject (ME)	
32	ExecutionTimeUnit (MA)	(ME: ProcessObject)
33	Name (MA)	(ME: ProcessObject)
34	ExecutionTimeInterval (MA)	(ME: ProcessObject)
35	SpecificationText (MA)	(ME: ProcessObject)
36	SpecificationLanguage (MA)	(ME: ProcessObject)
37	ProjectedAttribute (ME)	
38	SpecificationText (MA)	(ME: ProjectedAttribute)

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
39	SpecificationLanguage (MA)	(ME: ProjectedAttribute)
44	BriefDescription (MA)	(ME: SemanticInformationObject)
45	FullDescription (MA)	(ME: SemanticInformationObject)
51	**TextualConstraint** (ME)	
52	BriefDescription (MA)	(ME: **TextualConstraint**)
53	ConstraintExpression (MA)	(ME: **TextualConstraint**)
54	FullDescription (MA)	(ME: **TextualConstraint**)
55	ConstraintLanguage (MA)	(ME: **TextualConstraint**)
56	ToolUser (ME)	
57	FullName (MA)	(ME: ToolUser)
58	**SystemName** (MA)	(ME: ToolUser)
63	IsProjectionOf (MR)	0:N ProjectedAttribute.*IsProjectionOf*.-Attribute 0:N
68	CreatedBy (MR)	0:N RootEntity.*CreatedBy*.ToolUser 0:1
69	Has (MR)	**1**:1 RootEntity.*Has*.**AlternateName** 0:N
70	LastUpdatedBy (MR)	0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1
71	Uses (MR)	0:N RootEntity.*Uses*.AlternateName 0:1
72	IsCategorizedIn (MR)	0:N SemanticInformationObject.-*IsCategorizedIn*.AbstractionLevel 0:N
73	ProducedBy (MR)	**1**:N SemanticInformationObject.*ProducedBy*.-**Derivation** 0:N
74	UsedIn (MR)	**1**:N SemanticInformationObject.*UsedIn*.-**Derivation** 0:N
80	IsConstraintOn (MR)	0:N **TextualConstraint**.*IsConstraintOn*.-SemanticInformationObject **1**:N
185	IsRealizationOf (MA)	(ME: **Derivation**)
209	DataFlowModel (SA)	
210	DataFlowModel (ME)	
211	Name (MA)	(ME: DataFlowModel)
212	Type (MA)	(ME: DataFlowModel)
213	DFMProcessDefinition (ME)	
214	FunctionalArea (MA)	(ME: DFMProcessDefinition)
215	HasConcurrentChildren (MA)	(ME: DFMProcessDefinition)
216	IsControl (MA)	(ME: DFMProcessDefinition)
217	IsData (MA)	(ME: DFMProcessDefinition)
218	IsMaterial (MA)	(ME: DFMProcessDefinition)
221	FlowDefinition (ME)	
223	IsControl (MA)	(ME: FlowDefinition)
224	IsData (MA)	(ME: FlowDefinition)
225	IsMaterial (MA)	(ME: FlowDefinition)
227	Flow (ME)	
228	ContextDescription (MA)	(ME: Flow)
231	Name (MA)	(ME: Flow)
232	FlowProducerConsumer (ME)	

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
239	StoreDefinition (ME)	
240	IsControl (MA)	(ME: StoreDefinition)
241	IsData (MA)	(ME: StoreDefinition)
242	IsManual (MA)	(ME: StoreDefinition)
243	IsMaterial (MA)	(ME: StoreDefinition)
244	IsPermanent (MA)	(ME: StoreDefinition)
245	IsReadOnly (MA)	(ME: StoreDefinition)
247	Size (MA)	(ME: StoreDefinition)
248	SizeUnits (MA)	(ME: StoreDefinition)
1000	DataModeling (SA)	
1001	**AccessPath** (ME)	
1003	CandidateKey (ME)	
1004	IsPrimary (MA)	(ME: CandidateKey)
1005	Cluster (ME)	
1008	DataModel (ME)	
1010	ModelType (MA)	(ME: DataModel)
1011	Name (MA)	(ME: DataModel)
1012	DataModelObject (ME)	
1013	**DataModelSubset** (ME)	
1015	Name (MA)	(ME: **DataModelSubset**)
1016	Entity (ME)	
1017	AvgNumberOfOccurrences (MA)	(ME: Entity)
1018	DeletionTimePeriod (MA)	(ME: Entity)
1019	EntityType (MA)	(ME: Entity)
1020	InsertionTimePeriod (MA)	(ME: Entity)
1022	MaxNumberOfOccurrences (MA)	(ME: Entity)
1023	MinNumberOfOccurrences (MA)	(ME: Entity)
1024	NormalizationState (MA)	(ME: Entity)
1025	NumberOfDeletions (MA)	(ME: Entity)
1026	NumberOfInsertions (MA)	(ME: Entity)
1027	NumberOfReads (MA)	(ME: Entity)
1028	NumberOfUpdates (MA)	(ME: Entity)
1029	ReadTimePeriod (MA)	(ME: Entity)
1030	UpdateTimePeriod (MA)	(ME: Entity)
1031	Usage (MA)	(ME: Entity)
1032	**ForeignKey** (ME)	
1033	InheritableDataModelObject (ME)	
1034	IsAbstract (MA)	(ME: InheritableDataModelObject)
1035	**Key** (ME)	
1036	**ProjectionComponent** (ME)	
1037	Name (MA)	(ME: **ProjectionComponent**)
1038	SpecificationLanguage (MA)	(ME: **ProjectionComponent**)
1039	SpecificationText (MA)	(ME: **ProjectionComponent**)

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
1040	**Relationship** (ME)	
1041	InverseName (MA)	(ME: **Relationship**)
1042	**Role** (ME)	
1043	IsMaster (MA)	(ME: **Role**)
1044	IsSource (MA)	(ME: **Role**)
1045	RoleConstraint (ME)	
1046	Name (MA)	(ME: RoleConstraint)
1047	Operator (MA)	(ME: RoleConstraint)
1048	**RolePlayer** (ME)	
1049	AvgNumberOfOccurrences (MA)	(ME: **RolePlayer**)
1050	DeleteEffect (MA)	(ME: **RolePlayer**)
1051	DeletionTimePeriod (MA)	(ME: **RolePlayer**)
1052	InsertEffect (MA)	(ME: **RolePlayer**)
1053	InsertionTimePeriod (MA)	(ME: **RolePlayer**)
1054	IsDeleteDeferrable (MA)	(ME: **RolePlayer**)
1055	IsInsertDeferrable (MA)	(ME: **RolePlayer**)
1056	IsUpdateDeferrable (MA)	(ME: **RolePlayer**)
1057	MaxInnerCardinality (MA)	(ME: **RolePlayer**)
1058	MaxNumberOfOccurrences (MA)	(ME: **RolePlayer**)
1059	MaxOuterCardinality (MA)	(ME: **RolePlayer**)
1060	MinInnerCardinality (MA)	(ME: **RolePlayer**)
1061	MinNumberOfOccurrences (MA)	(ME: **RolePlayer**)
1062	MinOuterCardinality (MA)	(ME: **RolePlayer**)
1063	NumberOfDeletions (MA)	(ME: **RolePlayer**)
1064	NumberOfInsertions (MA)	(ME: **RolePlayer**)
1065	NumberOfReads (MA)	(ME: **RolePlayer**)
1066	NumberOfUpdates (MA)	(ME: **RolePlayer**)
1067	ReadTimePeriod (MA)	(ME: **RolePlayer**)
1068	UpdateEffect (MA)	(ME: **RolePlayer**)
1069	UpdateTimePeriod (MA)	(ME: **RolePlayer**)
1070	**SubtypeSet** (ME)	
1071	IsExclusive (MA)	(ME: **SubtypeSet**)
1072	Name (MA)	(ME: **SubtypeSet**)
1073	SubtypeListIsClosed (MA)	(ME: **SubtypeSet**)
1074	**SubtypeSetMembershipCriterion** (ME)	
1075	SpecificationLanguage (MA)	(ME: **SubtypeSetMembershipCriterion**)
1076	SpecificationText (MA)	(ME: **SubtypeSetMembershipCriterion**)
1077	Name (MA)	(ME: **Key**)
1078	SpecificationLanguage (MA)	(ME: **Key**)
1080	DiscriminatorValue (MA)	(ME: **SubtypeSetMembershipCriterion**)
1081	Name (MA)	(ME: **AccessPath**)
1082	SpecificationLanguage (MA)	(ME: **AccessPath**)

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
1083	SpecificationText (MA)	(ME: **AccessPath**)
1113	**EquivalenceSet** (ME)	
1114	HasMember (MR)	0:N **EquivalenceSet**.*HasMember*.-ComponentObject **2**:N
1118	Operator (MA)	(ME: DefinitionObject)
1119	Name (MA)	(ME: DefinitionObject)
1120	ContextDescription (MA)	(ME: Store)
1121	ContextIdentifier (MA)	(ME: Store)
1122	ContextDescription (MA)	(ME: DFMProcess)
1123	ContextIdentifier (MA)	(ME: DFMProcess)
1124	ContextDescription (MA)	(ME: ExternalAgent)
1125	ContextIdentifier (MA)	(ME: ExternalAgent)
1129	Port (ME)	
1130	HasRealTimeSemantics (MA)	(ME: DFMProcessDefinition)
1131	Contains (MR)	0:1 DefinitionObject.*Contains*.-ComponentObject 0:N
1139	**IsFormal** (MA)	(ME: Port)
1140	SpecificationText (MA)	(ME: DefinitionObject)
1141	SpecificationLanguage (MA)	(ME: DefinitionObject)
1142	IsAnalog (MA)	(ME: FlowDefinition)
1150	Name (MA)	(ME: DFMProcess)
1151	Name (MA)	(ME: ExternalAgent)
1152	Name (MA)	(ME: Port)
1153	Name (MA)	(ME: Store)
1700	Instantiates (MR)	0:1 AccessPath.*Instantiates*.Key 0:1
1701	IsDiscriminatorFor (MR)	0:N Attribute.*IsDiscriminatorFor*.SubtypeSet-MembershipCriterion 0:N
1702	IsInheritedFrom (MR)	0:N Attribute.*IsInheritedFrom*.Attribute 0:1
1703	Collects (MR)	0:N Cluster.*Collects*.DataModelObject 0:N
1704	Collects (MR)	0:N DataModel.*Collects*.DataModelObject 0:N
1705	ActsAs (MR)	**1**:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N
1706	IsMemberOf (MR)	0:N DataModelObject.*IsMemberOf*.-DataModelSubset 0:N
1707	Excludes (MR)	0:N DataModelSubset.*Excludes*.Attribute 0:N
1708	IsSubsetOf (MR)	0:N **DataModelSubset**.*IsSubsetOf*.-DataModel **1**:1
1709	IsConstructedWith (MR)	**1**:1 DefinitionObject.*IsConstructedWith*.-**ProjectionComponent** 0:N
1710	SequenceNumber (MA)	(MR: **1**:1 DefinitionObject.*IsConstructedWith*.-**ProjectionComponent** 0:N)
1711	IsIdentifiedBy (MR)	**1**:1 Entity.*IsIdentifiedBy*.**Key** 0:N
1712	IsAccessedUsing (MR)	**1**:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N
1713	Incorporates (MR)	0:N CandidateKey.*Incorporates*.Foreign-Key 0:N
1714	References (MR)	0:N **ForeignKey**.*References*.Candidate-

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
		Key **1**:1
1715	IsSubtypeIn (MR)	**1**:N InheritableDataModelObject.*IsSubtypeIn*.-**SubtypeSet** 0:N
1716	Incorporates (MR)	0:N AccessPath.*Incorporates*.Attribute 0:N
1717	SpecificationLanguage (MA)	(MR: **1**:N InheritableDataModelObject.-*IsSubtypeIn*.**SubtypeSet** 0:N)
1718	SpecificationText (MA)	(MR: **1**:N InheritableDataModelObject.-*IsSubtypeIn*.**SubtypeSet** 0:N)
1719	IsSupertypeFor (MR)	**1**:1 InheritableDataModelObject.-*IsSupertypeFor*.**SubtypeSet** 0:N
1721	IsFullProjectionOf (MR)	0:N **ProjectionComponent**.*IsFull-ProjectionOf*.DefinitionObject **1**:1
1722	IsProjectionOf (MR)	0:N **ProjectionComponent**.*IsProjectionOf*.-Attribute **1**:N
1723	SequenceNumber (MA)	(MR: 0:N **ProjectionComponent**.-*IsProjectionOf*.Attribute **1**:N)
1724	BelongsTo (MR)	**2**:N **Role**.*BelongsTo*.**Relationship 1**:1
1725	Incorporates (MR)	0:N RoleConstraint.*Incorporates*.-RoleConstraint 0:N
1726	Incorporates (MR)	0:N RoleConstraint.*Incorporates*.-RolePlayer 0:N
1727	Incorporates (MR)	0:N RoleConstraint.*Incorporates*.Semantic-InformationObject 0:N
1728	Incorporates (MR)	0:1 ForeignKey.*Incorporates*.RolePlayer 0:1
1729	IsSupportedBy (MR)	0:N RolePlayer.*IsSupportedBy*.Key 0:1
1730	Plays (MR)	**1**:N RolePlayer.*Plays*.**Role** 0:1
1731	Refines (MR)	0:N RolePlayer.*Refines*.RolePlayer 0:1
1732	RefinesForSubtype (MR)	0:1 RolePlayer.*RefinesForSubtype*.-DataModelObject 0:N
1733	Incorporates (MR)	0:N Key.*Incorporates*.Attribute 0:N
1734	IsAscending (MA)	(MR: 0:N Key.*Incorporates*.Attribute 0:N)
1735	SequenceNumber (MA)	(MR: 0:N Key.*Incorporates*.-SemanticInformationObject 0:N)
1736	StoreWithSupertype (MA)	(MR: **1**:N InheritableDataModelOb-ject.*IsSubtypeIn*.**SubtypeSet** 0:N)
1737	Selects (MR)	0:N **SubtypeSetMembershipCriterion**.-*Selects*.InheritableDataModelObject **1**:1
1738	IsAscending (MA)	(MR: 0:N AccessPath.*Incorporates*.-Attribute 0:N)
1739	SequenceNumber (MA)	(MR: 0:N AccessPath.*Incorporates*.-Attribute 0:N)
6000	FlowOutputPort (ME)	
6001	FlowInputPort (ME)	
6002	ExternalAgentDefinition (ME)	
6004	ExternalAgent (ME)	
6005	DFMProcess (ME)	
6010	SupportPort (ME)	

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
6012	ConstraintPort (ME)	
6014	ControlPort (ME)	
6015	FlowPort (ME)	
6023	ProducesOrConsumes (MR)	0:N FlowProducerConsumer.*ProducesOr-Consumes*.Flow 0:N
6024	Consumes (MR)	0:N FlowProducerConsumer.*Consumes*.-Flow 0:N
6025	Produces (MR)	0:N FlowProducerConsumer.*Produces*.-Flow 0:N
6026	HasRoot (MR)	0:1 DataFlowModel.*HasRoot*.DFMProcess-Definition 0:1
6028	Store (ME)	
6040	**SequenceNumber** (MA)	(MR: 0:N **ReferencedElement**.*DefinesPath*.-ComponentObject **1**:N)
6041	ContextIdentifier (MA)	(ME: Flow)
8000	ComponentObject (ME)	
8002	DefinitionObject (ME)	
8020	**ReferencedElement** (ME)	
8022	References (MR)	0:N ComponentObject.*References*.Definition-Object 0:1
8025	DefinesPath (MR)	0:N **ReferencedElement**.*DefinesPath*.-ComponentObject **1**:N
8026	SequenceNumber (MA)	(MR: 0:1 DefinitionObject.*Contains*.-ComponentObject 0:N)
10000	PresentationLocationAnd-Connectivity (SA)	
10002	Diagram (ME)	
10003	**Point** (ME)	
10004	**GraphicalElement** (ME)	
10005	**PositionedElement** (ME)	
10006	Edge (ME)	
10007	**EdgeElement** (ME)	
10008	**RelativePoint** (ME)	
10009	AbsolutePoint (ME)	
10010	Annotation (ME)	
10011	Node (ME)	
10013	**AnnotationArgument** (ME)	
10014	SemanticObjectReference (ME)	
10051	Represents (MR)	0:N PresentationInformationObject.*Repre-sents*.SemanticObjectReference 0:N
10052	AppearsOn (MR)	0:N **GraphicalElement**.*AppearsOn*.-Diagram **1**:1
10053	IsLocatedOn (MR)	0:N **Point**.*IsLocatedOn*.Diagram **1**:1
10054	HasCenter (MR)	0:N **PositionedElement**.*HasCenter*.Point **1**:1
10055	IsAttachedTo (MR)	0:N Edge.*IsAttachedTo*.GraphicalElement 0:2
10056	ConsistsOf (MR)	**1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
10057	HasEnd (MR)	0:N **EdgeElement**.*HasEnd*.Point **2**:2
10058	IsRelativeTo (MR)	0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1
10059	Uses (MR)	**1**:1 Annotation.*Uses*.-**Annotation-Argument** 0:N
10060	IsDependentUpon (MR)	0:N Point.*IsDependentUpon*.-PositionedElement 0:1
10101	**Position** (MA)	(ME: AbsolutePoint)
10102	DerivationLanguage (MA)	(ME: Annotation)
10103	DerivationText (MA)	(ME: Annotation)
10104	DataBlock (MA)	(ME: **AnnotationArgument**)
10105	ExtentX (MA)	(ME: Diagram)
10106	ExtentY (MA)	(ME: Diagram)
10107	ExtentZ (MA)	(ME: Diagram)
10108	DotsPerUnit (MA)	(ME: Diagram)
10109	Unit (MA)	(ME: Diagram)
10110	**ReferencedMetaObjectInstance** (MA)	(ME: SemanticObjectReference)
10111	ReferencedMetaAttributeName (MA)	(ME: SemanticObjectReference)
10114	SequenceNumber (MA)	(MR: **1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N)
10117	SequenceNumber (MA)	(MR: **1**:1 Annotation.*Uses*.**Annotation-Argument** 0:N)
10118	MIMEType (MA)	(ME: **AnnotationArgument**)
10119	MIMESubtype (MA)	(ME: **AnnotationArgument**)
10121	ExtentX (MA)	(ME: **PositionedElement**)
10122	ExtentY (MA)	(ME: **PositionedElement**)
10123	ExtentZ (MA)	(ME: **PositionedElement**)
10124	**DeltaX** (MA)	(ME: **RelativePoint**)
10125	**DeltaY** (MA)	(ME: **RelativePoint**)
10126	DeltaZ (MA)	(ME: **RelativePoint**)
10131	Transform11 (MA)	(ME: **RelativePoint**)
10132	Transform12 (MA)	(ME: **RelativePoint**)
10133	Transform13 (MA)	(ME: **RelativePoint**)
10134	Transform21 (MA)	(ME: **RelativePoint**)
10135	Transform22 (MA)	(ME: **RelativePoint**)
10136	Transform23 (MA)	(ME: **RelativePoint**)
10137	Transform31 (MA)	(ME: **RelativePoint**)
10138	Transform32 (MA)	(ME: **RelativePoint**)
10139	Transform33 (MA)	(ME: **RelativePoint**)
1140b[825]	SpecificationText (MA)	(ME: **Key**)

825 Dieses Meta-Objekt weist ursprünglich einen bereits vergebenen Surrogatwert auf. Um den Wert eindeutig zu machen, wurde im Rahmen dieser Arbeit dem zweiten Auftreten des Surrogatwertes ein kleines „b“ angefügt.

CDIF-Meta-Identifier	Name	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
1733b[825]	Incorporates (MR)	0:N Key.*Incorporates*.SemanticInformation-Object 0:N
1736b[825]	Specifies (MR)	**1**:1 SubtypeSet.*Specifies*.**SubtypeSet-MembershipCriterion** 0:N

Tabelle 6-1: Standardisierte Meta-Objekte sortiert nach dem Meta-Meta-Attribut „CDIFMetaIdentifier“

6.1.2 Querverweistabelle der standardisierten Meta-Objekte, sortiert nach dem Meta-Meta-Attribut „Name“

Die folgende Tabelle 6-2 listet sämtliche standardisierten Meta-Objekte des Integrierten EIA/CDIF-Meta-Modells, sortiert nach dem Meta-Meta-Attribut „Name“. In der ersten Spalte findet sich der Name des Meta-Objekts gemeinsam mit der Abkürzung seines Metatyps, in der zweiten Spalte wird der Surrogatwert des Meta-Meta-Attributs „CDIFMetaIdentifier“ angeführt. Für Meta-Attribute wird in der dritten Spalte in Klammern eingeschlossen das AttribuierbareMetaObjekt gezeigt, für das es definiert wurde. Für Meta-Beziehungstypen wird in der dritten Spalte der vollqualifizierte Namen angegeben, wobei auch die Kardinalitäten zur Verbesserung des Referenzcharakters der Tabelle mit angeführt sind.

Wie im Rahmen dieser Arbeit üblich werden Meta-Entitätstypen, die zwingend in Meta-Beziehungstypen des Integrierten EIA/CDIF-Meta-Modells enthalten sein müssen, fett dargestellt. Ebenso wird im entsprechenden Meta-Beziehungstyp zusätzlich auch der entsprechende minimale Kardinalitätswert fett hervorgehoben. Desgleichen werden die Namen von zwingend vorgeschriebenen Meta-Attributen fett gesetzt.

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
AbsolutePoint (ME)	10009	
AbstractionLevel (ME)	12	
AccessPath (ME)	1001	
ActsAs (MR)	1705	**1**:1 DataModelObject.*ActsAs*.**RolePlayer** 0:N
AlternateName (ME)	14	
Annotation (ME)	10010	
AnnotationArgument (ME)	10013	
AppearsOn (MR)	10052	0:N **GraphicalElement**.*AppearsOn*.-Diagram **1**:1
Attribute (ME)	17	
AvgNumberOfOccurrences (MA)	1017	(ME: Entity)
AvgNumberOfOccurrences (MA)	1049	(ME: **RolePlayer**)
BelongsTo (MR)	1724	**2**:N **Role**.*BelongsTo*.**Relationship 1**:1
BriefDescription (MA)	44	(ME: SemanticInformationObject)
BriefDescription (MA)	52	(ME: **TextualConstraint**)

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
CDIFIdentifier (MA)	5	(AMO: RootObject)
CandidateKey (ME)	1003	
Cluster (ME)	1005	
Collects (MR)	1703	0:N Cluster.*Collects*.DataModelObject 0:N
Collects (MR)	1704	0:N DataModel.*Collects*.DataModelObject 0:N
Common (SA)	11	
ComponentObject (ME)	8000	
ConsistsOf (MR)	10056	**1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N
ConstraintExpression (MA)	53	(ME: **TextualConstraint**)
ConstraintLanguage (MA)	55	(ME: **TextualConstraint**)
ConstraintPort (ME)	6012	
Consumes (MR)	6024	0:N FlowProducerConsumer.*Consumes*.-Flow 0:N
Contains (MR)	1131	0:1 DefinitionObject.*Contains*.Component-Object 0:N
ContextDescription (MA)	1122	(ME: DFMProcess)
ContextDescription (MA)	228	(ME: Flow)
ContextDescription (MA)	1120	(ME: Store)
ContextDescription (MA)	1124	(ME: ExternalAgent)
ContextIdentifier (MA)	1123	(ME: DFMProcess)
ContextIdentifier (MA)	1121	(ME: Store)
ContextIdentifier (MA)	6041	(ME: Flow)
ContextIdentifier (MA)	1125	(ME: ExternalAgent)
ControlPort (ME)	6014	
CreatedBy (MR)	68	0:N RootEntity.*CreatedBy*.ToolUser 0:1
DFMProcess (ME)	6005	
DFMProcessDefinition (ME)	213	
DataBlock (MA)	10104	(ME: **AnnotationArgument**)
DataFlowModel (ME)	210	
DataFlowModel (SA)	209	
DataModel (ME)	1008	
DataModelObject (ME)	1012	
DataModelSubset (ME)	1013	
DataModeling (SA)	1000	
DataObject (ME)	22	
DateCreated (MA)	6	(AMO: RootObject)
DateUpdated (MA)	7	(AMO: RootObject)
DefaultValue (MA)	18	(ME: Attribute)
DefinesPath (MR)	8025	0:N **ReferencedElement**.*DefinesPath*.-ComponentObject **1**:N
DefinitionObject (ME)	8002	
DeleteEffect (MA)	1050	(ME: **RolePlayer**)
DeletionTimePeriod (MA)	1018	(ME: Entity)
DeletionTimePeriod (MA)	1051	(ME: **RolePlayer**)

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
DeltaX (MA)	10124	(ME: **RelativePoint**)
DeltaY (MA)	10125	(ME: **RelativePoint**)
DeltaZ (MA)	10126	(ME: **RelativePoint**)
Derivation (ME)	26	
DerivationLanguage (MA)	29	(ME: **Derivation**)
DerivationLanguage (MA)	10102	(ME: Annotation)
DerivationText (MA)	27	(ME: **Derivation**)
DerivationText (MA)	10103	(ME: Annotation)
Diagram (ME)	10002	
DiscriminatorValue (MA)	1080	(ME: **SubtypeSetMembershipCriterion**)
DotsPerUnit (MA)	10108	(ME: Diagram)
Edge (ME)	10006	
EdgeElement (ME)	10007	
Entity (ME)	1016	
EntityType (MA)	1019	(ME: Entity)
EquivalenceSet (ME)	1113	
Excludes (MR)	1707	0:N DataModelSubset.*Excludes.*Attribute 0:N
ExecutionTimeInterval (MA)	34	(ME: ProcessObject)
ExecutionTimeUnit (MA)	32	(ME: ProcessObject)
ExtentX (MA)	10105	(ME: Diagram)
ExtentX (MA)	10121	(ME: **PositionedElement**)
ExtentY (MA)	10106	(ME: Diagram)
ExtentY (MA)	10122	(ME: **PositionedElement**)
ExtentZ (MA)	10107	(ME: Diagram)
ExtentZ (MA)	10123	(ME: **PositionedElement**)
ExternalAgent (ME)	6004	
ExternalAgentDefinition (ME)	6002	
Flow (ME)	227	
FlowDefinition (ME)	221	
FlowInputPort (ME)	6001	
FlowOutputPort (ME)	6000	
FlowPort (ME)	6015	
FlowProducerConsumer (ME)	232	
ForeignKey (ME)	1032	
Foundation (SA)	10	
FullDescription (MA)	45	(ME: SemanticInformationObject)
FullDescription (MA)	54	(ME: **TextualConstraint**)
FullName (MA)	57	(ME: ToolUser)
FunctionalArea (MA)	214	(ME: DFMProcessDefinition)
GraphicalElement (ME)	10004	
Has (MR)	69	**1**:1 RootEntity.*Has.***AlternateName** 0:N
HasCenter (MR)	10054	0:N **PositionedElement**.*HasCenter.*Point **1**:1
HasConcurrentChildren (MA)	215	(ME: DFMProcessDefinition)

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
HasEnd (MR)	10057	0:N **EdgeElement**.*HasEnd*.Point **2**:2
HasMember (MR)	1114	0:N **EquivalenceSet**.*HasMember*.Component-Object **2**:N
HasRealTimeSemantics (MA)	1130	(ME: DFMProcessDefinition)
HasRoot (MR)	6026	0:1 DataFlowModel.*HasRoot*.DFMProcess-Definition 0:1
Incorporates (MR)	1716	0:N AccessPath.*Incorporates*.Attribute 0:N
Incorporates (MR)	1713	0:N CandidateKey.*Incorporates*.Foreign-Key 0:N
Incorporates (MR)	1728	0:1 ForeignKey.*Incorporates*.RolePlayer 0:1
Incorporates (MR)	1733	0:N Key.*Incorporates*.Attribute 0:N
Incorporates (MR)	1733b[826]	0:N Key.*Incorporates*.SemanticInformation-Object 0:N
Incorporates (MR)	1725	0:N RoleConstraint.*Incorporates*.Role-Constraint 0:N
Incorporates (MR)	1726	0:N RoleConstraint.*Incorporates*.Role-Player 0:N
Incorporates (MR)	1727	0:N RoleConstraint.*Incorporates*.Semantic-InformationObject 0:N
InheritableDataModelObject (ME)	1033	
InsertEffect (MA)	1052	(ME: **RolePlayer**)
InsertionTimePeriod (MA)	1020	(ME: Entity)
InsertionTimePeriod (MA)	1053	(ME: **RolePlayer**)
Instantiates (MR)	1700	0:1 AccessPath.*Instantiates*.Key 0:1
InverseName (MA)	1041	(ME: **Relationship**)
IsAbstract (MA)	1034	(ME: InheritableDataModelObject)
IsAccessedUsing (MR)	1712	**1**:1 Entity.*IsAccessedUsing*.**AccessPath** 0:N
IsAnalog (MA)	1142	(ME: FlowDefinition)
IsAscending (MA)	1738	(MR: 0:N AccessPath.*Incorporates*.-Attribute 0:N)
IsAscending (MA)	1734	(MR: 0:N Key.*Incorporates*.Attribute 0:N)
IsAttachedTo (MR)	10055	0:N Edge.*IsAttachedTo*.GraphicalElement 0:2
IsCategorizedIn (MR)	72	0:N SemanticInformationObject.-*IsCategorizedIn*.AbstractionLevel 0:N
IsConstraintOn (MR)	80	0:N **TextualConstraint**.*IsConstraintOn*.-SemanticInformationObject **1**:N
IsConstructedWith (MR)	1709	**1**:1 DefinitionObject.*IsConstructedWith*.-**ProjectionComponent** 0:N
IsControl (MA)	216	(ME: DFMProcessDefinition)
IsControl (MA)	240	(ME: StoreDefinition)
IsControl (MA)	223	(ME: FlowDefinition)
IsData (MA)	217	(ME: DFMProcessDefinition)

826 Dieses Meta-Objekt weist ursprünglich einen bereits vergebenen Surrogatwert auf. Um den Wert eindeutig zu machen, wurde im Rahmen dieser Arbeit dem zweiten Auftreten des Surrogatwertes ein kleines „b“ angefügt.

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
IsData (MA)	241	(ME: StoreDefinition)
IsData (MA)	224	(ME: FlowDefinition)
IsDeleteDeferrable (MA)	1054	(ME: **RolePlayer**)
IsDependentUpon (MR)	10060	0:N Point.*IsDependentUpon*.Positioned-Element 0:1
IsDiscriminatorFor (MR)	1701	0:N Attribute.*IsDiscriminatorFor*.SubtypeSet-MembershipCriterion 0:N
IsExclusive (MA)	1071	(ME: **SubtypeSet**)
IsFormal (MA)	1139	(ME: Port)
IsFullProjectionOf (MR)	1721	0:N **ProjectionComponent**.*IsFull-ProjectionOf*.DefinitionObject **1**:1
IsIdentifiedBy (MR)	1711	**1**:1 Entity.*IsIdentifiedBy*.**Key** 0:N
IsInheritedFrom (MR)	1702	0:N Attribute.*IsInheritedFrom*.Attribute 0:1
IsInsertDeferrable (MA)	1055	(ME: **RolePlayer**)
IsLocatedOn (MR)	10053	0:N **Point**.*IsLocatedOn*.Diagram **1**:1
IsManual (MA)	242	(ME: StoreDefinition)
IsMaster (MA)	1043	(ME: **Role**)
IsMaterial (MA)	218	(ME: DFMProcessDefinition)
IsMaterial (MA)	225	(ME: FlowDefinition)
IsMaterial (MA)	243	(ME: StoreDefinition)
IsMemberOf (MR)	1706	0:N DataModelObject.*IsMemberOf*.DataModel-Subset 0:N
IsOptional (MA)	19	(ME: Attribute)
IsPermanent (MA)	244	(ME: StoreDefinition)
IsPrimary (MA)	1004	(ME: CandidateKey)
IsProjectionOf (MR)	63	0:N ProjectedAttribute.*IsProjectionOf*.-Attribute 0:N
IsProjectionOf (MR)	1722	0:N **ProjectionComponent**.*IsProjectionOf*.-Attribute **1**:N
IsReadOnly (MA)	245	(ME: StoreDefinition)
IsRealizationOf (MA)	185	(ME: **Derivation**)
IsRelatedTo (MR)	3	0:N RootEntity.*IsRelatedTo*.RootEntity 0:N
IsRelativeTo (MR)	10058	0:N **RelativePoint**.*IsRelativeTo*.Point **1**:1
IsSource (MA)	1044	(ME: **Role**)
IsSubsetOf (MR)	1708	0:N **DataModelSubset**.*IsSubsetOf*.Data-Model **1**:1
IsSubtypeIn (MR)	1715	**1**:N InheritableDataModelObject.*IsSubtypeIn*.-**SubtypeSet** 0:N
IsSupertypeFor (MR)	1719	**1**:1 InheritableDataModelObject.*IsSupertype-For*.**SubtypeSet** 0:N
IsSupportedBy (MR)	1729	0:N RolePlayer.*IsSupportedBy*.Key 0:1
IsUpdateDeferrable (MA)	1056	(ME: **RolePlayer**)
Key (ME)	1035	
LastUpdatedBy (MR)	70	0:N RootEntity.*LastUpdatedBy*.ToolUser 0:1
MIMESubtype (MA)	10119	(ME: **AnnotationArgument**)

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
MIMEType (MA)	10118	(ME: **AnnotationArgument**)
MaxInnerCardinality (MA)	1057	(ME: **RolePlayer**)
MaxNumberOfOccurrences (MA)	1022	(ME: Entity)
MaxNumberOfOccurrences (MA)	1058	(ME: **RolePlayer**)
MaxOuterCardinality (MA)	1059	(ME: **RolePlayer**)
MinInnerCardinality (MA)	1060	(ME: **RolePlayer**)
MinNumberOfOccurrences (MA)	1023	(ME: Entity)
MinNumberOfOccurrences (MA)	1061	(ME: **RolePlayer**)
MinOuterCardinality (MA)	1062	(ME: **RolePlayer**)
ModelType (MA)	1010	(ME: DataModel)
Name (MA)	13	(ME: AbstractionLevel)
Name (MA)	23	(ME: DataObject)
Name (MA)	1081	(ME: **AccessPath**)
Name (MA)	1153	(ME: Store)
Name (MA)	1152	(ME: Port)
Name (MA)	231	(ME: Flow)
Name (MA)	1151	(ME: ExternalAgent)
Name (MA)	1150	(ME: DFMProcess)
Name (MA)	211	(ME: DataFlowModel)
Name (MA)	1072	(ME: **SubtypeSet**)
Name (MA)	1046	(ME: RoleConstraint)
Name (MA)	1037	(ME: **ProjectionComponent**)
Name (MA)	1077	(ME: **Key**)
Name (MA)	1119	(ME: DefinitionObject)
Name (MA)	1015	(ME: **DataModelSubset**)
Name (MA)	1011	(ME: DataModel)
Name (MA)	21	(ME: Attribute)
Name (MA)	33	(ME: ProcessObject)
Node (ME)	10011	
NormalizationState (MA)	1024	(ME: Entity)
NumberOfDeletions (MA)	1025	(ME: Entity)
NumberOfDeletions (MA)	1063	(ME: **RolePlayer**)
NumberOfInsertions (MA)	1026	(ME: Entity)
NumberOfInsertions (MA)	1064	(ME: **RolePlayer**)
NumberOfReads (MA)	1027	(ME: Entity)
NumberOfReads (MA)	1065	(ME: **RolePlayer**)
NumberOfUpdates (MA)	1028	(ME: Entity)
NumberOfUpdates (MA)	1066	(ME: **RolePlayer**)
Operator (MA)	1118	(ME: DefinitionObject)
Operator (MA)	1047	(ME: RoleConstraint)
OtherLongName (MA)	15	(ME: **AlternateName**)
OtherName (MA)	16	(ME: **AlternateName**)
Plays (MR)	1730	**1**:N RolePlayer.*Plays*.**Role** 0:1

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
Point (ME)	10003	
Port (ME)	1129	
Position (MA)	10101	(ME: AbsolutePoint)
PositionedElement (ME)	10005	
PresentationInformationObject (ME)	30	
PresentationLocationAnd-Connectivity (SA)	10000	
ProcessObject (ME)	31	
ProducedBy (MR)	73	**1**:N SemanticInformationObject.*ProducedBy*.-**Derivation** 0:N
Produces (MR)	6025	0:N FlowProducerConsumer.*Produces*.-Flow 0:N
ProducesOrConsumes (MR)	6023	0:N FlowProducerConsumer.*ProducesOr-Consumes*.Flow 0:N
ProjectedAttribute (ME)	37	
ProjectionComponent (ME)	1036	
ReadTimePeriod (MA)	1029	(ME: Entity)
ReadTimePeriod (MA)	1067	(ME: **RolePlayer**)
ReferencedElement (ME)	8020	
ReferencedMetaAttributeName (MA)	10111	(ME: SemanticObjectReference)
ReferencedMetaObjectInstance (MA)	10110	(ME: SemanticObjectReference)
References (MR)	8022	0:N ComponentObject.*References*.Definition-Object 0:1
References (MR)	1714	0:N **ForeignKey**.*References*.Candidate-Key **1**:1
Refines (MR)	1731	0:N RolePlayer.*Refines*.RolePlayer 0:1
RefinesForSubtype (MR)	1732	0:1 RolePlayer.*RefinesForSubtype*.Data-ModelObject 0:N
Relationship (ME)	1040	
RelativePoint (ME)	10008	
Represents (MR)	10051	0:N PresentationInformationObject.*Repre-sents*.SemanticObjectReference 0:N
Role (ME)	1042	
RoleConstraint (ME)	1045	
RolePlayer (ME)	1048	
RootEntity (ME)	2	
RootObject (AMO)	1	
Selects (MR)	1737	0:N **SubtypeSetMembershipCriterion**.-*Selects*.InheritableDataModelObject **1**:1
SemanticInformationObject (ME)	4	
SemanticObjectReference (ME)	10014	
SequenceNumber (MA)	1739	(MR: 0:N AccessPath.*Incorporates*.-Attribute 0:N)

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
SequenceNumber (MA)	1710	(MR: **1**:1 DefinitionObject.*IsConstructedWith*.-**ProjectionComponent** 0:N)
SequenceNumber (MA)	1735	(MR: 0:N Key.*Incorporates*.-SemanticInformationObject 0:N)
SequenceNumber (MA)	1723	(MR: 0:N **ProjectionComponent**.-*IsProjectionOf*.Attribute **1**:N)
SequenceNumber (MA)	6040	(MR: 0:N **ReferencedElement**.*DefinesPath*.-ComponentObject **1**:N)
SequenceNumber (MA)	10114	(MR: **1**:1 Edge.*ConsistsOf*.**EdgeElement** 0:N)
SequenceNumber (MA)	10117	(MR: **1**:1 Annotation.*Uses*.**Annotation-Argument** 0:N)
SequenceNumber (MA)	8026	(MR: 0:1 DefinitionObject.*Contains*.-ComponentObject 0:N)
Size (MA)	247	(ME: StoreDefinition)
SizeUnits (MA)	248	(ME: StoreDefinition)
SpecificationLanguage (MA)	36	(ME: ProcessObject)
SpecificationLanguage (MA)	1078	(ME: **Key**)
SpecificationLanguage (MA)	1717	(MR: **1**:N InheritableDataModelObject.-*IsSubtypeIn*.**SubtypeSet** 0:N)
SpecificationLanguage (MA)	39	(ME: ProjectedAttribute)
SpecificationLanguage (MA)	1075	(ME: **SubtypeSetMembershipCriterion**)
SpecificationLanguage (MA)	1038	(ME: **ProjectionComponent**)
SpecificationLanguage (MA)	1141	(ME: DefinitionObject)
SpecificationLanguage (MA)	1082	(ME: **AccessPath**)
SpecificationText (MA)	35	(ME: ProcessObject)
SpecificationText (MA)	1140b[827]	(ME: **Key**)
SpecificationText (MA)	1140	(ME: DefinitionObject)
SpecificationText (MA)	1718	(MR: **1**:N InheritableDataModelObject.-*IsSubtypeIn*.**SubtypeSet** 0:N)
SpecificationText (MA)	1076	(ME: **SubtypeSetMembershipCriterion**)
SpecificationText (MA)	1039	(ME: **ProjectionComponent**)
SpecificationText (MA)	38	(ME: ProjectedAttribute)
SpecificationText (MA)	1083	(ME: **AccessPath**)
Specifies (MR)	1736b[827]	**1**:1 SubtypeSet.*Specifies*.**SubtypeSet-MembershipCriterion** 0:N
Store (ME)	6028	
StoreDefinition (ME)	239	
StoreWithSupertype (MA)	1736	(MR: **1**:N InheritableDataModelObject.-*IsSubtypeIn*.**SubtypeSet** 0:N)
SubtypeListIsClosed (MA)	1073	(ME: **SubtypeSet**)
SubtypeSet (ME)	1070	

827 Dieses Meta-Objekt weist ursprünglich einen bereits vergebenen Surrogatwert auf. Um den Wert eindeutig zu machen, wurde im Rahmen dieser Arbeit dem zweiten Auftreten des Surrogatwertes ein kleines „b“ angefügt.

Name	CDIF-Meta-Identifier	Vollqualifizierter Name (für MR) beziehungsweise Name des AttribuierbarenMetaObjekts, zu dem das angegebene MA gehört
SubtypeSetMembershipCriterion (ME)	1074	
SupportPort (ME)	6010	
SystemName (MA)	58	(ME: ToolUser)
TextualConstraint (ME)	51	
TimeCreated (MA)	8	(AMO: RootObject)
TimeUpdated (MA)	9	(AMO: RootObject)
ToolUser (ME)	56	
Transform11 (MA)	10131	(ME: **RelativePoint**)
Transform12 (MA)	10132	(ME: **RelativePoint**)
Transform13 (MA)	10133	(ME: **RelativePoint**)
Transform21 (MA)	10134	(ME: **RelativePoint**)
Transform22 (MA)	10135	(ME: **RelativePoint**)
Transform23 (MA)	10136	(ME: **RelativePoint**)
Transform31 (MA)	10137	(ME: **RelativePoint**)
Transform32 (MA)	10138	(ME: **RelativePoint**)
Transform33 (MA)	10139	(ME: **RelativePoint**)
Type (MA)	212	(ME: DataFlowModel)
Unit (MA)	10109	(ME: Diagram)
UpdateEffect (MA)	1068	(ME: **RolePlayer**)
UpdateTimePeriod (MA)	1030	(ME: Entity)
UpdateTimePeriod (MA)	1069	(ME: **RolePlayer**)
Usage (MA)	1031	(ME: Entity)
UsedIn (MR)	74	**1**:N SemanticInformationObject.*UsedIn*.-**Derivation** 0:N
Uses (MR)	10059	**1**:1 Annotation.*Uses*.**Annotation-Argument** 0:N
Uses (MR)	71	0:N RootEntity.*Uses*.AlternateName 0:1

Tabelle 6-2: Standardisierte Meta-Objekte sortiert nach dem Meta-Meta-Attribut „Name“

6.1.3 Querverweistabelle der standardisierten AttribuierbarenMetaObjekte gemeinsam mit ihren Meta-Attributen, sortiert nach dem vollqualifizierten Namen

Die folgende Tabelle 6-3 listet sämtliche standardisierten AttribuierbarenMeta-Objekte[828] des Integrierten EIA/CDIF-Meta-Modells, sortiert nach dem vollqualifizierten Namen, der in der ersten Spalte zu finden ist, gemeinsam mit der Abkürzung seines Metatyps in Klammern und dem Surrogatwert des Meta-Meta-Attributs „CDIFMetaIdentifier“ in EIA/CDIF-kodierter Form[829]. In derselben Zeile wird in der letzten Spalte die Kurzbezeichnung des GegenstandsBereiches angeführt, in dem das AttribuierbareMetaObjekt benutzt wird.[830]

In der zweiten Spalte wird der Surrogatwert des Meta-Meta-Attributs „CDIF-MetaIdentifier“ angeführt. Für Meta-Attribute wird in der dritten Spalte in Klammern eingeschlossen das AttribuierbareMetaObjekt gezeigt, für das es definiert wurde. Für Meta-Beziehungstypen wird in der dritten Spalte der vollqualifizierte Namen angegeben, wobei auch die Kardinalitäten zur Verbesserung des Referenzcharakters der Tabelle mit angeführt sind.

Sofern ein AttribuierbaresMetaObjekt über Meta-Attribute verfügt, werden sie in alphabetisch aufsteigender Form unmittelbar daran angefügt gezeigt, wobei wie bei der Darstellung im Haupttext dieser Arbeit, auch der Datentyp, die Wertemenge (englisch: „Domain“) sowie die Optionalität angegeben wird.

Wie im Rahmen dieser Arbeit üblich werden Meta-Entitätstypen, die zwingend in Meta-Beziehungstypen des Integrierten EIA/CDIF-Meta-Modells enthalten sein müssen, fett dargestellt. Ebenso wird im entsprechenden Meta-Beziehungstyp zusätzlich auch der entsprechende minimale Kardinalitätswert

828 Es fehlen daher in dieser Darstellung die MetaObjekte vom Typ „SubjectArea“.

829 Die „ENCODING.1“-Kodierung für Werte vom EIA/CDIF-Datentyp „Identifier“ erfolgt einfach dadurch, daß der Surrogatwert in Sternen (‘*’) eingeschlossen wird. Entsprechende Werte werden mit einer kleineren Schriftauszeichnung dargestellt, wie dies auch im Haupttext dieser Arbeit der Fall ist.

830 Sofern mehr als ein GegenstandsBereich das AttribuierbareMetaObjekt benutzt, werden alle entsprechenden Kurzbezeichnungen angeführt, wobei die einzelnen Werte durch ein Komma voneinander abgetrennt werden.

fett hervorgehoben. Desgleichen werden die Namen von zwingend vorgeschriebenen Meta-Attributen fett gesetzt, sodaß im Umkehrschluß, mit normaler Schrift gezeigte Meta-Attribute optional sind.[831]

Vollqualifizierter Name des AttribuierbarenMetaObjekts Meta-Attributname	Gegenstands-Bereich
AbsolutePoint (ME) – *10009*	PLAC
Position *10101* – Point	
AbstractionLevel (ME) – *12*	CMMN
Name *13* – Enumerated {Conceptual, Logical, Physical}	
AccessPath (ME) – *1001*	DMOD
Name *1081* – String(256)	
SpecificationLanguage *1082* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
SpecificationText *1083* – Text	
0:N AccessPath.*Incorporates*.Attribute 0:N (MR) – *1716*	DMOD
IsAscending *1738* – Boolean	
SequenceNumber *1739* – Integer	
0:1 AccessPath.*Instantiates*.Key 0:1 (MR) – *1700*	DMOD
AlternateName (ME) – *14*	CMMN
OtherLongName *15* – String(1024)	
OtherName *16* – String(256)	
Annotation (ME) – *10010*	PLAC
DerivationLanguage *10102* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
DerivationText *10103* – Text	
1:1 Annotation.*Uses*.**AnnotationArgument** 0:N (MR) – *10059*	PLAC
SequenceNumber *10117* – Integer	
AnnotationArgument (ME) – *10013*	PLAC
DataBlock *10104* – Text	
MIMESubtype *10119* – String(32)	
MIMEType *10118* – String(32)	
Attribute (ME) – *17*	DMOD, DFM
DefaultValue *18* – String(1024)	
IsOptional *19* – Boolean	
Name *21* – String(256)	
0:N Attribute.*IsDiscriminatorFor*.SubtypeSetMembershipCriterion 0:N (MR) – *1701*	DMOD
0:N Attribute.*IsInheritedFrom*.Attribute 0:1 (MR) – *1702*	DMOD
CandidateKey (ME) – *1003*	DMOD
IsPrimary *1004* – Boolean	

831 Wie aus den obigen Erläuterungen hervorgeht, wurde versucht, diese Tabelle so kompakt wie möglich zu gestalten und gleichzeitig sämtliche relevanten Definitionen aus dem Integrierten EIA/CDIF-Meta-Modell darin aufzunehmen.

Vollqualifizierter Name des AttribuierbarenMetaObjekts		Gegenstands-Bereich
	Meta-Attributname	
0:N	CandidateKey.*Incorporates*.ForeignKey 0:N (MR) – *1713*	DMOD
	Cluster (ME) – *1005*	DMOD
0:N	Cluster.*Collects*.DataModelObject 0:N (MR) – *1703*	DMOD
	ComponentObject (ME) – *8000*	DMOD, DFM
0:N	ComponentObject.*References*.DefinitionObject 0:1 (MR) – *8022*	DFM
	ConstraintPort (ME) – *6012*	DFM
	ControlPort (ME) – *6014*	DFM
	DFMProcess (ME) – *6005*	DFM
	ContextDescription *1122* – Text	
	ContextIdentifier *1123* – String(32)	
	Name *1150* – String(256)	
	DFMProcessDefinition (ME) – *213*	DFM
	FunctionalArea *214* – String(256)	
	HasConcurrentChildren *215* – Boolean	
	HasRealTimeSemantics *1130* – Boolean	
	IsControl *216* – Boolean	
	IsData *217* – Boolean	
	IsMaterial *218* – Boolean	
	DataFlowModel (ME) – *210*	DFM
	Name *211* – String(256)	
	Type *212* – Enumerated {Current, Required}	
0:1	DataFlowModel.*HasRoot*.DFMProcessDefinition 0:1 (MR) – *6026*	DFM
	DataModel (ME) – *1008*	DMOD
	ModelType *1010* – String(64)	
	Name *1011* – String(256)	
0:N	DataModel.*Collects*.DataModelObject 0:N (MR) – *1704*	DMOD
	DataModelObject (ME) – *1012*	DMOD
1:1	DataModelObject.*ActsAs*.**RolePlayer** 0:N (MR) – *1705*	DMOD
0:N	DataModelObject.*IsMemberOf*.DataModelSubset 0:N (MR) – *1706*	DMOD
	DataModelSubset (ME) – *1013*	DMOD
	Name *1015* – String(256)	
0:N	DataModelSubset.*Excludes*.Attribute 0:N (MR) – *1707*	DMOD
0:N	**DataModelSubset**.*IsSubsetOf*.DataModel **1**:1 (MR) – *1708*	DMOD
	DataObject (ME) – *22*	CMMN
	Name *23* – String(256)	
	DefinitionObject (ME) – *8002*	DMOD, DFM
	Name *1119* – String(256)	
	Operator *1118* – Enumerated {AND, XOR, OR}	
	SpecificationLanguage *1141* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
	SpecificationText *1140* – Text	
0:1	DefinitionObject.*Contains*.ComponentObject 0:N (MR) – *1131*	DMOD, DFM
	SequenceNumber *8026* – Integer	

Vollqualifizierter Name des AttribuierbarenMetaObjekts		Gegenstands-Bereich
	Meta-Attributname	
1:1	DefinitionObject.*IsConstructedWith*.**ProjectionComponent** 0:N (MR) – *1709*	DMOD
	SequenceNumber *1710* – Integer	
	Derivation (ME) – *26*	CMMN
	DerivationLanguage *29* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
	DerivationText *27* – Text	
	IsRealizationOf *185* – Boolean	
	Diagram (ME) – *10002*	PLAC
	DotsPerUnit *10108* – Float	
	ExtentX *10105* – Float	
	ExtentY *10106* – Float	
	ExtentZ *10107* – Float	
	Unit *10109* – Enumerated {PerMeter, PerCentimeter, PerMillimeter, PerInch, PerMicrometer}	
	Edge (ME) – *10006*	PLAC
1:1	Edge.*ConsistsOf*.**EdgeElement** 0:N (MR) – *10056*	PLAC
	SequenceNumber *10114* – Integer	
0:N	Edge.*IsAttachedTo*.GraphicalElement 0:2 (MR) – *10055*	PLAC
	EdgeElement (ME) – *10007*	PLAC
0:N	**EdgeElement**.*HasEnd*.Point **2**:2 (MR) – *10057*	PLAC
	Entity (ME) – *1016*	DMOD
	AvgNumberOfOccurrences *1017* – Float	
	DeletionTimePeriod *1018* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	EntityType *1019* – Enumerated {Kernel, Characteristic, Associative}	
	InsertionTimePeriod *1020* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	MaxNumberOfOccurrences *1022* – Integer	
	MinNumberOfOccurrences *1023* – Integer	
	NormalizationState *1024* – Enumerated {UNF, 1NF, 2NF, 3NF, BCNF, 4NF, 5NF}	
	NumberOfDeletions *1025* – Float	
	NumberOfInsertions *1026* – Float	
	NumberOfReads *1027* – Float	
	NumberOfUpdates *1028* – Float	
	ReadTimePeriod *1029* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	UpdateTimePeriod *1030* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	Usage *1031* – Text	
1:1	Entity.*IsAccessedUsing*.**AccessPath** 0:N (MR) – *1712*	DMOD
1:1	Entity.*IsIdentifiedBy*.**Key** 0:N (MR) – *1711*	DMOD
	EquivalenceSet (ME) – *1113*	DFM
0:N	**EquivalenceSet**.*HasMember*.ComponentObject **2**:N (MR) – *1114*	DFM
	ExternalAgent (ME) – *6004*	DFM
	ContextDescription *1124* – Text	

Vollqualifizierter Name des AttribuierbarenMetaObjekts			Gegenstands-Bereich
		Meta-Attributname	
		ContextIdentifier *1125* – String(32)	
		Name *1151* – String(256)	
	ExternalAgentDefinition (ME) – *6002*		DFM
	Flow (ME) – *227*		DFM
		ContextDescription *228* – Text	
		ContextIdentifier *6041* – String(32)	
		Name *231* – String(256)	
	FlowDefinition (ME) – *221*		DFM
		IsAnalog *1142* – Boolean	
		IsControl *223* – Boolean	
		IsData *224* – Boolean	
		IsMaterial *225* – Boolean	
	FlowInputPort (ME) – *6001*		DFM
	FlowOutputPort (ME) – *6000*		DFM
	FlowPort (ME) – *6015*		DFM
	FlowProducerConsumer (ME) – *232*		DFM
0:N	FlowProducerConsumer.*Consumes*.Flow 0:N (MR) – *6024*		DFM
0:N	FlowProducerConsumer.*Produces*.Flow 0:N (MR) – *6025*		DFM
0:N	FlowProducerConsumer.*ProducesOrConsumes*.Flow 0:N (MR) – *6023*		DFM
	ForeignKey (ME) – *1032*		DMOD
0:1	ForeignKey.*Incorporates*.RolePlayer 0:1 (MR) – *1728*		DMOD
0:N	**ForeignKey**.*References*.CandidateKey **1**:1 (MR) – *1714*		DMOD
	GraphicalElement (ME) – *10004*		PLAC
0:N	**GraphicalElement**.*AppearsOn*.Diagram **1**:1 (MR) – *10052*		PLAC
	InheritableDataModelObject (ME) – *1033*		DMOD
		IsAbstract *1034* – Boolean	
1:N	InheritableDataModelObject.*IsSubtypeIn*.**SubtypeSet** 0:N (MR) – *1715*		DMOD
		SpecificationLanguage *1717* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
		SpecificationText *1718* – Text	
		StoreWithSupertype *1736* – Boolean	
1:1	InheritableDataModelObject.*IsSupertypeFor*.**SubtypeSet** 0:N (MR) – *1719*		DMOD
	Key (ME) – *1035*		DMOD
		Name *1077* – String(256)	
		SpecificationLanguage *1078* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
		SpecificationText *1140b*[832] – Text	
0:N	Key.*Incorporates*.Attribute 0:N (MR) – *1733*		DMOD
		IsAscending *1734* – Boolean	

832 Dieses Meta-Objekt weist einen bereits vergebenen Surrogatwert auf. Um den Wert eindeutig zu machen, wurde im Rahmen dieser Arbeit dem zweiten Auftreten des Surrogatwertes ein kleines „b“ angefügt.

Vollqualifizierter Name des AttribuierbarenMetaObjekts		Gegenstands-Bereich
	Meta-Attributname	
0:N	Key.*Incorporates*.SemanticInformationObject 0:N (MR) – *1733b*[832]	DMOD
	SequenceNumber *1735* – Integer	
	Node (ME) – *10011*	PLAC
	Point (ME) – *10003*	PLAC
0:N	Point.*IsDependentUpon*.PositionedElement 0:1 (MR) – *10060*	PLAC
0:N	**Point**.*IsLocatedOn*.Diagram **1**:1 (MR) – *10053*	PLAC
	Port (ME) – *1129*	DFM
	IsFormal *1139* – Boolean	
	Name *1152* – String(256)	
	PositionedElement (ME) – *10005*	PLAC
	ExtentX *10121* – Float	
	ExtentY *10122* – Float	
	ExtentZ *10123* – Float	
0:N	**PositionedElement**.*HasCenter*.Point **1**:1 (MR) – *10054*	PLAC
	PresentationInformationObject (ME) – *30*	CMMN, PLAC
0:N	PresentationInformationObject.*Represents*.SemanticObjectReference 0:N (MR) – *10051*	PLAC
	ProcessObject (ME) – *31*	CMMN
	ExecutionTimeInterval *34* – Float	
	ExecutionTimeUnit *32* – Enumerated {Picosecond, Nanosecond, Microsecond, Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	Name *33* – String(256)	
	SpecificationLanguage *36* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
	SpecificationText *35* – Text	
	ProjectedAttribute (ME) – *37*	DMOD
	SpecificationLanguage *39* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
	SpecificationText *38* – Text	
0:N	ProjectedAttribute.*IsProjectionOf*.Attribute 0:N (MR) – *63*	DMOD
	ProjectionComponent (ME) – *1036*	DMOD
	Name *1037* – String(256)	
	SpecificationLanguage *1038* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
	SpecificationText *1039* – Text	
0:N	**ProjectionComponent**.*IsFullProjectionOf*.DefinitionObject **1**:1 (MR) – *1721*	DMOD
0:N	**ProjectionComponent**.*IsProjectionOf*.Attribute **1**:N (MR) – *1722*	DMOD
	SequenceNumber *1723* – Integer	
	ReferencedElement (ME) – *8020*	DFM
0:N	**ReferencedElement**.*DefinesPath*.ComponentObject **1**:N (MR) – *8025*	DFM
	SequenceNumber *6040* – Integer	
	Relationship (ME) – *1040*	DMOD
	InverseName *1041* – String(256)	

Vollqualifizierter Name des AttribuierbarenMetaObjekts		Gegenstands-Bereich
	RelativePoint (ME) – *10008*	PLAC
	DeltaX *10124* – Float	
	DeltaY *10125* – Float	
	DeltaZ *10126* – Float	
	Transform11 *10131* – Float	
	Transform12 *10132* – Float	
	Transform13 *10133* – Float	
	Transform21 *10134* – Float	
	Transform22 *10135* – Float	
	Transform23 *10136* – Float	
	Transform31 *10137* – Float	
	Transform32 *10138* – Float	
	Transform33 *10139* – Float	
0:N	**RelativePoint**.*IsRelativeTo*.Point **1**:1 (MR) – *10058*	PLAC
	Role (ME) – *1042*	DMOD
	IsMaster *1043* – Boolean	
	IsSource *1044* – Boolean	
2:N	**Role**.*BelongsTo*.**Relationship 1**:1 (MR) – *1724*	DMOD
	RoleConstraint (ME) – *1045*	DMOD
	Name *1046* – String(256)	
	Operator *1047* – Enumerated {AND, OR, XOR}	
0:N	RoleConstraint.*Incorporates*.RoleConstraint 0:N (MR) – *1725*	DMOD
0:N	RoleConstraint.*Incorporates*.RolePlayer 0:N (MR) – *1726*	DMOD
0:N	RoleConstraint.*Incorporates*.SemanticInformationObject 0:N (MR) – *1727*	DMOD
	RolePlayer (ME) – *1048*	DMOD
	AvgNumberOfOccurrences *1049* – Float	
	DeleteEffect *1050* – Enumerated {RESTRICTS, CASCADES, SETNULL, SETDEFAULT}	
	DeletionTimePeriod *1051* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	InsertEffect *1052* – Enumerated {RESTRICTS, CASCADES, SETNULL, SETDEFAULT}	
	InsertionTimePeriod *1053* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	IsDeleteDeferrable *1054* – Boolean	
	IsInsertDeferrable *1055* – Boolean	
	IsUpdateDeferrable *1056* – Boolean	
	MaxInnerCardinality *1057* – String(10)	
	MaxNumberOfOccurrences *1058* – Integer	
	MaxOuterCardinality *1059* – String(10)	
	MinInnerCardinality *1060* – String(10)	
	MinNumberOfOccurrences *1061* – Integer	
	MinOuterCardinality *1062* – String(10)	
	NumberOfDeletions *1063* – Float	
	NumberOfInsertions *1064* – Float	
	NumberOfReads *1065* – Float	

Vollqualifizierter Name des AttribuierbarenMetaObjekts		Gegenstands-Bereich
	Meta-Attributname	
	NumberOfUpdates *1066* – Float	
	ReadTimePeriod *1067* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
	UpdateEffect *1068* – Enumerated {RESTRICTS, CASCADES, SETNULL, SETDEFAULT}	
	UpdateTimePeriod *1069* – Enumerated {Millisecond, Second, Minute, Hour, Day, Week, Month, Year}	
0:N	RolePlayer.*IsSupportedBy*.Key 0:1 (MR) – *1729*	DMOD
1:N	RolePlayer.*Plays*.**Role** 0:1 (MR) – *1730*	DMOD
0:N	RolePlayer.*Refines*.RolePlayer 0:1 (MR) – *1731*	DMOD
0:1	RolePlayer.*RefinesForSubtype*.DataModelObject 0:N (MR) – *1732*	DMOD
	RootEntity (ME) – *2*	FND, CMMN
0:N	RootEntity.*CreatedBy*.ToolUser 0:1 (MR) – *68*	CMMN
1:1	RootEntity.*Has*.**AlternateName** 0:N (MR) – *69*	CMMN
0:N	RootEntity.*IsRelatedTo*.RootEntity 0:N (MR) – *3*	FND
0:N	RootEntity.*LastUpdatedBy*.ToolUser 0:1 (MR) – *70*	CMMN
0:N	RootEntity.*Uses*.AlternateName 0:1 (MR) – *71*	CMMN
	RootObject (AMO) – *1*	FND
	CDIFIdentifier *5* – Identifier	
	DateCreated *6* – Date	
	DateUpdated *7* – Date	
	TimeCreated *8* – Time	
	TimeUpdated *9* – Time	
	SemanticInformationObject (ME) – *4*	CMMN, DMOD
	BriefDescription *44* – String(1024)	
	FullDescription *45* – Text	
0:N	SemanticInformationObject.*IsCategorizedIn*.AbstractionLevel 0:N (MR) – *72*	CMMN
1:N	SemanticInformationObject.*ProducedBy*.**Derivation** 0:N (MR) – *73*	CMMN
1:N	SemanticInformationObject.*UsedIn*.**Derivation** 0:N (MR) – *74*	CMMN
	SemanticObjectReference (ME) – *10014*	PLAC
	ReferencedMetaAttributeName *10111* – Identifier	
	ReferencedMetaObjectInstance *10110* – Identifier	
	Store (ME) – *6028*	DFM
	ContextDescription *1120* – Text	
	ContextIdentifier *1121* – String(32)	
	Name *1153* – String(256)	
	StoreDefinition (ME) – *239*	DFM
	IsControl *240* – Boolean	
	IsData *241* – Boolean	
	IsManual *242* – Boolean	
	IsMaterial *243* – Boolean	
	IsPermanent *244* – Boolean	
	IsReadOnly *245* – Boolean	

Vollqualifizierter Name des AttribuierbarenMetaObjekts			Gegenstands-Bereich
		Meta-Attributname	
		Size *247* – Float	
		SizeUnits *248* – Enumerated {Bit, Byte, Kilobyte, Megabyte, Gigabyte, Terabyte}	
	SubtypeSet (ME) – *1070*		DMOD
		IsExclusive *1071* – Boolean	
		Name *1072* – String(256)	
		SubtypeListIsClosed *1073* – Boolean	
1:1	SubtypeSet.*Specifies*.**SubtypeSetMembershipCriterion** 0:N (MR) – *1736b*[833]		DMOD
	SubtypeSetMembershipCriterion (ME) – *1074*		DMOD
		DiscriminatorValue *1080* – String(256)	
		SpecificationLanguage *1075* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
		SpecificationText *1076* – Text	
0:N	**SubtypeSetMembershipCriterion**.*Selects*.InheritableDataModelObject **1**:1 (MR) – *1737*		DMOD
	SupportPort (ME) – *6010*		DFM
	TextualConstraint (ME) – *51*		CMMN
		BriefDescription *52* – String(1024)	
		ConstraintExpression *53* – Text	
		ConstraintLanguage *55* – Enumerated {Ada, C, COBOL, FORTRAN, MUMPS, Pascal, PL1, SQL, Other}	
		FullDescription *54* – Text	
0:N	**TextualConstraint**.*IsConstraintOn*.SemanticInformationObject **1**:N (MR) – *80*		CMMN
	ToolUser (ME) – *56*		CMMN
		FullName *57* – String(256)	
		SystemName *58* – String(32)	

Tabelle 6-3: Standardisierte AttribuierbareMetaObjekte gemeinsam mit ihren Meta-Attributen, sortiert nach dem vollqualifizierten Namen

833 Dieses Meta-Objekt weist ursprünglich einen bereits vergebenen Surrogatwert auf. Um den Wert eindeutig zu machen, wurde im Rahmen dieser Arbeit dem zweiten Auftreten des Surrogatwertes ein kleines „b“ angefügt.

6.2 Implementierungsbeispiele

In diesem Abschnitt erfolgen beispielhafte Implementierungen aufgrund der in Abschnitt 3.3, „Abbildungen des EIA/CDIF-Meta-Meta-Modells“ auf Seite 106ff festgelegten Spezifikationen für die entsprechende Abbildungen für das EIA/CDIF-Meta-Meta-Modell. Die folgende Abbildung 6-1 stellt das EIA/CDIF-Meta-Meta-Modell dar, wobei es im Unterschied zu den Abbildungen im Hauptteil auch die deutschen Übersetzungen für die Meta-Meta-Entitätstypen und Meta-Meta-Beziehungstypen in Klammern anführt.

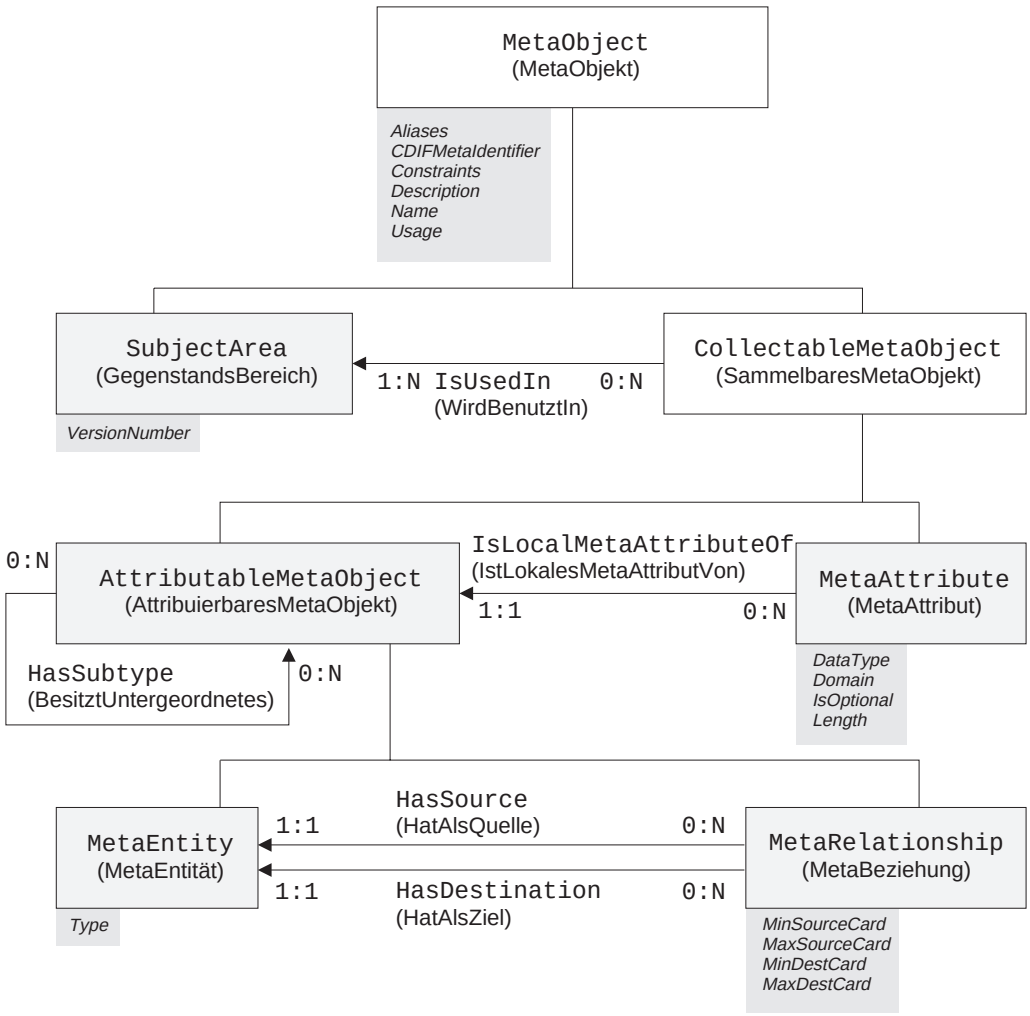


Abbildung 6-1: Das EIA/CDIF-Meta-Meta-Modell (mit deutschen Übersetzungen)

In weiterer Folge wird gezeigt, wie eine Implementierung im relationalen Datenbankverwaltungssystem ORACLE 7.3.2.3.0 erfolgen kann. Es wird hier besonders darauf wert gelegt, daß Löschanweisungen *sämtliche* Datensätze, die gemeinsam das entsprechende Meta-Objekt repräsentieren, aus den Implementierungstabellen entfernt werden. Abschließend wird exemplarisch gezeigt,

wie die nunmehr über eine relationale Datenbank zugänglichen Standarddefinitionen der EIA/CDIF-Meta-Objekte mit Hilfe von SQL auswertbar sind.

Daran anschließend wird eine Implementierung in der objektorientierten Programmiersprache Object Rexx vorgestellt, mit deren Hilfe Meta-Modelldaten aus einem im Rahmen dieser Arbeit definierten und in diesem Abschnitt dokumentierten Format in die implementierten Tabellen übertragen werden. Hierbei müssen die unterschiedlichen Meta-Meta-Attributwerte auf die entsprechenden, das EIA/CDIF-Meta-Meta-Modell repräsentierenden Tabellen, aufgeteilt werden. Dies erfolgt einfach, indem die entsprechenden, das EIA/CDIF-Meta-Meta-Modell repräsentierenden Object Rexx Klassen das Beschicken der relationalen Tabellen selbst übernehmen.

6.2.1 Implementierung des EIA/CDIF-Meta-Meta-Modells in ORACLE

Die Implementierung in diesem Abschnitt erfolgt auf Basis der Spezifikationen eines Tabellen- und VIEW-Systems zur Repräsentation des EIA/CDIF-Meta-Meta-Modells, wie sie in dieser Arbeit im Abschnitt 3.3.1, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf relationale Datenbankverwaltungssysteme“ auf Seite 108ff, erfolgt ist.

Als relationale Datenbank wird ORACLE in der Version 7.3.2.3.0 eingesetzt, als Programmiersprache für das Ausprogrammieren von gespeicherten Prozeduren[834], die durch definierte Trigger angestoßen werden, kommt die proprietäre, prozederale Erweiterung zu SQL, PL[835]/SQL[836] in der Version 2.3.2.3.0, zum Einsatz.

Zunächst werden die notwendigen Datenbankrechte in Form einer ORACLE-Rolle definiert, daran anschließend die ORACLE-Definitionen für den Aufbau der Tabellen, Sichtweisen (VIEW), sowie die Festlegungen für Trigger und gespeicherte Prozeduren angegeben. Der Abschnitt wird durch einige ORACLE-SQL-Beispiele abgeschlossen, die zeigen, wie man unter Kenntnis

834 Im Englischen spricht man von „stored procedures“.

835 Das Akronym „PL“ steht für die englische Bezeichnung „procedural language“ (deutsch: „prozedurale Sprache“) und weist auf die prozeduralen Erweiterungen zu SQL hin.

836 Vgl. für die PL/SQL-Programmierung vor allem [Urm96], aber auch die Ausführungen über SQL beispielsweise in [Date87], [DatDar97] sowie in [PanTau98].

der Datenbankstrukturen und der EIA/CDIF-Standards, Analysen an den gespeicherten EIA/CDIF-Meta-Modelldefinitionen durchführen kann.

6.2.1.1 Definition der ORACLE-Rolle „CDIF_ROLE“

In diesem Abschnitt wird eine Rolle namens „CDIF_ROLE“ definiert, der dann jene ORACLE-Datenbankrechte zugewiesen werden, die für eine Administration der EIA/CDIF-entsprechenden Tabellen notwendig sind. Über eine Zuweisung dieser Rolle an einen hypothetischen Benutzer „TESTUSER“, erhält dieser Benutzer sämtliche Rechte der Rolle zur Verfügung gestellt.

Die SQL-Anweisungen im folgenden Programmcode 6-1 sind grau hervorgehoben, um sie deutlich von den Kommentaren zu unterscheiden.

```
REM author: Rony G. Flatscher, 97-09-23
REM
REM Use this script to create a default CDIF_ROLE role for
REM a CDIF-repository administrator
REM    Note: Run this script from SQLPLUS as a DBA-user
REM    Note: CREATE CLUSTER and CREATE DATABASE LINK are not mandatory

PROMPT
PROMPT Dropping role CDIF_ROLE, if it exists already
DROP ROLE cdif_role;

PROMPT
PROMPT Creating role CDIF_ROLE ...
CREATE ROLE cdif_role;

PROMPT Assigning privileges to CDIF_ROLE ...
GRANT ALTER  SESSION
,     CREATE ANY SYNONYM
,     CREATE CLUSTER
,     CREATE DATABASE LINK
,     CREATE PROCEDURE
,     CREATE ROLE
,     CREATE SEQUENCE
,     CREATE SESSION
,     CREATE TABLE
,     CREATE TRIGGER
,     CREATE VIEW
,     DROP   ANY SYNONYM
TO    cdif_role;
```

```
PROMPT
PROMPT Creating user TESTUSER identified by TEST
CREATE USER testuser IDENTIFIED BY test
                     DEFAULT   TABLESPACE USERS
                     TEMPORARY TABLESPACE TEMP
                     QUOTA UNLIMITED ON USERS ;

PROMPT
PROMPT Assigning CDIF_ROLE to user TESTUSER
GRANT cdif_role TO TESTUSER;
```

Programmcode 6-1: Definition der Rolle „CDIF_ROLE“ und eines Testbenutzers

6.2.1.2 Definition der Tabellen, Views, Triggers und Stored Procedures

Der folgende Programmcode 6-2 beinhaltet sämtliche ORACLE PL/SQL-Anweisungen, mit deren Hilfe die Implementierung der notwendigen Datenbankstrukturen des EIA/CDIF-Meta-Meta-Modells entsprechend der in Abschnitt 3.3.1, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf relationale Datenbankverwaltungssysteme“ auf Seite 108ff, angegebenen Spezifikationen möglich ist. Die Anweisungen löschen zunächst eventuell bereits vorhandene Datenbankstrukturen, ehe die neuen angelegt werden.[837]

Für die Ausgabe von Debug-Informationen im Modul[838] „CDIF_Data“ des dargestellten Programmcodes werden Prozeduren aus dem mit der ORACLE-Datenbank mitgelieferten Modul namens „DBMS_OUTPUT“ benutzt. Sollen diese Hinweise sichtbar gemacht werden, muß in der entsprechenden SQL*PLUS[839]-Sitzung folgende Einstellung zuvor vorgenommen werden[840]:

```
SET SERVEROUTPUT ON SIZE 100000
```

837 Dieses Verhalten war während der Entwicklung und Implementierung notwendig und wurde zu Dokumentationszwecken beibehalten.

838 Es handelt sich um ORACLE-„Packages“, die aus einem Spezifikations- und einem Implementierungsteil bestehen und jeweils beliebig viele Prozeduren beinhalten können.

839 „SQL*PLUS“ ist die zeichenorientierte, interaktive Schnittstelle zur ORACLE-Datenbank und kann auch im Stapelbetrieb eingesetzt werden. Es ist möglich, mit einer Reihe von „SET“-Anweisungen unterschiedliche Parameter von SQL*PLUS zu setzen, die in weiterer Folge das Abarbeiten von (PL/)SQL-Anweisungen beziehungsweise die Formatierung von Ergebnissen beinflussen.

840 Damit wird ein 100.000 Zeichen umfassender Puffer für Ausgaben der Prozeduren von dem mit der Datenbank mitgelieferten Modul „DBMS_OUTPUT“ definiert. *Nach* dem Ablauf von SQL-Blöcken wird der Inhalt dieses Puffers aufgrund der Aktivierung von „SERVEROUTPUT“ gezeigt.

Kommentare sind entweder in die Zeichenketten '**/***' und '***/**' eingeschlossen oder werden durch zwei aufeinanderfolgende Bindestriche '**--**' eingeleitet und durch den Zeilenumbruch am Ende der Zeile beendet. Eine PL-SQL-Anweisung wird durch einen zwingend vorgeschriebenen Strichpunkt (';') abgeschlossen. Sofern ein Schrägstrich '/' allein auf einer Zeile steht, so bedeutet dies, daß die davor stehenden (PL/)SQL-Anweisung(en) zu Ende sind und ausgeführt werden sollen.

Nachdem die EIA/CDIF-Meta-Modelldefinitionen entsprechend dem EIA/CDIF-Meta-Meta-Modell aufgeteilt auf verschiedene M3-Tabellen[841] in der relationalen Datenbank abgelegt sind, muß sichergestellt werden, daß das Löschen von Meta-Meta-Objektdefinitionen aus beliebigen M3-Tabellen nicht zu Problemen führt:

- Das Löschen von Datensätzen aus der M3-Tabelle „MO_" ist unproblematisch, da aufgrund der definierten Referenzregeln, der Löschvorgang auf alle Tabellen übertragen wird, die von „MO_" abhängig sind.
- Das Löschen von Datensätzen aus den M3-Tabellen „SA_", „CMO_", „MA_", „AMO_", „ME_" und „MR_" muß dazu führen, daß auch sämtliche entsprechenden Einträge in jenen M3-Tabellen gelöscht werden müssen, die in den direkt und indirekt übergeordneten M3-Tabellen enthalten sind. Dies erfolgt mit Hilfe eines Datenbank-Triggers auf Zeilenebene (englisch: „row level") für jede M3-Tabelle. Hierbei wird eine Prozedur („add_to_delete") aus dem Modul „CDIF_Data" aufgerufen, das der Reihe nach sämtliche Surrogatwerte aus der Spalte „SURR" der zu löschenden Datensätze gemeinsam mit deren Meta-Typ erhält.
  Nachdem die DELETE-Anweisung fertig abgearbeit ist, wird mit Hilfe eines Datenbank-Triggers auf Anweisungsebene (englisch: „statement level") die Prozedur („delete_mo") aus dem Modul „CDIF_Data" aufgerufen, die die einzelnen Datensätze auch aus der M3-Tabelle „MO_" löscht. Aufgrund der definierten Referenzregeln wird der Löschvorgang auf alle Tabellen übertragen, die von „MO_" abhängig sind.

841 Tabellen, die Meta-Meta-Entitätstypen und Meta-Meta-Beziehungstypen des EIA/CDIF-Meta-Meta-Modells repräsentieren, werden im Rahmen dieses Abschnittes auch als „M3-Tabellen" bezeichnet,

Hierbei tritt das Problem von „Mutating Tables“[842] (deutsch: „verändernde Tabellen“) auf, das in diesem Fall darin besteht, daß innerhalb derselben Transaktion in der Ursprungstabelle[843] durch die referentiellen Integritätsbedingungen auch jener Datensatz daraus gelöscht werden soll, der dort zu dem Zeitpunkt nicht mehr existiert. Die Prozeduren des Moduls „CDIF_Data“ nehmen auf dieses Problem Rücksicht und lösen es entsprechend der im folgenden Programmcode 6-2 abgebildeten Programmanweisungen.

Weitere Besonderheiten sind in Form von prägnanten Kommentaren direkt in den Programmcode eingearbeitet, sodaß eine weitere Erläuterung nicht mehr als notwendig erscheint.[844]

Die SQL-Anweisungen im folgenden Programmcode 6-2 sind grau hervorgehoben, um sie deutlich von den Kommentaren zu unterscheiden.

---

842 Dabei handelt es sich um eine bestimmte Form des Phänomens, das durch die referentiellen Integritätsbedingungen bedingt ist. Eine allgemeine Beschreibung und eine Darstellung von weiteren Formen des „Mutating Tables“-Phänomens finden sich beispielsweise in [Urm96] auf Seite 256ff.

843 Es handelt sich um jene Tabelle, aus der Datensätze aufgrund von Benutzer- beziehungsweise Anwendungsaktionen gelöscht wurden.

844 Natürlich sind entsprechende Kenntnisse der ORACLE-Datenbank und von ORACLE-PL/SQL vorausgesetzt. Vgl. im Zusammenhang mit PL/SQL unter anderem auch die Fußnote 836 auf Seite 424 oben.

```
PROMPT OFF
REM cf. EIA/IS-107, p.41ff
REM author: Rony G. Flatscher, 97-09-23, 97-11-13
REM
PROMPT Dropping views and tables representing CDIF's M3-model ...
PROMPT
PROMPT Dropping package CDIF_Data, which ensures OO-behaviour in deletions...

DROP PACKAGE CDIF_Data ;

PROMPT Dropping triggers, ensuring OO-behavour in deletions of table-representations ...
DROP TRIGGER MO_Delete       ;
DROP TRIGGER MO_Delete_Stat  ;
DROP TRIGGER SA_Delete       ;
DROP TRIGGER SA_Delete_Stat  ;
DROP TRIGGER CMO_Delete      ;
DROP TRIGGER CMO_Delete_Stat ;
DROP TRIGGER MA_Delete       ;
DROP TRIGGER MA_Delete_Stat  ;
DROP TRIGGER AMO_Delete      ;
DROP TRIGGER AMO_Delete_Stat ;
DROP TRIGGER MR_Delete       ;
DROP TRIGGER MR_Delete_Stat  ;
DROP TRIGGER ME_Delete       ;
DROP TRIGGER ME_Delete_Stat  ;

PROMPT Dropping tables representing MRs ...
        /* MMR                  */
DROP TABLE IsUsedIn_               ;
DROP TABLE IsLocalMetaAttributeOf_ ;
DROP TABLE HasSubtype_             ;
DROP TABLE HasSource_              ;
DROP TABLE HasDestination_         ;

PROMPT Dropping Views ...
        /* drop views           */
DROP View MO                           ;
DROP View SA                           ;
DROP View CMO                          ;
DROP View MA                           ;
DROP View AMO                          ;
DROP View ME                           ;
DROP View MR                           ;
DROP View AMO_MR_ME                    ;
DROP View MR_Has_Source_Destination_ME ;

PROMPT Dropping Tables ...
        /* MME in reverse order (FK --> PK dependencies ... )   */
PROMPT ON
DROP TABLE MR_  ;
DROP TABLE ME_  ;
DROP TABLE AMO_ ;
DROP TABLE MA_  ;
DROP TABLE CMO_ ;
DROP TABLE SA_  ;
DROP TABLE MO_  ;
```

```
PROMPT Creating CDIF-M3 table definitions ...
PROMPT Creating Meta-Meta-Entities ...
REM ------------------------------------------------------------------------------------------
REM     MetaObject
PROMPT Creating MO-M3 table ...
CREATE TABLE MO_ (
            SURR               NUMBER( 6 )      , /* SURROGATE for Primary Key             */
            MO_TYPE            VARCHAR2( 5 )    , /* type of MO_: SA, MA, AMO, MR, ME      */
/* MO  */   ALIASES            VARCHAR2( 1536 ) , /*   in reality: 1,024 Bytes, STRING     */
            CDIFMetaIdentifier VARCHAR2(   40 ) , /* * in reality:    32 Bytes, IDENTIFIER */
            Constraints        VARCHAR2( 2000 ) , /*   in reality: no bounds,   TEXT       */
            Description        LONG             , /* * in reality: no bounds,   TEXT       */
            Name               VARCHAR2(   40 ) , /* * in reality:    32 Bytes, IDENTIFIER */
            LongName           VARCHAR2(  120 ) , /* * in reality: ME | ME.MR.ME  TEXT     */
            Usage              VARCHAR2( 2000 ) , /*   in reality: no bounds,   TEXT       */
                   CONSTRAINT MO_PK PRIMARY KEY ( SURR )
            );
REM datatype LONG needed, because VARCHAR2 in Oracle 7.3 has a limit of 2,000 chars and there
REM are more chars in MO_.Description;
REM ------------------------------------------------------------------------------------------
REM     SubjectArea
PROMPT Creating SA-M3 table ...
CREATE TABLE SA_ (
            SURR               NUMBER( 6 )      , /* SURROGATE for Primary Key             */
/* SA  */   VersionNumber      VARCHAR2(   16 ) , /* * in reality:     8 Bytes, STRING     */
            ShortHand          VARCHAR2( 6 )    , /* CDIF-shorthand while developed        */
            CDIFNumber         VARCHAR2( 20 )   , /* CDIF-Standard Number                  */
                   CONSTRAINT  SA_FK FOREIGN KEY ( SURR ) REFERENCES MO_ ( SURR )
                                      ON DELETE CASCADE
            );
REM ------------------------------------------------------------------------------------------
REM     CollectableMetaObject
PROMPT Creating CMO-M3 table ...
CREATE TABLE CMO_ (
            SURR               NUMBER( 6 )      , /* SURROGATE for Primary Key             */
                   CONSTRAINT CMO_FK FOREIGN KEY ( SURR ) REFERENCES MO_ ( SURR )
                                      ON DELETE CASCADE
            );
REM ------------------------------------------------------------------------------------------
REM     MetaAttribute
PROMPT Creating MA-M3 table ...
CREATE TABLE MA_ (
            SURR               NUMBER( 6 )      , /* SURROGATE for Primary Key             */
/* MA  */   DataType           VARCHAR2(   40 ) , /* * in reality:    32 Bytes, ENUMERATED */
            Domain             VARCHAR2( 2000 ) , /*   in reality: no bounds,   TEXT       */
            IsOptional         VARCHAR2(   10 ) , /* * in reality:     1 Byte,  BOOLEAN    */
            Length             VARCHAR2(   10 ) , /*   in reality:     4 Bytes, INTEGER    */
                   CONSTRAINT  MA_FK FOREIGN KEY ( SURR ) REFERENCES MO_ ( SURR )
                                      ON DELETE CASCADE
            );
REM ------------------------------------------------------------------------------------------
REM     AttributableMetaObject
PROMPT Creating AMO-M3 table ...
CREATE TABLE AMO_ (
            SURR               NUMBER( 6 )      , /* SURROGATE for Primary Key             */
                   CONSTRAINT AMO_FK FOREIGN KEY ( SURR ) REFERENCES MO_ ( SURR )
                                      ON DELETE CASCADE
            );
```

```
REM ------------------------------------------------------------------------------------------
REM     MetaEntity
PROMPT Creating ME-M3 table ...
CREATE TABLE ME_ (
             SURR               NUMBER( 6 )      , /* SURROGATE for Primary Key             */
/* ME  */    Type               VARCHAR2(   40 ) , /* * in reality:    32 Bytes, ENUMERATED  */
                    CONSTRAINT  ME_FK FOREIGN KEY ( SURR ) REFERENCES MO_ ( SURR )
                                      ON DELETE CASCADE
             );
REM ------------------------------------------------------------------------------------------
REM     MetaRelationship
PROMPT Creating MR-M3 table ...
CREATE TABLE MR_ (
             SURR               NUMBER( 6 )      , /* SURROGATE for Primary Key             */
/* MR  */    MinDestCard        VARCHAR2(   10 ) , /* * in reality:    10 Bytes, STRING      */
             MaxDestCard        VARCHAR2(   10 ) , /* * in reality:    10 Bytes, STRING      */
             MinSourceCard      VARCHAR2(   10 ) , /* * in reality:    10 Bytes, STRING      */
             MaxSourceCard      VARCHAR2(   10 ) , /* * in reality:    10 Bytes, STRING      */
                    CONSTRAINT  MR_FK FOREIGN KEY ( SURR ) REFERENCES MO_ ( SURR )
                                      ON DELETE CASCADE
             );
REM ------------------------------------------------------------------------------------------

PROMPT Adding Primary Key definition for all MO_ subtables
ALTER TABLE  SA_  ADD CONSTRAINT  SA_PK PRIMARY KEY ( SURR ) ;
ALTER TABLE CMO_  ADD CONSTRAINT CMO_PK PRIMARY KEY ( SURR ) ;
ALTER TABLE  MA_  ADD CONSTRAINT  MA_PK PRIMARY KEY ( SURR ) ;
ALTER TABLE AMO_  ADD CONSTRAINT AMO_PK PRIMARY KEY ( SURR ) ;
ALTER TABLE  ME_  ADD CONSTRAINT  ME_PK PRIMARY KEY ( SURR ) ;
ALTER TABLE  MR_  ADD CONSTRAINT  MR_PK PRIMARY KEY ( SURR ) ;

PROMPT Creating Meta-Meta-Relationships ...
REM ------------------------------------------------------------------------------------------
REM     IsUsedIn-MR
PROMPT Creating IsUsedIn-M3 table ...
CREATE TABLE IsUsedIn_ (
             SURR                   NUMBER( 6 ) , /* SURROGATE for Primary Key             */
             Source                 NUMBER( 6 ) , /* FK to PK of CMO      */
             Destination            NUMBER( 6 ) , /* FK to PK of SA       */
                    CONSTRAINT IUI_FK1 FOREIGN KEY ( Source ) REFERENCES CMO_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT IUI_FK2 FOREIGN KEY ( Destination ) REFERENCES SA_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT IUI_PK PRIMARY KEY ( SURR )
             );
REM ------------------------------------------------------------------------------------------
REM     IsLocalMetaAttributeOf-MR
PROMPT Creating IsLocalMetaAttributeOf-M3 table ...
CREATE TABLE IsLocalMetaAttributeOf_ (
             SURR                   NUMBER( 6 ) , /* SURROGATE for Primary Key             */
             Source                 NUMBER( 6 ) , /* FK to PK of MA       */
             Destination            NUMBER( 6 ) , /* FK to PK of AMO      */
                    CONSTRAINT ILMAO_FK1 FOREIGN KEY ( Source ) REFERENCES MA_ ( SURR )
                                         ON DELETE CASCADE ,
                    CONSTRAINT ILMAO_FK2 FOREIGN KEY ( Destination ) REFERENCES AMO_ ( SURR )
                                         ON DELETE CASCADE ,
                    CONSTRAINT ILMAO_PK PRIMARY KEY ( SURR )
             );
```

```
REM ------------------------------------------------------------------------------------------
REM     HasSubtype-MR
PROMPT Creating HasSubtype-M3 table ...
CREATE TABLE HasSubtype_ (
             SURR                    NUMBER( 6 ) , /* SURROGATE for Primary Key             */
             Source                  NUMBER( 6 ) , /* FK to PK of AMO      */
             Destination             NUMBER( 6 ) , /* FK to PK of AMO      */
                    CONSTRAINT HST_FK1 FOREIGN KEY ( Source ) REFERENCES AMO_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT HST_FK2 FOREIGN KEY ( Destination ) REFERENCES AMO_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT HST_PK PRIMARY KEY ( SURR )
             );
REM ------------------------------------------------------------------------------------------
REM     HasSource-MR
PROMPT Creating HasSource-M3 table ...
CREATE TABLE HasSource_ (
             SURR                    NUMBER( 6 ) , /* SURROGATE for Primary Key             */
             Source                  NUMBER( 6 ) , /* FK to PK of MR       */
             Destination             NUMBER( 6 ) , /* FK to PK of ME       */
                    CONSTRAINT HS_FK1 FOREIGN KEY ( Source ) REFERENCES MR_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT HS_FK2 FOREIGN KEY ( Destination ) REFERENCES ME_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT HS_PK PRIMARY KEY ( SURR )
             );
REM ------------------------------------------------------------------------------------------
REM     HasSource-MR
PROMPT Creating HasSource-M3 table ...
CREATE TABLE HasDestination_ (
             SURR                    NUMBER( 6 ) , /* SURROGATE for Primary Key             */
             Source                  NUMBER( 6 ) , /* FK to PK of MR       */
             Destination             NUMBER( 6 ) , /* FK to PK of ME       */
                    CONSTRAINT HD_FK1 FOREIGN KEY ( Source ) REFERENCES MR_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT HD_FK2 FOREIGN KEY ( Destination ) REFERENCES ME_ ( SURR )
                                       ON DELETE CASCADE ,
                    CONSTRAINT HD_PK PRIMARY KEY ( SURR )
             );
PROMPT
PROMPT Creating views ...
REM ------------------------------------------------------------------------------------------
REM View MO
PROMPT Creating View MO ...
CREATE OR REPLACE VIEW MO AS
               SELECT * FROM MO_;
REM ------------------------------------------------------------------------------------------
REM View SA
PROMPT Creating View SA ...
CREATE OR REPLACE VIEW SA AS
               SELECT MO_.*, SA_.ShortHand, SA_.CDIFNumber, SA_.VersionNumber
               FROM   MO_, SA_
               WHERE  MO_.SURR = SA_.SURR
               ;
```

```
REM ------------------------------------------------------------------------------------------
REM View CMO
PROMPT Creating View CMO ...

CREATE OR REPLACE VIEW CMO AS                       /* no attributes of its own     */
               SELECT MO_.*
               FROM   MO_, CMO_
               WHERE  MO_.SURR = CMO_.SURR
               ;
REM ------------------------------------------------------------------------------------------
REM View MA
PROMPT Creating View MA ...
CREATE OR REPLACE VIEW MA AS
               SELECT CMO.*, MA_.DataType, MA_.Domain, MA_.IsOptional, MA_.Length
               FROM   CMO, MA_
               WHERE  CMO.SURR = MA_.SURR ;
REM ------------------------------------------------------------------------------------------
REM View AMO
PROMPT Creating View AMO ...
CREATE OR REPLACE VIEW AMO AS                       /* no attributes of its own     */
               SELECT CMO.*
               FROM   CMO, AMO_
               WHERE  CMO.SURR = AMO_.SURR
               ;
REM ------------------------------------------------------------------------------------------
REM View ME
PROMPT Creating View ME ...
CREATE OR REPLACE VIEW ME AS
               SELECT AMO.*, ME_.Type
               FROM   AMO, ME_
               WHERE  AMO.SURR = ME_.SURR
               ;
REM ------------------------------------------------------------------------------------------
REM View MR
PROMPT Creating View MR ...
CREATE OR REPLACE VIEW MR AS
               SELECT AMO.*, MR_.MinSourceCard, MR_.MaxSourceCard,
                             MR_.MinDestCard,   MR_.MaxDestCard
               FROM   AMO, MR_
               WHERE  AMO.SURR = MR_.SURR
               ;
REM ------------------------------------------------------------------------------------------
REM View MR_Has_Source_Destination_ME, needed for CDIF_Data.Delete_MO() among other things ...
PROMPT Creating View MR_Has_Source_Destination_ME ...
CREATE OR REPLACE VIEW MR_Has_Source_Destination_ME
              ( TARGET, SOURCE, DESTINATION )
               AS
               SELECT 'Source', Source, Destination
               FROM   HasSource_
            UNION
               SELECT 'Destination', Source, Destination
               FROM   HasDestination_
               ;
```

```
REM ------------------------------------------------------------------------------------------
REM a view to contain a union of AMO, ME, MR; defining a UNION directly does not work under
REM Oracle 7.3 due to the existence of a LONG column
REM
REM The following works by using outer-joins as implemented by ORACLE

PROMPT Creating View AMO_MR_ME ...

CREATE OR REPLACE VIEW AMO_MR_ME AS
               SELECT AMO.*, MR_.MinSourceCard, MR_.MaxSourceCard,
                             MR_.MinDestCard,   MR_.MaxDestCard,
                             ME_.Type
               FROM   AMO, MR_, ME_
               WHERE  AMO.SURR = MR_.SURR (+) AND AMO.SURR = ME_.SURR ( + )
               ;

/* show how many MRs and MEs there are:
   SELECT COUNT( * ), COUNT( minDestCard ), COUNT( type ) FROM amo_mr_mE ;
*/

REM ------------------------------------------------------------------------------------------
PROMPT Creating package CDIF_Data, ensuring OO-behaviour in deletions of table-representations
...

PROMPT Creating PACKAGE Header  CDIF_Data ...
CREATE OR REPLACE PACKAGE CDIF_Data AS
   -- type definitions
   TYPE t_surr IS    TABLE of MO_.surr%TYPE    INDEX BY BINARY_INTEGER ;
   TYPE t_bDelete    IS TABLE of BOOLEAN       INDEX BY BINARY_INTEGER ;

   -- variable definitions
   v_SURR_array     t_surr ;                    -- table of surrogates to delete from MO_
   v_counter        BINARY_INTEGER := 0 ;       -- counter to contain # of array entries
   v_mo_type        mo_.mo_type%type := NULL ;  -- field to contain very first triggered delete
   v_bDebug         BOOLEAN := TRUE  ;          -- set debugging mode

   v_bRecurse       BOOLEAN := NULL ;           -- recursing deletion of MAs and MRs ?

   -- add a SURR to be deleted from MO_
   PROCEDURE add_to_delete( tmp_surr IN MO_.surr%TYPE, tmp_MO_Type IN MO_.mo_type%TYPE ) ;

   -- delete MO_s
   PROCEDURE delete_mo( tmp_MO_Type IN MO_.mo_type%TYPE  ) ;

   -- cleanup (delete MR's where one ME is missing, delete MA's without AMO )
   PROCEDURE cleanup;

END CDIF_Data ;
/                                          -- end of stored procedure code (specification)
```

```
PROMPT Creating PACKAGE Body    CDIF_Data ...
CREATE OR REPLACE PACKAGE BODY CDIF_Data AS

   PROCEDURE add_to_delete( tmp_surr IN MO_.surr%TYPE, tmp_MO_Type IN MO_.mo_type%TYPE ) IS
   BEGIN
      IF CDIF_Data.v_bDebug = TRUE THEN
         dbms_output.put_line( 'add_to_delete() – tmp_MO_Type: ' || tmp_MO_Type ||
                               ' SURR: ' || to_char( tmp_surr )  ) ;
         dbms_output.put_line( '...       CDIF_Data.v_MO_Type: ' ||
                               NVL( CDIF_Data.v_mo_type , '<nul>' ) ) ;
      END IF ;

      IF CDIF_Data.v_mo_type IS NULL THEN
         CDIF_Data.v_mo_type := tmp_MO_Type ;   -- memorize

         IF CDIF_Data.v_bDebug = TRUE THEN
            dbms_output.put_line( '...       ==> ==>  setting to: ' || tmp_MO_Type ) ;
         END IF ;
      END IF;

      IF CDIF_Data.v_MO_type = 'MO' THEN        -- don't memorize, if MO was directly deleted
         RETURN;                                -- BEFORE DELETE FOR EACH ROW on MO_ !
      END IF;

      IF CDIF_Data.v_mo_type = tmp_MO_Type THEN
         CDIF_Data.v_counter := CDIF_Data.v_counter + 1 ;              -- increase counter
         CDIF_Data.v_surr_array( CDIF_Data.v_counter ) := tmp_surr;
      END IF;
   END;

   PROCEDURE delete_mo( tmp_MO_Type IN MO_.mo_type%TYPE  ) IS
      v_Counter BINARY_INTEGER ;
   BEGIN
      IF tmp_MO_Type <> CDIF_Data.v_mo_type THEN
         RETURN ;       -- don't do anything, not of level initiating deletion
      END IF;

      IF CDIF_Data.v_bDebug = TRUE THEN
         dbms_output.put_line( 'delete_mo() – deleting ' || TO_CHAR( CDIF_Data.v_counter ) ||
                               ' records from mo_ ...' ) ;
      END IF ;

      FOR v_Counter IN 1..CDIF_Data.v_counter LOOP       -- loop over collected rows
          DELETE FROM MO_
          WHERE  SURR = CDIF_Data.v_surr_array( v_Counter ) ;
      END LOOP ;

      IF CDIF_Data.v_bDebug = TRUE THEN
         dbms_output.put_line( 'delete_MO() – resetting v_counter and v_mo_type table ...' ) ;
      END IF;

      CDIF_Data.v_counter    := 0 ;        -- reset counter, i.e. empty array
      CDIF_Data.v_mo_type    := NULL ;

      IF CDIF_Data.v_bRecurse IS NULL THEN -- only do additional clean-up, if not already
                                           -- doing so
         CDIF_Data.v_bRecurse := TRUE ;
         cleanup;               -- make sure, integrity is there, remove illegal MR's and MA's
         CDIF_Data.v_bRecurse := NULL ;
```

```
         IF CDIF_Data.v_bDebug = TRUE THEN
            dbms_output.put_line( 'delete_MO() – CDIF_Data.v_counter: ' ||
                                  TO_CHAR( CDIF_Data.v_counter ) );
         END IF;
      END IF ;
   END;

   PROCEDURE cleanup IS
   BEGIN
      IF CDIF_Data.v_bDebug = TRUE THEN
         dbms_output.put_line( 'cleanup(), delete leftover MR''s and MA''s ...' ) ;
      END IF ;

      -- delete uncollected CMOs
      IF CDIF_Data.v_bDebug = TRUE THEN
         dbms_output.put_line( '            – deleting uncollected CMOs' ) ;
      END IF;

      DELETE FROM CMO_      -- delete CMOs, which are not collected
      WHERE  SURR NOT IN
             (  SELECT   Source
                FROM     IsUsedIn_
             ) ;

      -- delete damaged MRs, i.e. MRs with only 1 ME as Source or Destination (the other
      -- ME got deleted)
      IF CDIF_Data.v_bDebug = TRUE THEN
         dbms_output.put_line( '            – deleting possible left-over MRs' ) ;
      END IF;

      DELETE FROM MO_      -- delete MRs which got one ME (soruce or destination) deleted
      WHERE  SURR IN
             ( SELECT   Source
               FROM     MR_Has_Source_Destination_ME
               GROUP BY Source
               HAVING   MOD( COUNT( Source ), 2 ) = 1
             ) ;

      -- delete leftover MAs
      IF CDIF_Data.v_bDebug = TRUE THEN
         dbms_output.put_line( '            – deleting possible left-over MAs' ) ;
      END IF;

      DELETE FROM MO_      -- delete MAs, which don't have have any AMO target anymore
      WHERE  SURR IN
             (  SELECT   SURR
                FROM     MO_
                WHERE    MO_Type = 'MA' AND
                         SURR NOT IN (SELECT Source FROM IsLocalMetaAttributeOf_ )
             ) ;
   END;

   /* initialization code  */
BEGIN
   CDIF_Data.v_bDebug := TRUE  ;             -- setting of debug-mode
END CDIF_Data;
/                                            -- end of stored procedure code (implementation)
```

```
REM ------------------------------------------------------------------------------------------
PROMPT Creating triggers, ensuring OO-behavour in deletions of table-representations ...
/* ------------------------------------------------------------------------- */
-- TRIGGERs for MO_
PROMPT Creating TRIGGER MO_Delete ...
CREATE OR REPLACE TRIGGER MO_Delete     -- row-level trigger, collect deleted SURR
   BEFORE DELETE ON MO_
   FOR EACH ROW
BEGIN
   CDIF_Data.add_to_delete( :OLD.surr, 'MO' );        -- memorize to delete from mo_
END MO_Delete;
/
PROMPT Creating TRIGGER MO_Delete_Stat ...
CREATE OR REPLACE TRIGGER MO_Delete_Stat        -- statement level trigger, delete from MO_
       AFTER  DELETE ON MO_                     -- (and via cascading all dependent table-rows)
BEGIN
   CDIF_Data.delete_mo( 'MO' ) ;
END MO_Delete_Stat ;
/
/* ------------------------------------------------------------------------- */
-- TRIGGERs for SA_
PROMPT Creating TRIGGER SA_Delete ...
CREATE OR REPLACE TRIGGER SA_Delete     -- row-level trigger, collect deleted SURR
   AFTER DELETE ON SA_
   FOR EACH ROW
BEGIN
   CDIF_Data.add_to_delete( :OLD.surr, 'SA' );        -- memorize to delete from mo_
END SA_Delete;
/
PROMPT Creating TRIGGER SA_Delete_Stat ...
CREATE OR REPLACE TRIGGER SA_Delete_Stat        -- statement level trigger, delete from MO_
       AFTER DELETE ON SA_                      -- (and via cascading all dependent table-rows)
BEGIN
   CDIF_Data.delete_mo( 'SA' ) ;
END SA_Delete_Stat ;
/
/* ------------------------------------------------------------------------- */
-- TRIGGERs for CMO_
PROMPT Creating TRIGGER CMO_Delete ...
CREATE OR REPLACE TRIGGER CMO_Delete     -- row-level trigger, collect deleted SURR
   AFTER DELETE ON CMO_
   FOR EACH ROW
BEGIN
   CDIF_Data.add_to_delete( :OLD.surr, 'CMO' );        -- memorize to delete from mo_
END CMO_Delete;
/
PROMPT Creating TRIGGER CMO_Delete_Stat ...
CREATE OR REPLACE TRIGGER CMO_Delete_Stat        -- statement level trigger, delete from MO_
       AFTER DELETE ON CMO_                      -- (and via cascading all dependent table-rows)
BEGIN
   CDIF_Data.delete_mo( 'CMO' ) ;
END CMO_Delete_Stat ;
/
```

```
/* ---------------------------------------------------------------------------- */
-- TRIGGERs for MA_
PROMPT Creating TRIGGER MA_Delete ...
CREATE OR REPLACE TRIGGER MA_Delete     -- row-level trigger, collect deleted SURR
   AFTER DELETE ON MA_
   FOR EACH ROW

BEGIN
   CDIF_Data.add_to_delete( :OLD.surr, 'MA' );        -- memorize to delete from mo_
END MA_Delete;
/
PROMPT Creating TRIGGER MA_Delete_Stat ...
CREATE OR REPLACE TRIGGER MA_Delete_Stat        -- statement level trigger, delete from MO_
       AFTER DELETE ON MA_                      -- (and via cascading all dependent table-rows)
BEGIN
   CDIF_Data.delete_mo( 'MA' ) ;
END MA_Delete_Stat ;
/
/* ---------------------------------------------------------------------------- */
-- TRIGGERs for AMO_
PROMPT Creating TRIGGER AMO_Delete ...
CREATE OR REPLACE TRIGGER AMO_Delete     -- row-level trigger, collect deleted SURR
   AFTER DELETE ON AMO_
   FOR EACH ROW
BEGIN
   CDIF_Data.add_to_delete( :OLD.surr, 'AMO' );        -- memorize to delete from mo_
END AMO_Delete;
/
PROMPT Creating TRIGGER AMO_Delete_Stat ...
CREATE OR REPLACE TRIGGER AMO_Delete_Stat        -- statement level trigger, delete from MO_
       AFTER DELETE ON AMO_                      -- (and via cascading all dependent table-rows)
BEGIN
   CDIF_Data.delete_mo( 'AMO' ) ;
END AMO_Delete_Stat ;
/
/* ---------------------------------------------------------------------------- */
-- TRIGGERs for MR_
PROMPT Creating TRIGGER MR_Delete ...
CREATE OR REPLACE TRIGGER MR_Delete     -- row-level trigger, collect deleted SURR
   AFTER DELETE ON MR_
   FOR EACH ROW
BEGIN
   CDIF_Data.add_to_delete( :OLD.surr, 'MR' );        -- memorize to delete from mo_
END MR_Delete;
/
PROMPT Creating TRIGGER MR_Delete_Stat ...
CREATE OR REPLACE TRIGGER MR_Delete_Stat        -- statement level trigger, delete from MO_
       AFTER DELETE ON MR_                      -- (and via cascading all dependent table-rows)
BEGIN
   CDIF_Data.delete_mo( 'MR' ) ;
END MR_Delete_Stat ;
/
/* ---------------------------------------------------------------------------- */
-- TRIGGERs for ME_
PROMPT Creating TRIGGER ME_Delete ...
CREATE OR REPLACE TRIGGER ME_Delete     -- row-level trigger, collect deleted SURR
   AFTER DELETE ON ME_
   FOR EACH ROW
BEGIN
   CDIF_Data.add_to_delete( :OLD.surr, 'ME' );        -- memorize to delete from mo_
END ME_Delete;
/
PROMPT Creating TRIGGER ME_Delete_Stat ...
```

```
CREATE OR REPLACE TRIGGER ME_Delete_Stat        -- statement level trigger, delete from MO_
       AFTER DELETE ON ME_                      -- (and via cascading all dependent table-rows)
BEGIN
   CDIF_Data.delete_mo( 'ME' ) ;
END ME_Delete_Stat ;
/
```

Programmcode 6-2: Definition der Tabellen, VIEWs, Triggers und Stored Prozeduren in ORACLE's PL/SQL

6.2.1.3 EIA/CDIF-Analysebeispiele in SQL

In diesem Abschnitt werden exemplarisch SQL-Anweisungen dokumentiert, die demonstrieren, wie die in einer relationalen Datenbank abgelegten, standardisierten Definitionen der EIA/CDIF-Meta-Modelle ausgewertet werden können.[845] Sofern das Resultat gezeigt wird, bezieht es sich immer auf das Integrierte EIA/CDIF-Meta-Modell[846], das die Definitionen aus allen (fünf) per Anfang Jänner 1998 standardisierten Meta-Modellen beinhaltet.[847]

6.2.1.3.1 Kleine Übersicht über die Meta-Objekttypen

Der folgende Programmcode 6-3 zeigt eine Möglichkeit, wie man mit Hilfe einer definierten Sichtweise eine Aufstellung über die Anzahl der unterschiedlichen Meta-Objekttypen erhält. Das besondere daran ist, daß das Ergebnis aus nur einer einzigen Zeile besteht.

```
CREATE OR REPLACE VIEW view_Nr_of_instances_raw (MO, SA, CMO, MA, AMO, ME, MR) AS
   SELECT COUNT( * ), 0, 0, 0, 0, 0, 0 from mo
UNION
   SELECT 0, COUNT( * ), 0, 0, 0, 0, 0 FROM sa
UNION
   SELECT 0, 0, COUNT( * ), 0, 0, 0, 0 FROM cmo
UNION
   SELECT 0, 0, 0, COUNT( * ), 0, 0, 0 FROM ma
UNION
   SELECT 0, 0, 0, 0, COUNT( * ), 0, 0 FROM amo
UNION
   SELECT 0, 0, 0, 0, 0, COUNT( * ), 0 FROM me
UNION
   SELECT 0, 0, 0, 0, 0, 0, COUNT( * ) FROM mr
;

/* create or replace view view_Nr_of_instances (MO, SA, CMO, MA, AMO, ME, MR) as */
SELECT MAX( mo )   mo, MAX( sa ) sa,
       MAX( cmo ) cmo, MAX( ma ) ma,
       MAX( amo ) amo, MAX( me ) me, MAX( mr ) mr
FROM view_nr_of_instances_raw;

DROP view view_Nr_of_instances_raw;                -- remove view from system
```

Programmcode 6-3: Kleine Übersicht über die Meta-Objekttypen

---

845 Im Rahmen dieser Arbeiten wurden weit über 100 SQL-Anweisungen erarbeitet, die die unterschiedlichsten Überprüfungen und Auswertungen erlauben.

846 Vgl. auch Abschnitt 4.2, ‘Das „Integrierte EIA/CDIF-Meta-Modell“’ auf Seite 267ff.

847 Beim Studium dieser kleinen Auswahl an SQL-Anweisungen soll das Potential deutlich gemacht werden, das in einer systematischen Bearbeitung sowie Auswertung der standardisierten – und in einem relationalen Datenbankverwaltungssystem wie ORACLE vorgehaltenen – EIA/CDIF-Meta-Objektdefinitionen mit SQL liegt.

Die Ausführung der im Programmcode 6-3 enthaltenen SQL-Anweisungen führt zu folgender Ausgabe:

```
       MO        SA       CMO        MA       AMO[848]       ME        MR
--------- --------- --------- --------- --------- --------- ---------
      297         5       292       169       123        62        60
```

6.2.1.3.2 Kleine Auswertung über Meta-Attribute

Der folgende Programmcode 6-4 besteht aus einer SQL-Anweisung, die die Anzahl der optionalen und zwingenden Meta-Attribute auswertet, aufgeschlüsselt nach GegenstandsBereichen. Dafür werden zwei Meta-Meta-Beziehungstypen und drei Meta-Meta-Entitätstypen benötigt.

```
SELECT sa_.surr, SA_.Shorthand, MO_.mo_type, ma_.isoptional, count( * )
       FROM   SA_, MO_, MA_, IsUsedIn_, IsLocalMetaAttributeOf_
       WHERE
              MO_.SURR = IsUsedIn_.Source
           AND
              SA_.SURR = IsUsedIn_.Destination
           AND
              MO_.SURR = IsLocalMetaAttributeOf_.Destination
           AND
              MA_.SURR = IsLocalMetaAttributeOf_.Source
       GROUP BY SA_.surr, SA_.Shorthand, MO_.mo_type, ma_.isoptional
;
```

Programmcode 6-4: Aufzählung von Meta-Attributen, getrennt nach Optionalität und GegenstandsBereichen

Die Ausführung der im Programmcode 6-4 enthaltenen SQL-Anweisung führt zu folgender Ausgabe:

```
      SURR SHORTH MO_TY ISOPTIONAL  COUNT(*)
--------- ------ ----- ---------- ---------
        1 FND    AMO   False              1
        1 FND    AMO   True               4
       27 CMMN   ME    False              2
       27 CMMN   ME    True              18
      164 DMOD   ME    True              71
      164 DMOD   MR    True              10
      656 DFM    ME    False              1
      656 DFM    ME    True              40
      656 DFM    MR    False              1
      656 DFM    MR    True               1
      887 PLAC   ME    False              4
      887 PLAC   ME    True              24
      887 PLAC   MR    True               2
```

848 In dieser Anzahl sind sämtliche Meta-Entitätstypen und Meta-Beziehungstypen enthalten sowie die einzige direkte Instanz vom Meta-Meta-Entitätstyp AttributableMetaObject: „RootObject“.

6.2.1.3.3 Surrogatwertverletzungen

Im Meta-Modell für den Gegenstandsbereich „Datenmodellierung“ wurden im Zuge von Qualitätskontrollen mit Object Rexx-Programmen drei Surrogatfehler gefunden, indem drei für das Meta-Meta-Attribut „CDIFMetaIdentifier“ bereits vergebene Werte noch einmal Meta-Objekten zugewiesen wurden.

Eine äquivalente Überprüfung, die eine Liste jener Meta-Objekte liefert, deren Surrogatwerte öfter als einmal vergeben sind, ist im Programmcode 6-5 beinhaltet. Die erste SQL-Anweisung liefert als Ergebnis eine Liste jener Surrogatwerte des Meta-Meta-Attributs „CDIFMetaIdentifier“, die öfter als einmal vergeben wurden, wobei gezählt wird, wie oft ein derartiger Wert benutzt wurde.

Die zweite SQL-Anweisung listet die Meta-Objekte mit ihren Namen und weiteren Informationen, die von der fehlerhaften Mehrfachvergabe der Surrogatwerte betroffen sind. Für den Fall, daß Werte für das Meta-Meta-Attribut „CDIFMetaIdentifier“ mehrfach vergeben wurden, stellt nur mehr die Spalte „SURR“ die Unterscheidbarkeit der unterschiedlichen Meta-Objekte sicher.

```
PROMPT find duplicate usage of CDIFMetaIdentifier values ...
      SELECT CDIFMetaIdentifier , COUNT( * )
      FROM   MO_
      GROUP BY CDIFMetaIdentifier
      HAVING   COUNT( * ) > 1
;

PROMPT show types and names of duplicate usage of CDIFMetaIdentifier values ...
      SELECT
             MO_Type,
             Surr,
             SUBSTR( CDIFMetaIdentifier, 1, 4  ) "C-M-I",
             SUBSTR( Name              , 1, 20 ) "Name" ,
             SUBSTR( LongName          , 1, 50 ) "LongName"
      FROM  MO_
      WHERE
            CDIFMetaIdentifier IN
            (
                SELECT CDIFMetaIdentifier
                FROM   MO_
                GROUP BY CDIFMetaIdentifier
                HAVING   COUNT( * ) > 1
            )
      ORDER BY MO_Type, Name, LongName
;
```

Programmcode 6-5: Qualitätskontrolle der Surrogatwerte

Die Ausführung der im Programmcode 6-5 enthaltenen SQL-Anweisungen führt zu folgenden Ausgaben:

```
CDIFMETAIDENTIFIER                      COUNT(*)
--------------------------------------- ---------
1140                                            2
1733                                            2
1736                                            2


MO_TY   SURR C-M- Name                LongName
----- ------ ---- -------------------- --------------------------------------------------
MA       209 1140 SpecificationText
MA       187 1140 SpecificationText
MA       282 1736 StoreWithSupertype
MR       284 1733 Incorporates         Key.Incorporates.Attribute
MR       286 1733 Incorporates         Key.Incorporates.SemanticInformationObject
MR       300 1736 Specifies            SubtypeSet.Specifies.SubtypeSetMembershipCriterion
```

6.2.1.3.4 AttribuierbareMetaObjekte mit Mehrfachvererbung

Der folgende Programmcode 6-6 besteht aus einer SQL-Anweisung, die jene AttribuierbarenMetaObjekte dar stellt, für die eine Mehrfachvererbung definiert ist. Es werden die AttribuierbarenMetaObjekte mit ihren direkten Supertypen gezeigt, sortiert nach dem Meta-Typ, dem vollqualifizierten Namen, dem GegenstandsBereich in der Reihenfolge ihrer Eintragung in der Datenbank[849] sowie nach dem vollqualifizierten Namen der direkten Supertypen.

```
PROMPT show all subordinate AMOs with mult/inh and their SUPERTYPEs ...
      SELECT
             ShortHand,
             LPAD( T2.CDIFMetaIdentifier, 5  )        "C-M-I",
             T2.mo_type, substr( T2.LongName, 1, 30 ) "LongName (SUBTYPE) ",
             LPAD( T1.CDIFMetaIdentifier, 5  )        "C-M-I",
             T1.mo_type, substr( T1.LongName, 1, 30 ) "LongName (SUPERTYPE)"
      FROM   MO T2, HasSubtype_, MO T1, IsUsedIn_, SA
      WHERE
            T2.LongName IN
            (
               SELECT T2.LongName
               FROM   AMO T1, HasSubtype_, AMO T2
               WHERE
                     T1.surr = HasSubtype_.Source
                   AND
                     T2.surr = HasSubtype_.Destination
               GROUP BY T2.LongName
               HAVING COUNT( * ) > 1
            )
          AND
            T2.Surr = HasSubtype_.Destination
          AND
            T1.Surr = HasSubtype_.Source
          AND
            T2.SURR = IsUsedIn_.Source
          AND
            SA.SURR = IsUsedIn_.Destination
      ORDER BY T2.MO_Type, T2.LongName, SA.surr, T1.LongName
;
```

Programmcode 6-6: AttribuierbareMetaObjekte mit Mehrfachvererbung, gemeinsam mit ihren direkten Supertypen

Die Ausführung der im Programmcode 6-6 enthaltenen SQL-Anweisung führt zu folgender Ausgabe:

---

849 Die Sortierung erfolgt hierbei nach den Surrogatwerten der GegenstandsBereiche, sodaß die Reihenfolge letztendlich durch die der Einspeisung (vgl. Abschnitt 6.2.2.4, „Überführung von Meta-Modelldaten aus dem CTI-Datenformat in ORACLE-Tabellen“ auf Seite 479ff weiter unten) der entsprechenden Meta-Modelldatendefinitionen in die Datenbank bestimmt wird: für diese Arbeit wurde die Reihenfolge: FND, CMMN, DMOD, DFM und PLAC gewählt.

```
SHORTH C-M-I MO_TY LongName (SUBTYPE)             C-M-I MO_TY LongName (SUPERTYPE)
------ ----- ----- ------------------------------ ----- ----- --------------------------
DMOD    1005 ME    Cluster                         1012 ME    DataModelObject
DMOD    1005 ME    Cluster                         8002 ME    DefinitionObject
DMOD    1016 ME    Entity                          8002 ME    DefinitionObject
DMOD    1016 ME    Entity                          1033 ME    InheritableDataModelObject
DFM     6015 ME    FlowPort                         232 ME    FlowProducerConsumer
DFM     6015 ME    FlowPort                        1129 ME    Port
DMOD    1040 ME    Relationship                    8002 ME    DefinitionObject
DMOD    1040 ME    Relationship                    1033 ME    InheritableDataModelObject
```

6.2.1.3.5 Verteilung der zwingend vorgeschriebenen Meta-Beziehungstypen auf die GegenstandsBereiche

Der folgende Programmcode 6-7 zeigt, wie man mit Hilfe einer einfachen SQL-Anweisung eine Verteilung von zwingend vorgeschriebenen Meta-Beziehungstypen, nach GegenstandsBereichen getrennt, erhalten kann.

```
PROMPT Total AMOs with mandatory MAs ...
SELECT COUNT( * ) "AMO w mMA" , mo_type
FROM   mo_
WHERE
       MO_TYPE IN ('MR', 'ME', 'AMO' )
     AND
       EXISTS ( SELECT *
                FROM  IsLocalMetaAttributeOf_, MA_
                WHERE IsLocalMetaAttributeOf_.Destination = mo_.surr
                     AND
                      Source = MA_.SURR
                     AND
                      MA_.IsOptional = 'False'
                )
GROUP BY mo_type
;
```

Programmcode 6-7: Kleine Übersicht über die Meta-Objekttypen

Die Ausführung der im Programmcode 6-7 enthaltenen SQL-Anweisung führt zu folgender Ausgabe:

```
MO w mMA MO_TY
-------- -----
       1 AMO
       6 ME
       1 MR
```

6.2.1.3.6 Auflistung der Längen von verschiedenen zeichenketten-basierten Werten

Der folgende Programmcode 6-8 zeigt, wie man mit Hilfe einer einfachen SQL-Anweisung die maximalen Längen von Namen berechnen lassen kann.

```
SELECT MAX( LENGTH( mo.name ) )     "length-name",
       MAX( LENGTH( mo.longname ) ) "length-longname",
       MAX( LENGTH( ma.datatype ) ) "length-datatype"
FROM   mo, ma;
```

Programmcode 6-8: Kleine Übersicht über die Meta-Objekttypen

Die Ausführung der im Programmcode 6-8 enthaltenen SQL-Anweisung führt zu folgender Ausgabe:

```
length-name length-longname length-datatype
----------- --------------- ---------------
         35              64              10
```

6.2.1.3.7 Verteilung der AttribuierbarenMetaObjekte mit und ohne Meta-Attribute, nach GegenstandsBereichen getrennt

Der folgende Programmcode 6-9 zeigt, wie man mit Hilfe einer entsprechend definierten Sichtweise und mit einer einfachen SQL-Anweisung die Verteilung der AttribuierbarenMetaObjekte mit und ohne Meta-Attribute, getrennt nach GegenstandsBereichen, berechnen lassen kann.

```
CREATE OR REPLACE VIEW view_amos_and_ma_raw
                      ( sa_surr, sa_shorthand, cmo_with_ma, cmo_no_ma, mo_type ) AS
    SELECT                                      /* AMOs with MAs */
           sa.surr, sa.shorthand,
           COUNT( cmo.mo_type ), 0, cmo.mo_type
    FROM   sa, IsUsedIn_, cmo
    WHERE  cmo.mo_type <> 'MA' AND
           sa.surr = IsUsedIn_.Destination AND IsUsedIn_.Source = cmo.surr
           AND     EXISTS( SELECT surr
                           FROM   IsLocalMetaAttributeOf_
                           WHERE  destination = cmo.surr )
    GROUP BY sa.surr, sa.shorthand, cmo.mo_type
UNION
    SELECT                                      /* AMOs without MAs */
           sa.surr, sa.shorthand,
           0, COUNT( cmo.mo_type ), cmo.mo_type
    FROM   sa, IsUsedIn_, cmo
    WHERE  cmo.mo_type <> 'MA' AND
           sa.surr = IsUsedIn_.Destination AND IsUsedIn_.Source = cmo.surr
           AND NOT EXISTS( SELECT surr
                           FROM   IsLocalMetaAttributeOf_
                           WHERE  destination = cmo.surr )
    GROUP BY sa.surr, sa.shorthand, cmo.mo_type
;
```

```
SELECT sa_surr, sa_shorthand,
       MAX( cmo_with_ma ) "# with MAs",
       MAX( cmo_no_ma   ) "# without MAs",
       MO_TYPE
FROM   view_amos_and_ma_raw
GROUP  BY sa_surr, sa_shorthand, mo_type
ORDER  BY sa_surr, DECODE( mo_type, 'ME', 2, 'MR', 3, 'MA', 4, 1 );
```

Programmcode 6-9: Verteilung der AttribuierbarenMetaObjekte mit und ohne Meta-Attribute, nach GegenstandsBereichen getrennt

Die Ausführung der im Programmcode 6-9 enthaltenen SQL-Anweisungen führt zu folgender Ausgabe:

```
   SA_SURR SA_SHO # with MAs # without MAs MO_TY
---------- ------ ---------- ------------- -----
         1 FND             1             0 AMO
         1 FND             0             1 ME
         1 FND             0             1 MR
        27 CMMN            8             2 ME
        27 CMMN            0             8 MR
       164 DMOD           18             4 ME
       164 DMOD            7            27 MR
       656 DFM            11            11 ME
       656 DFM             2             6 MR
       887 PLAC            7             6 ME
       887 PLAC            2             8 MR
```

6.2.2 Implementierung des EIA/CDIF-Meta-Meta-Modells in Object Rexx

Die Implementierungsbeispiele[850] in diesem Abschnitt erfolgen auf Basis der Spezifikationen für eine Klassenhierarchie zur Repräsentation der EIA/CDIF-Meta-Meta-Entitätstypen und einem Satz von Relationsobjekten, die die EIA/CDIF-Meta-Meta-Beziehungstypen repräsentieren, wie sie in dieser Arbeit im Abschnitt 3.3.2, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf Object Rexx“ auf Seite 124ff, erfolgt sind.

Die Implementierungssprache ist Object Rexx, die Mitte 1998 frei für AIX, Linux und OS/2 Warp zur Verfügung steht. Eine Version für Windows/95 und Windows/NT kann (Stand: Juni 1998) von IBM gekauft werden. [851, 852]

Grundsätzlich werden die vom Autor geschaffenen und im WWW weltweit frei zur Verfügung gestellten Object Rexx-Module in [W3ORX8][853] benötigt, da sie

---

850 Im Rahmen dieser Arbeit wurden zahlreiche Object Rexx-Skripts und Object Rexx-Programme erstellt, sodaß es sich in diesem Abschnitt tatsächlich nur um eine zusammenhängende und für sich allein ablauffähige Sammlung von Implementierungsbeispielen handelt.

851 Die Linux-Version kann von [W3ORX-LINUX], die OS/2 Warp 3 (und als Update für OS/2 Warp 4) von [W3ORX-OS2] aus dem Internet geladen werden. Informationen finden sich auf der Homepage von Object Rexx unter [W3ORexx] sowie allgemeine und umfassende Informationen zur Rexx-Sprachfamilie auf der Homepage von Rexx unter [W3Rexx].

852 Für Object Rexx gibt es eine Reihe von einführenden Büchern, beispielsweise: [Ende97], [TurWah97], [VeTrUr96] und [WahHolTur97]. Darüber hinaus können auch die Ausführungen in [Flat96c] und [Flat96d] interessant sein, die zum einen die Kopplung von Object Rexx-Programmen über die zur Verfügung stehenden, verschiedenen „Umgebungen“ erstmalig im Detail vorstellen, sowie die wesentlichen Object Rexx-Klasseneigenschaften diskutieren und anhand von Beispielen demonstrieren. Unter anderem findet sich ein etwa 100zeiliges Object Rexx-Programm, das die echten Objekteigenschaften der auf IBM’s SOM basierenden Workplace-Shell von OS/2 dazu ausnützt, um paßwortgeschützte Ordner zu implementieren.

853 Es handelt sich dabei um die folgenden Module: „`class_ref.cmd`“ (Erzeugung und Verwaltung von Referenzen), „`class_rel.cmd`“ (Klassen, die die Object Rexx-Klasse „Relation“ spezialisieren), „`get_answ.cmd`“ (enthält einfache Routine zur direkten Tastaturabfrage), „`html_util.cmd`“ (Klassen und Routinen für die Erzeugung von HTML-Dokumenten), „`is_util.cmd`“ (verschiedene Object Rexx-Testroutinen), „`nls_util.cmd`“ (Klassen und Routinen für die codepagebasierte Manipulation von Zeichenketten), „`rgf_util.cmd`“ (Modul, das über „`::REQUIRES`“-Direktiven die meisten weiteren Module einbindet), „`routine_find_file.cmd`“ (Routine zum Suchen nach Dateien), „`routine_ok.cmd`“ (Routine zum Ausgeben des Wahrheitswertes), „`routine_pp.cmd`“ (Routine zum Einfassen von Zeichenketten in eckige Klammern), „`routine_pp_number.cmd`“ (Routine zum Formatieren von Zahlen), „`routine_strip_quote.cmd`“ (Routine zum Entfernen/Hinzufügen von Anführungszeichen und Apostrophen), „`routine_usify.cmd`“ (Routine zum Erzeugen einer 7-Bit-US-ASCII-Zeichenkette, die mit einem Buchstaben beginnt), „`sgmlentity_util.cmd`“

Fortsetzung folgt auf der nächsten Seite.

in den folgenden Object Rexx-Programmen zum Teil in „::REQUIRES“-Direktiven referenziert werden.[854]

Für den Zugriff auf relationale Datenbanken wird auf die frei zur Verfügung stehende Rexx-Funktionsbibliothek[855] „RexxSQL“ zurückgegriffen, die einheitliche Programmierschnittstellen für zahlreiche unterschiedliche relationale Datenbanken[856] auf verschiedenen Plattformen[857] zur Verfügung stellt.[858]

(Klassen und Routinen zum Verwalten von SGML-Entities) und endlich „sort_util.cmd“ (versatile Routinen zum Sortieren von Objekten in Sammelobjekten).

854 Mit Hilfe von [W3ORX7] können (Object) Rexx-Programme analysiert und ausgewertet werden. Bemerkenswerterweise werden von Anfang an, die bis zum Erscheinen der Linux-Version Ende 1997 undokumentierten Security-Manager-Methoden mitausgewertet, von denen der Autor im Rahmen des Beta-Tests der Programmiersprache Kenntnis erlangt hatte. Die Ergebnisse werden wahlweise in ASCII- oder in einem entsprechend schöneren Layout in HTML-Dateien aufbereitet.

855 Für die (prozedurale) Programmiersprache Rexx, die ursprünglich von IBM entwickelt wurde, erfolgten betriebssystemunabhängige Schnittstellendefinitionen im Rahmen von IBM's „System Application Architecture“ (abgekürzt: „SAA“). Mit deren Hilfe können Prozeduren und Funktionen in anderen Programmiersprachen als Rexx erstellt werden, die in Folge von Rexx aus direkt aufrufbar sind. Dies hatte bisher den Effekt, daß die in Rexx eingebauten Standardfunktionen in ihrem Umfang in den letzten zehn Jahren fast nicht erweitert werden mußten.
Vgl. in diesem Zusammenhang auch [W3Rexx]. Diese WWW-Homepage wird von Mike F. Cowlishaw geführt, dem Begründer von Rexx und der frei verfügbaren Java-Programmiersprache „NetRexx“ (vgl. auch [W3NetRexx]). Es finden sich auch zahlreiche Verweise auf die verschiedensten kommerziellen und freien Implementierungen der Rexx-Programmiersprache.

856 Vgl. [W3Rexx-SQL]. Es wurden von Mark Hessling bisher für folgende relationale Datenbanken RexxSQL-Funktionspakete erstellt: „DB2“ (a.k.a. „Universal Database“), „Generic ODBC“, „Informix“, „Open Ingres“, „mSQL“ (Versionen 1.0.16, 2.0), „MySQL“,, „Openlinkd UDBC“, „ORACLE“ (Versionen 7.x, 8.x), „PostgreSQL“, „Solid Server“, „Sybase SQL Anywhere“ und „Sybase System 10“ (11).

857 Es handelt sich hierbei um die Betriebssysteme OS/2, Unix (Linux), Windows/95 und Windows/NT.

858 Es ist auch bemerkenswert, daß die RexxSQL-Funktionspakete für die unterschiedlichen relationalen Datenbanken *gleichzeitig* in ein und demselben Rexx-Programm angesprochen werden können. Somit kann die Übertragung von Tupeln aus der relationalen Datenbank eines Herstellers *direkt* in eine relationale Datenbank eines anderen Herstellers erfolgen!
Grundsätzlich ist es möglich, daß Rexx-Programme, die „RexxSQL“ einsetzen und keine proprietären Erweiterungen der entsprechenden relationalen Datenbanken ausnutzen, ohne Änderungen am Programmcode weiterlaufen, obwohl das Datenbankverwaltungssysteme in der Zwischenzeit ausgetauscht wurde.

6.2.2.1 Kurzeinführung in die Programmiersprache Object Rexx

In diesem Abschnitt erfolgt mit Hilfe von kurzen, kommentierten Rexx-Programmen eine kurze Darstellung von Object Rexx[859], um das Verständnis der folgenden Object Rexx-Programme zu erleichtern.[860] Es ist nahezu zu 100 %[861] mit der – ursprünglich auch von IBM entwickelten – prozeduralen Rexx-Programmiersprache kompatibel, sodaß zunächst in einem eigenen Unterabschnitt eine Kurzcharakterisierung von Rexx erfolgt.

6.2.2.1.1 Rexx

Die Programmiersprache Rexx wurde von dem IBM-Mitarbeiter Mike F. Cowlishaw von 1979 bis 1982 als ausdrücklich endbenutzerfreundliche Programmiersprache entwickelt: „The primary design goal has been that it should be genuinely easy to use both by computer professionals and by 'casual' general users“[862]. 1983 wurde sie erstmals als ein IBM-Produkt für das IBM-Mainframe-Betriebssystem „Virtual Machine/System Product“ (abgekürzt: „VM/SP“) und „Conversational Monitor System“ (abgekürzt: „CMS“) als Systeminterpretersprache eingesetzt.[863]

Die Programmiersprache Rexx wird in der Regel durch einen Interpreter realisiert. Sie weist keine Datentypen außer uninterpretierte Zeichenketten

---

859 Object Rexx wurde ursprünglich 1995 von IBM über ein Betatestprogramm für OS/2-Entwickler zur Verfügung gestellt, wobei die von Grund auf neu erfolgte Entwicklung dieser Programmiersprache aus einem IBM-„High-Tech“-Budgettopf finanziert wurde.

860 Vgl. auch die Erläuterungen zu Object Rexx in Abschnitt 3.3.2, „Spezifikationen für die Abbildung des EIA/CDIF-Meta-Meta-Modells auf Object Rexx“ auf Seite 124ff.

861 In der Dokumentation zu Object Rexx werden diese Unterschiede im Abschnitt „Migration“ hervorgehoben, betreffen aber im großen und ganzen kaum „klassische“ Rexx-Programme. Der wichtigste Unterschied liegt darin, daß für die „STREAM“-Funktion der Dateizeiger in einen Lese- und in einen Schreibzeiger aufgeteilt wurde. Damit diese Änderung nicht zu unbeabsichtigten Fehlern bei alten Rexx-Programmen führt, muß in Positionierungsanweisungen die Angabe „READ“ oder „WRITE“ angegeben werden, sodaß in diesem Zusammenhang unter Object Rexx ein Fehler erzwungen wird. Diese Änderung ist nebenbei auf den ANSI-Rexx-Standard zurückzuführen, der 1996 verabschiedet wurde (vgl. [ANSI96]).

862 Vgl. [Cowl90], Seite 1, „Section 1“, erster Absatz Mitte. Vgl. auch die Ausführungen zur Geschichte der Programmiersprache ebenda, Seiten 15 bis 16.

863 In weiterer Folge wurde Rexx von IBM zur SAA-Systemintersprache deklariert und daher auf weitere strategische IBM-Betriebssysteme portiert: MVS, OS/400 und OS/2. Parallel dazu entstanden weitere kommerzielle und freie Versionen dieser Programmiersprache für defacto sämtliche, kommerziell verfügbare Betriebssysteme.

(englisch: „String“) auf, sodaß beispielsweise erst bei Auftreten von arithmetischen Operationen die Zeichenketten daraufhin untersucht werden, ob sie gültige Zahlen[864] beinhalten.

Im folgenden Programmcode 6-10[865, 866] sind alle Rexx-reservierten Wörter, Rexx-Funktionen beziehungsweise weitere syntaktische Rexx-Elemente fett hervorgehoben.[867] Jedes Rexx-Programm muß mit einem Kommentar in der ersten Zeile, ersten Spalte beginnen. Kommentare beginnen mit „**/***“ und enden mit „***/**“, wobei sie sich über beliebig viele Zeilen erstrecken und beliebig ineinander verschachtelt sein dürfen.

Rexx-Anweisungen werden durch einen Strichpunkt abgeschlossen, der allerdings vom Rexx-Interpreter auch durch das Zeilenende impliziert wird, sodaß Rexx-Programme normalerweise keine Strichpunkte aufweisen. Ist eine Rexx-Anweisung besonders lang und wird sie auf mehrere Zeilen aufgeteilt, so muß am Ende einer jeden Zeile ein Komma stehen, das darauf hinweist, daß die Anweisung in der nächsten Zeile seine Fortsetzung findet.

Die Variable „**a**“ erhält in Zeile 2 den Wert „**1**“, die Variable „**b**“ in Zeile 3 den Wert „**2**“ zugewiesen; es ist bei Zahlen belanglos, ob die Zeichenkettenbegrenzungszeichen (**"** – Anführungszeichen) oder (**'** – Apostroph) mit angegeben sind oder nicht. Für die Ausgabe von Zeichenketten und Variableninhalten wird die Rexx-Anweisung „**SAY**“ benutzt. In Zeile vier werden die Variableninhalte, durch ein Leerzeichen getrennt, ausgegeben, in Zeile fünf wird der Verkettungsoperator ‘**||**’ verwendet, sodaß das Leerzeichen entfällt und die Zeichenkette aus dem Wert „**12**“ besteht. Zeile 6 beinhaltet eine Addition, die als Summe den Wert „**3**“ ergibt, da die Variableninhalte als Zahlen aufgefaßt werden.

---

864 Eine bemerkenswerte Eigenschaft von Rexx ist, daß für arithmetische Operationen eine beliebige Anzahl an Ziffern angegeben werden kann, mit denen gerechnet werden soll: beispielsweise ergibt „`1/3`“ den Wert „`0.33333333333333333333333333333333`“, wenn zuvor die „`NUMERIC DIGITS 30`“-Anweisung gegeben wurde. Bei einer Angabe von „`1000`“ Ziffern statt „`30`“ in der vorhergehenden Anweisung wäre das berechnete Ergebnis eine entsprechend lange Zahl.

865 Siehe Seite 454 weiter unten.

866 Die in eckigen Klammern vorangestellten Zeilennummern sind *nicht* Bestandteil der exemplarischen Rexx-Programmanweisungen.

867 Die in Kommentaren fett und kursiv hervorgehobenen Zeichen stellen das Ergebnis der entsprechenden Rexx-Anweisung dar.

Unbekannte Rexx-Anweisungen führen nicht sofort zu einem Fehler, sondern werden zunächst in Form einer Zeichenkette an die „Umgebung“ weitergereicht, also üblicherweise dem Programm übergeben, das den Rexx-Interpreter aufgerufen hat. Der Standardfall ist die Kommandozeilenumgebung, sodaß der Kommandointerpreter des Betriebssystems im folgenden Programmcode 6-10 die unbekannte Anweisung „**dir**“ (Zeile 7) ausführt. Erst wenn die „Umgebung“ einen Fehler anzeigt, wird das Rexx-Programm selbst mit einer entsprechenden Fehlermeldung abgebrochen.

Zeile 8 zeigt den Aufruf der Funktion[868] „**tus_a**“ mit zwei Argumenten und liefert als Ergebnis den Wert „**4**“. Die Funktion „**tus_a**“ ist in den Zeilen 15 bis 18 definiert und verfügt über einen eigenen Geltungsbereich, nachdem unmittelbar nach der Sprungmarke, die durch einen nachfolgenden Doppelpunkt als solche definiert ist, die Rexx-Anweisung „**PROCEDURE**“ angeführt ist.[869] Zeile 16 zerlegt das Argument in zwei Werte und weist sie lokalen Variablen zu, die nicht zufällig auch „**a**“ und „**b**“ heißen. Die Änderung am Wert der Variablen „**a**“ ist nur in dieser Funktion sichtbar, sodaß nach dem Funktionsaufruf in Zeile 8, in Zeile 9 die Variableninhalte von „**a**“ und „**b**“ unverändert die des aufrufenden Programmabschnitts sind.

Zeile 10 stellt eine zur Zeile 8 alternative Form des Funktionsaufrufes von „**tus_a**“ dar. Ein Funktionswert wird hierbei von Rexx in der Funktion selbst der globalen Systemvariablen „**RESULT**“ zugewiesen, deren Inhalt in Zeile 11 für die Darstellung benutzt wird.

Zeile 12 ruft die Funktion „**tus_b**“ auf, die in den Zeilen 19 bis 22 definiert ist. Im Unterschied zur Funktion „**tus_a**“ in Zeile 15, fehlt nach der Marke „**tus_b:**“ die „**PROCEDURE**“-Anweisung, sodaß die Funktion über keinen lokalen Geltungsbereich verfügt. Änderungen an den Variableninhalten „**a**“ und „**b**“ erfolgen in den Variablen des Aufrufers und bleiben daher auch nach der Rück-

868 Wie üblich liefern Funktionen Funktionswerte, Prozeduren nicht. Entsprechend wird in Rexx bei der Rückkehranweisung „RETURN“ in Funktionen ein Funktionswert mitangegeben, bei Prozeduren nicht.

869 Es wäre möglich, unmittelbar an die „PROCEDURE“-Anweisung mit einer „EXPOSE“-Anweisung den Zugriff auf Variablen des Aufrufers zu gewähren und somit den lokalen Geltungsbereich gezielt zu erweitern.

kehr aus der Funktion verändert. Dies wird durch die Ausgabe in Zeile 13 gezeigt, wo im Unterschied zur Zeile 9 die Variable „**a**“ den Wert „**3**“ aufweist.

```
      /* This is a Rexx comment and must start at line 1 and column 1 */
[1]   a = 1
[2]   b = "2"
[3]   SAY a b             /* yields: 1 2 */
[4]   SAY a || b          /* yields: 12  */
[5]   SAY a + b           /* yields: 3   */
[6]   "dir"               /* not a valid Rexx statement, will be given to the
                             invoker of the Rexx interpreter, usually the shell;
                             under DOS, OS/2 and Windows this results into a
                             file listing */
[7]   SAY tus_a( a, b )   /* yields: 4                                  */
[8]   SAY "a =" a ‘b =‘ b /* yields: a = 1 b = 2                        */
[9]   CALL tus_a a, b ;   /* (optional) semi-colon ends a Rexx statement */
[10]  SAY RESULT          /* "RESULT" contains the return value of the last
                             called function (set by Rexx), yields: 4    */
[11]  SAY tus_b( a, b )   /* yields: 5                                  */
[12]  SAY "a =" a ‘b =‘ b /* yields: a = 3 b = 2                        */
[13]  EXIT -1             /* ends the program, return value: -1         */

[14]  TUS_A : PROCEDURE /* opens its own scope containing its own local variables   */
[15]     PARSE ARG a, b /* parses argument and assigns the values to local variables*/
[16]     a = a * 2      /* value of "a" gets doubled .........                    .*/
[17]     RETURN a + b   /* produces and returns a result value to the caller          */

[18]  TUS_B :           /* same scope as caller (!) == no local variables             */
[19]     PARSE ARG a, b /* parses argument and assigns the values to global variables */
[20]     a = a * 3      /* value of "a" gets tripled .........                    .*/
[21]     RETURN a + b   /* produces and returns a result value to the caller          */
```

Programmcode 6-10: Beispiel für ein Rexx-Programm

6.2.2.1.2 Object Rexx

Die Programmiersprache Object Rexx wurde von IBM aufgrund von Kundenanforderungen als eine von Grund auf neu konzipierte, objektorientierte Programmiersprache entwickelt. Damit die Investitionen in Rexx[870] nicht umsonst erfolgten, mußte Object Rexx damit entsprechend den Kundenwünschen vollständig kompatibel sein. Die ursprüngliche Entwicklungsplattform war OS/2, allerdings wurde von Anfang an das Ziel verfolgt, Object Rexx auf soviele Plattformen wie möglich zu portieren. Die objektorientierte Programmiersprache ist mittlerweile (Stand: Juni 1998) kostenlos für AIX, Linux[871] und für OS/2 Warp

870 In diesem Zusammenhang wird Rexx häufig auch als „prozedurales Rexx“ oder auch „traditionelles Rexx“ (scherzhaft abgekürzt: „T-Rexx“) bezeichnet.

871 Damit ist es den IBM-Entwicklern möglich, auf Kundenwunsch Object Rexx auf jede POSIX-Plattform mit relativ geringem Adaptionsaufwand zu portieren. Dies wird durch die

Fortsetzung folgt auf der nächsten Seite.

verfügbar und wird für die Windows/95 und Windows/NT in Form eines Interpreters und gebündelt mit einer graphischen Entwicklungsumgebung verkauft.

Das Objektmodell von Object Rexx lehnt sich sehr stark an Smalltalk an, indem beispielsweise Klassen als Klassenobjekte zur Laufzeit existieren und genauso wie Instanzen über Attribute und Methoden verfügen können.[872] Object Rexx-Klassendefinitionen werden üblicherweise in Form von Klassen- und Methodendirektiven, die mit einem doppelten Doppelpunkt eingeleitet werden („**::**“) festgelegt. Ähnlich wie in Smalltalk[873] können aber defacto sämtliche Definitionen auch zur Laufzeit erfolgen, wobei es auch möglich ist, die Klasseneigenschaften zu ändern[874] und unmittelbar daran anschließend weitere Instanzen – mit veränderten Eigenschaften – zu bilden. Die speziellen Object Rexx-Variablen „self“ und „super“ enthalten entsprechend Smalltalk eine Referenz auf das Klassenobjekt der Instanz beziehungsweise auf das Klassenobjekt der über die Generalisierungshierarchie („Vererbungshierarchie“) direkt übergeordneten Klasse. Im Unterschied zu Smalltalk ist es auch möglich, das Konzept der Mehrfachvererbung einzusetzen. Der Nachrichtenoperator von Object Rexx ist die Tilde („**~**“), in Object Rexx auch „Twiddle“ genannt. Kaskadierende Nachrichten werden durch eine Doppeltilde realisiert.

Object Rexx verfügt über eine Reihe eingebauter Klassen[875], die direkt aus der Umgebung[876], die die Kopplung von Object Rexx-Programmen untereinander

---

im Juni 1998 neu hinzugekommene AIX-Version der objektorientierten Programmiersprache unterstrichen.

872 Somit kann zwischen Klassen- und Instanzattributen und zwischen Klassen- und Instanzmethoden unterschieden werden.

873 Vgl. [GolRob83], [Kras83], [Gold84] sowie [Hoff87].

874 Dies beinhaltet das Ändern oder das Erweitern um weitere Methoden. Sofern eine einzige Instanz nur in einem begrenzten Methodenumfang sich von anderen unterscheiden soll (englisch: „one-off-object“), steht dafür eine entsprechende Klassenmethode zur Verfügung (es handelt sich hierbei um die Methode „`ENHANCED`“), zudem gibt es die Möglichkeit mit der Objektmethode „`SETMETHOD`“, Methoden zu löschen, überschreiben oder hinzuzufügen.

875 Es handelt sich unter anderem um die Klassen für Sammlungen von Objekten („`.Array`“, „`.Bag`“, „`.Directory`“, „`.List`“, „`.Queue`“, „`.Relation`“, „`.Set`“, „`.Table`“), eine Iteratorklasse („`.Supplier`“), Klassen zur Definition von Metaklassen („`.Class`“), von Methoden („`.Methods`“) und von Nachrichten („`.Message`“), die synchron oder asynchron versendbar sind. Darüber hinaus existieren Klassen zur Zeitsteuerung („`.Alarm`“), zur Überwachung („`.Monitor`“), für assoziative Felder („`.Stem`“), für Ein- und Ausgabe von Datenströmen („`.Stream`“), sowie eine Zeichenkettenklasse („`.String`“).
Verschiedene Eigenschaften von Object Rexx-Klassen werden in [Flat96d] vorgestellt und diskutiert.

und mit dem Interpreter erlaubt, angesprochen werden können. Eine Besonderheit liegt darin, daß sämtliche klassischen Rexx-Variablen vom Datentyp „String“ sind, also Instanzen der Klasse „**`.String`**“, wobei sie ihren definierten Wert nie ändern können. Wird ein Variablenwert vom Typ „String“ geändert, dann erfolgt dies, indem der Interpreter automatisch eine neue Stringinstanz bildet, die den neuen Wert erhält.[877]

Im Unterschied zum klassischen Rexx ist es möglich, (Object) Rexx-Module zu bilden, die über die „**`::REQUIRES`**“-Direktive referenziert werden können. Öffentlich definierte Routinen und Klassen aus solchen Modulen stehen in weiterer Folge zur Verfügung.[878] (Object) Rexx-Programme werden vom Object Rexx Interpreter zunächst syntaktisch vollständig[879] auf Korrektheit überprüft, daran anschließend werden die Direktiven[880], die durch einen Doppelpunkt eingeleitet werden, befolgt beziehungsweise umgesetzt und zuletzt[881] verzweigt die Programmablaufkontrolle an den Anfang des Programmes, wobei das Ende des Programmes durch die letzte Programmanweisung oder die erste Direktive angezeigt wird.

---

876 Es gibt eine unbenannte Umgebung für jedes Object Rexx-Modul, ein benanntes Verzeichnis „`.local`“ für jede Object Rexx Sitzung und eine Umgebung „`.environment`“, die unter OS/2 global allen Object Rexx-Programmen zur Verfügung steht.
Das Konzept der Umgebung (englisch: „environment“) wird in [Flat96c] erstmals detailliert erklärt und diskutiert.

877 Somit sind Instanzen vom Typ „String“ unveränderlich. Dies ist notwendig, damit Object Rexx-Programme mit prozeduralen Rexx-Programmen kompatibel bleiben.

878 Es gibt eine Reihe weiterer, interessant erscheinender Eigenschaften der Programmiersprache – beispielsweise der Sicherheitsmanager („Security Manager“), mit dessen Hilfe beliebige und beliebig sichere Sicherheitspolitiken implementiert werden können – , die in der Dokumentation selbst sowie in den entsprechenden Object Rexx Büchern zu finden sind (vgl. zum Beispiel. [Ende97], [TurWah97], [VeTrUr96] sowie [WahHolTur97]).

879 Dies hat im Zusammenhang mit traditionellen Rexx-Programmen zu Überraschungen geführt, da durch diese Kontrolle erstmals eine vollständige syntaktische Überfprüfung erfolgt. Bis dahin wurden lediglich jene Programmanweisungen überprüft, die als nächstes zur Ausführung angestanden sind. Somit sind Fälle eingetreten, wo in Programmzweigen syntaktisch falsche Rexx-Programmanweisungen – beispielsweise durch Tippfehler bedingt – enthalten sind, die über Jahre hinweg nicht aufgesucht wurden und somit auch zu keinen Fehlern führten.

880 Es gibt vier Direktiven: „`::REQUIRES`“ zum Laden von Modulen, „`::ROUTINES`“ zum Definieren von Routinen, die Funktions- oder Prozedurcharakter haben können, „`::CLASS`“ zur Definition von Klassen und „`::METHOD`“ zum Festlegen von Methoden.

881 Damit ist es möglich, daß während der Initialisierung von Klassenobjekten als Folge der Abarbeitung der Klassendirektiven bereits Object Rexx-Programmcode zur Ausführung gelangt, ehe die Kontrolle an den Beginn des Programmes übergeben wird. Für Module besitzen die Object Rexx-Anweisungen am Anfang üblicherweise Initialisierungscharakter für das Modul selbst.

Im folgenden Programmcode 6-11[882] sind jene Anweisungsteile fett hervorgehoben, die Object Rexx-spezifisch sind; Direktiven werden grau hinterlegt. Zeile 4 ist semantisch mit Zeile 3 äquivalent und demonstriert, wie die objektorientierte Schreibweise[883] aussieht. „**.output**“ ist ein Monitorobjekt, das standardmäßig Ausgaben auf „**stdout**“ umleitet. Somit wird die Nachricht „SAY“ dem Objekt „**.output**“ geschickt, wobei als Argument das Ergebnis des Ausdruckes mitgegeben wird, also „**a: 1 + b: 2 = 3**“.

In Zeile 6 wird ein Objekt von der in den Zeilen 13 bis 26 definierten Klasse „**test**“ gebildet, dessen Klassenobjekt über die Umgebung unter der Bezeichnung „**.test**“[884] angesprochen werden kann, indem die Klassenmethode zur Bildung von neuen Instanzen (Objekten) „**NEW**“[885] gesendet wird. Als Argument wird der Wert „**Max**“ übergeben. Somit wird zunächst die Klassenmethode „**new**“ in Zeile 18 aufgerufen, der Instanzzähler „**Counter**“ um eins im Wert erhöht, anschließend die Nachricht an die übergeordnete Klassenmethode „**new**“ durch die „**FORWARD**“-Anweisung in Zeile 21[886] weitergeleitet, die die Instanz bildet und automatisch die „**INIT**“-Instanznachricht an sie schickt. In der Methode in Zeile 23 wird in Zeile 24 mit Hilfe der „**EXPOSE**“-Anweisung der direkte Zugriff auf das Instanzattribut „**Name**“ eröffnet und mit der „**USE ARG**“-Anweisung einerseits das Argument „**Max**“ entgegengenommen und direkt dem Instanzattribut (eine Instanzvariable) zugewiesen. In Zeile 7 wird dementsprechend eine Instanz (ein Objekt) gebildet, die im Instanzattribut „**Name**“ den Wert „**Moritz**“ speichert.

---

882 Siehe Seite 459 weiter unten.

883 Tatsächlich wandelt der Parser klassische Object Rexx-Programme intern in die objektorientierte semantisch äquivalente Darstellung um. Für das Programmieren kann daher wahlweise die klassische und die objektorientierte Anweisungsform benutzt werden.

884 Der führende Punkt weist den Object Rexx-Interpreter darauf hin, daß die dem Punkt folgende Zeichenkette den Index in das Umgebungsverzeichnis darstellt. Somit wird versucht, aus dem Umgebungsverzeichnis jenes Objekt abzufragen, das mit dem angegebenen Index abgespeichert wurde. Das besondere dabei ist, daß der Interpreter bei solchen „Umgebungssymbolen“ (in der englischen Dokumentation der Programmiersprache als „environment symbol“ bezeichnet) selbständig der Reihe nach sämtliche Umgebungsverzeichnisse durchsucht, bis das gesuchte Objekt gefunden wird, ansonsten wird eine Ausnahmebedingung angezeigt.

885 Die Groß- und Kleinschreibung von Methodennamen ist üblicherweise – wie die Namen der Variablen oder Sprungmarken – irrelevant. Der Rexx-Parser wandelt die Zeichenkette grundsätzlich in Großbuchstaben um. Eine Ausnahme bilden Bezeichner, die in Zeichenkettenbegrenzungszeichen eingeschlossen sind (Anführungszeichen oder Apostroph).

886 Im vorliegenden Fall handelt es sich um die „NEW“-Methode der Klasse „.Class“.

Zeile 8 sendet die Klassenmethode „**counter**“ an das Klassenobjekt für die Klasse „**test**“ und erhält im Gegenzug den aktuellen Wert als Ergebnis zurück, zudem wird mit der Nachricht „**ID**“[887] der Klassenname abgefragt und mit ausgegeben.

In Zeile 9 werden die Werte im jeweiligen Instanzattribut der Objekte von „**one**“ und „**two**“ abgefragt und ausgegeben. Zeile 10 setzt diese Werte in eckige Klammern ein, indem jeweils die private[888] Routine „**pp**“ in Zeile 11 dafür aufgerufen wird.

Wenn das Programm im folgenden Programmcode 6-11 gestartet wird, kommt es – wie bereits weiter oben erwähnt – zunächst zu einer syntaktischen Überprüfung der Programmanweisungen. Anschließend arbeitet der Interpreter die Direktiven ab und legt in weiterer Folge aufgrund der Definitionen in den Zeilen 11 bis 12[889] eine Routine an, die er unter dem Namen „**PP**“ zur Verfügung stellt.

Daran anschließend wird für Zeile 13 ein Klassenobjekt und die dazugehörenden Methodenobjekte aus den entsprechenden Direktiven gebildet, die sich in den Zeilen 14 bis 26 finden. Dem frisch gebildeten Klassenobjekt wird die Klassenmethode „**INIT**“ gesandt.

Daran anschließend wird der Programmablauf am Anfang des Programms mit dem Kommentar fortgeführt und endet mit Zeile 10, nachdem danach die erste – zu diesem Zeitpunkt bereits abgearbeitete – Direktive beginnt.

---

887 Die Instanzmethode „NEW“ ist in der Object Rexx Metaklasse „.Class“ definiert und steht daher seinen Klassenobjekten zur Verfügung.

888 Diese Routine ist privat, da der Zusatz „PUBLIC“ in der entsprechenden Direktive fehlt und ist damit nur innerhalb dieses Programmes sichtbar.

889 Die Programmanweisungen von Direktiven enden an der nächsten Direktive oder am Ende der Programmdatei.

```
      /* This is a Rexx-comment and must always start at the 1st line/1st column */
[1]   a = 1
[2]   b = "2"

[3]   SAY "a:" a "+ b:" b "=" a + b                                  /* yields: 3 */
[4]   .output ~ SAY( "a:" a "+ b:" b "=" a ~ "+"( b ) ) /* the OO-way, yields: 3 */

[5]   SAY "--- now objects get created ---"

[6]   one = .test ~ new( "Max" )
[7]   two = .test ~ new( "Moritz" )

[8]   SAY "There have been" .test ~ counter "instances of class" .test ~ id "created".
[9]   SAY one ~ name       "und"     two ~ name   "diese beiden ..."
[10]  SAY pp( one ~ name ) "und" pp( two ~ name ) "diese beiden ..."

[11]  :: ROUTINE pp                           /* "poor man's pretty print ..." */
[12]     RETURN "[" || ARG( 1 ) || "]"

[13]  :: CLASS test PUBLIC
                  /* class methods    */
[14]  :: METHOD Init CLASS  /* called right after the class-object was created        */
[15]     SAY  "Hi, I am the INIT-Class Method, the class object was just created..."
[16]     self ~ counter = 0 /* initialize class attribute "counter" to "0"            */

[17]  :: METHOD counter CLASS ATTRIBUTE       /* this defines a class attribute       */

[18]  :: METHOD new CLASS   /* called when an instance of this class is to be created */
[19]     EXPOSE counter     /* establish direct access to class attribute "counter"   */
[20]     counter = counter + 1    /* increase counter                                  */
[21]     FORWARD CLASS ( super )  /* let superclass proceed with initialization       */

[22]              /* instance methods */
[23]  :: METHOD Init
[24]     EXPOSE  name       /* establish direct access to instance attribute "name"   */
[25]     USE ARG name       /* retrieve object directly into the instance attribute   */

[26]  :: METHOD name ATTRIBUTE                /* this defines an instance attribute   */
```

Programmcode 6-11: Beispiel für ein Object Rexx-Programm

Das Object Rexx-Programm im vorstehenden Programmcode 6-11 liefert daher folgende Ausgaben:

```
Hi, I am the INIT-Class Method, the class object was just created...
a: 1 + b: 2 = 3
a: 1 + b: 2 = 3
--- now objects get created ---
There have been 2 instances of class TEST created.
Max und Moritz diese beiden ...
[Max] und [Moritz] diese beiden ...
```

6.2.2.2 Definition eines Object Rexx-Moduls zum Zugriff auf und Einfügen von Meta-Modelldefinitionen in die dem EIA/CDIF-Meta-Meta-Modell entsprechenden ORACLE-Tabellen („M3SQL.CLS“)

In diesem Abschnitt werden die Klassen- und Routinendeklarationen des Object Rexx-Moduls „M3SQL.cmd“ im Programmcode 6-12 dokumentiert. Es benötigt den Zugriff auf Klassen beziehungsweise Routinen, die in den Modulen „rxSQLutil.cmd“ und „sort_Util.cmd“ festgelegt sind.[890]

Zunächst wird eine Klassenhierarchie definiert, die das EIA/CDIF-Meta-Meta-Modell abbildet. In der Initialisierungsmethode für die Klassenobjekte werden in Klassenattributen (Klassenvariablen) der Name der entsprechenden relationalen Tabelle, eine Liste der Bezeichner für die Meta-Meta-Attribute, die darauf abgestimmten „SELECT“- und „INSERT“-Anweisungen[891] und die Datentypen der relationalen Spalten der Meta-Meta-Attribute gespeichert.

Für jede Klasse werden Klassenattribute definiert, die unter anderem die Bezeichnung der entsprechenden relationalen Tabelle und die Namen der entsprechenden Meta-Meta-Attribute beinhalten, wobei letztere auch als Spaltenbezeichner in den entsprechenden relationalen Tabellen vorkommen. In der Initialisierungsmethode auf Klassenebene der Wurzelklasse „M0“ wird für jede Klasse mit Hilfe der Routine „make_sql_stmts“ eine entsprechende INSERT- und SELECT-SQL-Anweisung mit numerischen Platzhaltern[892] definiert.

---

890 Das Object Rexx-Modul „rxSQLutil.cmd“ ist im folgenden Abschnitt 6.2.2.3, auf Seite 475 unten, dokumentiert, das Object Rexx-Modul „sort_Util.cmd“ ist Bestandteil des über das Internet beziehbare Archiv der Object Rexx Utility-Funktionen in [Flat96c] beziehungsweise [Flat96d].

891 In der Object Rexx-Klasse „M0“, die den EIA/CDIF-Meta-Meta-Entitätstyp „MetaObject“ repräsentiert, wird eine Klassenmethode namens „prep_insert“ vorgesehen, mit deren Hilfe – für eine beschleunigte Interaktion mit den entsprechenden relationalen Datenbanken – die SQL-Anweisungen auch von der Datenbank präpariert (englisch: „prepared“) und auf Klassenebene gespeichert werden. Im Object Rexx-Programm „CTI2SQL.cmd“, das im Abschnitt 6.2.2.4.2, Seite 482 unten, wird diese Methode im Initialisierungsteil für die Klassenobjekte aufgerufen.
Mit der Klassenmethode „dispose“ werden sämtliche Arbeitsspeicherbereiche für die präparierten SQL-Anweisungen ordnungsgemäß freigegeben.

892 Numerische Platzhalter werden de facto von allen relationalen Datenbankverwaltungssystemen unterstützt, mit denen díe entsprechenden – von dem Australier Mark Hessling entwickelten – „RexxSQL“-Rexx-Funktionspakete (siehe weiter oben) interagieren können.

Sämtliche weiteren Klassendefinitionen ererben direkt oder indirekt diese Methoden über die Generalisierungshierarchie von Object Rexx. Die weiteren Klassendefinitionen für EIA/CDIF-Meta-Meta-Entitätstypen unterscheiden sich im wesentlichen dadurch, daß die Klasseninitialisierungsmethode „`INIT`“ die entsprechenden Meta-Meta-Attribute als Liste im Klassenattribut „`attrib_list`“ speichern, sowie durch die Definition von Klassenattributen mit Hilfe von Attributmethoden, um die entsprechenden Meta-Meta-Attribute zu repräsentieren.

Die folgende Liste skizziert prägnant die weiteren Routinendeklarationen, die im folgenden Programmcode 6-12 enthalten sind:

Routine	Beschreibung
`check4null`	Liefert eine leere Zeichenkette für das Argument „`.nil`“, sonst das Argument selbst.
`DELETE_SA`	Öffentliche Routine, löscht den angegebenen GegenstandsBereich und die entsprechenden Beziehungen zu den SammelbarenMetaObjekten.
`INSERT_ROW`	Überprüft, ob das gewünschte Meta-Objekt bereits in der Datenbank existiert und benützt es wieder für den neuen GegenstandsBereich, ansonsten wird das entsprechende Meta-Objekt neu angelegt; wird von der Klassenmethode „`insert`“ aufgerufen.
`INSERT_REL`	Öffentliche Routine, fügt die Beziehung in die angegebene Tabelle ein, die einen EIA/CDIF-Meta-Meta-Beziehungstyp repräsentiert, die Quell- und Ziel-Meta-Objekttypen werden als Implementierungs-Surrogate repräsentiert.
`NEXT_SURR`	Öffentliche Routine, die den bisher höchsten, vergebenen Surrogatwert feststellt und liefert einen um eins er-

höhten und damit noch nicht vergebenen Surrogatwert zurück.[893]

PP Liefert den als Argument übergebenen Wert in eckigen Klammern eingeschlossen zurück.

Der folgende Programmcode 6-12 dokumentiert das Modul „M3SQL.CLS“.

893 Sofern ein Einfügevorgang in die EIA/CDIF-Meta-Meta-Modelltabellen im Mehrbenutzerbetrieb geplant ist, müßte eine eigene und damit über Sitzungen hinweg sperrbare Tabelle angelegt werden, die als Spalte und als einzigen Datensatz den letzten vergebenen Surrogatwert beinhaltet. Wird ein neuer Surrogatwert erstellt, muß dieser Datensatz sofort entsprechend geändert werden.
Für die Oracle-Implementierung bieten sich auch die ORACLE-Sequences an, hätten aber die Implementierung nichtportabel gemacht.

```
/*
   author:        Rony G. Flatscher
   date:          97-09
   purpose:       read from RGF-CDIF-Interchange-file format and put meta-model data
                  into Database
   pre-requisite: needs an error free x2i-file (processed e.g. with x2i.cmd)
   usage:         require it
   needs:         Mark Hessling's "RexxSQL" interface (available for OS/2, Unix, Windows95/NT),
                  initialised and connected
*/

:: REQUIRES rxSQLutil.cmd       /* Mark Hessling's RexxSQL routines     */
:: REQUIRES sort_Util.cmd       /* get access to sort-routines          */

/* ---------------------------------------------------------------------------- */
:: CLASS MO     PUBLIC
:: METHOD INIT CLASS
        /* define list of attributes */
   self ~ attrib_list = .list ~ of( "SURR", "MO_TYPE", "ALIASES", "CDIFMetaIdentifier", ,
                                    "Constraints", "Description", "Name", "LongName", "Usage" )
   self ~ table_name = "MO_"            /* Base table name              */
   CALL make_sql_stmts self             /* generate SQL statements      */

:: METHOD attrib_list CLASS ATTRIBUTE   /* list of MMAs                                 */
:: METHOD dt_stem     CLASS ATTRIBUTE   /* stem-array of datatypes                      */
:: METHOD sql_insert  CLASS ATTRIBUTE   /* insert statement                             */
:: METHOD sql_select  CLASS ATTRIBUTE   /* select statement                             */
:: METHOD stmt_insert CLASS ATTRIBUTE   /* name of prepared insert-statement            */
:: METHOD stmt_select CLASS ATTRIBUTE   /* name of prepared select-statement            */
:: METHOD table_name  CLASS ATTRIBUTE   /* table name, also used as the cursor name     */

:: METHOD prep_insert CLASS             /* connect and prepare insert statement */
   EXPOSE sql_insert sql_select table_name stmt_select stmt_insert
   stmt_insert = table_name || "I"      /* create unique statement name for insertions  */
   IF sqlprepare( stmt_insert , sql_insert ) < 0 THEN
      rxSQL.Abort( 'prepare:' stmt_insert , , sqlca.)

        /* create datatype definition stem dynamically, using the SELECT-statement with
           the same order of columns as the INSERT-statement */
   stmt_select = table_name || "S"      /* create unique statement name for selects     */
   IF sqlprepare( stmt_select , sql_select  ) < 0 THEN
      rxSQL.Abort( 'prepare:' pp( stmt_select ), , sqlca.)
   IF sqldescribe( stmt_select , "AHA"  ) < 0 THEN      /* get datatypes from RDBMS     */
      rxSQL.Abort( 'describe:' pp( stmt_select ), , sqlca.)

   new_dt. = ""
   DO i = 1 TO AHA.COLUMN.TYPE.0
      new_dt.i = aha.COLUMN.TYPE.i
   END
   new_dt.0 = ( i - 1 )         /* # of elements                */
   self ~ dt_stem = new_dt.     /* assign data-type stem-array to class variable             */

:: METHOD dispose     CLASS     /* dispose of working-areas of insert- and select-statements */
   EXPOSE stmt_insert stmt_select

   IF sqlDispose( stmt_insert ) < 0 THEN rxSQL.Abort('dispose:' pp( stmt_insert ), , sqlca.)
   IF sqlDispose( stmt_select ) < 0 THEN rxSQL.Abort('dispose:' pp( stmt_select ), , sqlca.)
```

```
:: METHOD insert   CLASS        /* insert a record into RDBMS   */
   USE ARG valueDir
   RETURN insertRow( valueDir, .mo )

/* ------- -------- INSTANCE methods -------- -------- */
:: METHOD INIT
   USE ARG tmpDir                       /* Directory to contain                        */
                                        /* set the object variables from directory     */
   DO item OVER .mo ~ attrib_list
      .message ~ new( self, item || "=", "I", tmpDir ~ entry( item ) ) ~ send   /* set value */
   END
                                        /* Define Attribute methods     */
:: METHOD ALIASES               ATTRIBUTE
:: METHOD CDIFMetaIdentifier    ATTRIBUTE
:: METHOD Constraints           ATTRIBUTE
:: METHOD Description           ATTRIBUTE
:: METHOD LongName              ATTRIBUTE
:: METHOD MO_TYPE               ATTRIBUTE
:: METHOD Name                  ATTRIBUTE
:: METHOD SURR                  ATTRIBUTE
:: METHOD Usage                 ATTRIBUTE

:: METHOD SortValue                     /* Value to sort after          */
   RETURN self ~ Name

/* ---------------------------------------------------------------------------- */
:: CLASS SA             SUBCLASS MO     PUBLIC
:: METHOD INIT CLASS
        /* define list of attributes */
   self ~ attrib_list = .list ~ of( "SURR", "ShortHand" , "CDIFNumber", "VersionNumber" )
   self ~ table_name = "SA_"            /* Base table name              */
   CALL make_sql_stmts self             /* generate SQL statements      */

:: METHOD insert   CLASS                /* insert a record into RDBMS   */
   EXPOSE top
   USE ARG valueDir
   valueDir ~ MO_TYPE = "SA"            /* set type information         */
     /* make sure that top-down inserts take place, otherwise FK-constraints are violated ! */
   FORWARD CLASS( super ) CONTINUE
   IF RESULT THEN                       /* "RESULT" contains result of last return value    */
      RETURN insertRow( valueDir, .sa )
   ELSE RETURN .False

/* ------- -------- INSTANCE methods -------- -------- */
:: METHOD INIT
   USE ARG tmpDir                       /* Directory to contain */
                                        /* set the object variables from directory      */
   FORWARD CLASS ( super ) CONTINUE
   DO item OVER .sa ~ attrib_list
      .message ~ new( self, item || "=", "I", tmpDir ~ entry( item ) ) ~ send   /* set value */
   END
                                        /* make sure that an SA has a long name */
   IF self ~ LongName = .nil THEN self ~ LongName = self ~ Name
                                        /* Define Attribute methods     */
:: METHOD CDIFNumber            ATTRIBUTE       /* official EIA/CDIF-number of standard     */
:: METHOD SURR                  ATTRIBUTE
:: METHOD ShortHand             ATTRIBUTE       /* abbreviation used while developing SA    */
:: METHOD VersionNumber         ATTRIBUTE
```

```
/* ------------------------------------------------------------------------------ */
:: CLASS CMO            SUBCLASS MO     PUBLIC
:: METHOD INIT CLASS
        /* define list of attributes */
   self ~ attrib_list = .list ~ of( "SURR" )
   self ~ table_name = "CMO_"           /* Base table name              */
   CALL make_sql_stmts self

:: METHOD insert   CLASS                /* insert a record into RDBMS   */
   USE ARG valueDir
      /* make sure that top-down inserts take place, otherwise FK-constraints are violated ! */
   FORWARD CLASS( super ) CONTINUE
   IF RESULT THEN                       /* "RESULT" contains result of last return value     */
      RETURN insertRow( valueDir, .cmo )
   ELSE RETURN .False

/* ------- -------- INSTANCE methods -------- -------- */
:: METHOD INIT
   USE ARG tmpDir                       /* Directory to contain */
                                        /* set the object variables from directory      */
   FORWARD CLASS ( super ) CONTINUE
   DO item OVER .cmo ~ attrib_list
      .message ~ new( self, item || "=", "I", tmpDir ~ entry( item ) ) ~ send   /* set value */
   END
   self ~ bDefinedInSA = ( tmpDir ~ bDefinedInSA )      /* explicitly used in SA        */
                                     /* Define Attribute methods     */
:: METHOD bDefinedInSA          ATTRIBUTE
:: METHOD SURR                  ATTRIBUTE
```

```
/* ------------------------------------------------------------------------------ */
:: CLASS MA             SUBCLASS CMO    PUBLIC

:: METHOD INIT CLASS
   EXPOSE attrib_list
        /* define list of attributes */
   self ~ attrib_list = .list ~ of( "SURR", "DataType", "Domain", "IsOptional", "Length" )
   self ~ table_name = "MA_"    /* Base table name      */
   CALL make_sql_stmts self

:: METHOD insert   CLASS                /* insert a record into RDBMS   */
   USE ARG valueDir
   valueDir ~ MO_TYPE = "MA"            /* set type information         */
      /* make sure that top-down inserts take place, otherwise FK-constraints are violated ! */
   FORWARD CLASS( super ) CONTINUE
   IF RESULT THEN                       /* "RESULT" contains result of last return value     */
      RETURN insertRow( valueDir, .ma )
   ELSE RETURN .False

/* ------- -------- INSTANCE methods -------- -------- */
:: METHOD INIT
   USE ARG tmpDir                       /* Directory to contain */
                                        /* set the object variables from directory      */
   FORWARD CLASS ( super ) CONTINUE
   DO item OVER .ma ~ attrib_list
      .message ~ new( self, item || "=", "I", tmpDir ~ entry( item ) ) ~ send   /* set value
*/
   END
   self ~ BelongsTo = .nil              /* initialise to .nil           */
                                        /* Define Attribute methods     */
:: METHOD BelongsTo             ATTRIBUTE       /* points to AMO, MA belongs to */
:: METHOD DataType              ATTRIBUTE
:: METHOD Domain                ATTRIBUTE
:: METHOD IsOptional            ATTRIBUTE
:: METHOD Length                ATTRIBUTE
:: METHOD SURR                  ATTRIBUTE

:: METHOD SortValue                     /* Value to sort after          */
   RETURN self ~ BelongsTo ~ LongName || ":" self ~ name
```

```
/* ------------------------------------------------------------------------------ */
:: CLASS AMO            SUBCLASS CMO    PUBLIC
:: METHOD INIT CLASS
        /* define list of attributes */
   self ~ attrib_list = .list ~ of( "SURR" )
   self ~ table_name = "AMO_"           /* Base table name      */
   CALL make_sql_stmts self

:: METHOD insert   CLASS                /* insert a record into RDBMS   */
   USE ARG valueDir
   IF valueDir ~ MO_TYPE = .nil THEN    /* RootObject in hand ?         */
      valueDir ~ MO_TYPE = "AMO"        /* set type information         */
      /* make sure that top-down inserts take place, otherwise FK-constraints are violated ! */
   FORWARD CLASS( super ) CONTINUE
   IF RESULT THEN                       /* "RESULT" contains result of last return value     */
      RETURN insertRow( valueDir, .amo )
   ELSE RETURN .False

/* ------- -------- INSTANCE methods -------- -------- */
:: METHOD INIT
   USE ARG tmpDir                       /* Directory to contain */
                                        /* set the object variables from directory      */
   FORWARD CLASS ( super ) CONTINUE
   DO item OVER .amo ~ attrib_list
      .message ~ new( self, item || "=", "I", tmpDir ~ entry( item ) ) ~ send   /* set value */
   END
   self ~ SubtypeOf = tmpDir ~ SubtypeOf
                                        /* make sure that all AMOs have a long name     */
   IF self ~ LongName = .nil THEN self ~ LongName = self ~ Name
   self ~ localMA = .relation ~ new     /* create an empty relation for local MAs       */
                                        /* Define Attribute methods     */
:: METHOD LocalMA               ATTRIBUTE       /* LocalMA[ name ] = MA-object  */
:: METHOD SURR                  ATTRIBUTE
:: METHOD SubtypeOf             ATTRIBUTE       /* string of Supertypes         */

:: METHOD SortValue                     /* Value to sort after          */
   RETURN self ~ LongName
```

```
/* ------------------------------------------------------------------------------ */
:: CLASS ME             SUBCLASS AMO    PUBLIC
:: METHOD INIT CLASS
     /* define list of attributes    */
   self ~ attrib_list = .list ~ of( "SURR", "Type" )
   self ~ table_name = "ME_"            /* Base table name             */
   CALL make_sql_stmts self

:: METHOD insert   CLASS                /* insert a record into RDBMS   */
   USE ARG valueDir
   valueDir ~ MO_TYPE = "ME"            /* set type information        */
      /* make sure that top-down inserts take place, otherwise FK-constraints are violated ! */
   FORWARD CLASS( super ) CONTINUE
   IF RESULT THEN                       /* "RESULT" contains result of last return value     */
      RETURN insertRow( valueDir, .me )
   ELSE RETURN .False

/* ------- -------- INSTANCE methods -------- -------- */
:: METHOD INIT
   USE ARG tmpDir                       /* Directory to contain */
                                        /* set the object variables from directory      */
   FORWARD CLASS ( super ) CONTINUE
   DO item OVER .me ~ attrib_list
      .message ~ new( self, item || "=", "I", tmpDir ~ entry( item ) ) ~ send   /* set value */
   END
                                        /* Define Attribute methods     */
:: METHOD SURR                  ATTRIBUTE
:: METHOD Type                  ATTRIBUTE
```

```
/* ------------------------------------------------------------------------------ */
:: CLASS MR              SUBCLASS AMO    PUBLIC
:: METHOD INIT CLASS
                                        /* define list of attributes */
   self ~ attrib_list = .list ~ of( "SURR", "MinSourceCard", "MaxSourceCard", ,
                                           "MinDestCard"  , "MaxDestCard" )
   self ~ table_name = "MR_"            /* Base table name      */
   CALL make_sql_stmts self

:: METHOD insert   CLASS                /* insert a record into RDBMS   */
   USE ARG valueDir
   valueDir ~ MO_TYPE = "MR"            /* set type information         */
      /* make sure that top-down inserts take place, otherwise FK-constraints are violated ! */
   FORWARD CLASS( super ) CONTINUE
   IF RESULT THEN                       /* "RESULT" contains result of last return value     */
      RETURN insertRow( valueDir, .mr )
   ELSE RETURN .False

/* ------- -------- INSTANCE methods -------- -------- */
:: METHOD INIT
   USE ARG tmpDir                       /* Directory to contain */
                                        /* set the object variables from directory      */
   FORWARD CLASS ( super ) CONTINUE
   DO item OVER .mr ~ attrib_list
      .message ~ new( self, item || "=", "I", tmpDir ~ entry( item ) ) ~ send   /* set value */
   END
                                        /* Define Attribute methods     */
:: METHOD MinDestCard           ATTRIBUTE
:: METHOD MaxDestCard           ATTRIBUTE
:: METHOD MinSourceCard         ATTRIBUTE
:: METHOD MaxSourceCard         ATTRIBUTE
:: METHOD SURR                  ATTRIBUTE
```

```
/* ------------------------------------------------------------------------------ */
/* inserts the appropriate (determined by clsObject) fragment into the RDBMS;
   returns .False, if MO exists already, .True if insertions are o.k.   */
/* insert row into database     */
:: ROUTINE insertRow
   USE ARG tmpObject, clsObject

   IF clsObject = .mo THEN      /* check on first invocation per tmpObj only !  */
   DO
      tmpCID = ( tmpObject ~ CDIFMetaIdentifier )
      sql_check  = "SELECT SURR, CDIFMetaIdentifier, Name, MO_Type, Longname" ,
                   "FROM   MO_"                                              ,
                   "WHERE  CDIFMetaIdentifier = '" || tmpCID || "'"
      IF sqlcommand( "query", sql_check ) < 0 THEN      /* execute statement    */
         rxSQL.Abort('command: insertRow() -' sql_check, , sqlca.)
      rowNum = query.SURR.0     /* get # of retrieved records                   */

      IF rowNum > 0 THEN        /* MO with CDIFMetaIdentifier exists already    */
      DO
         tmpMO_Type = tmpObject ~ mo_type
         tmpName    = tmpObject ~ name
         DO i = 1 TO query.SURR.0    /* loop, see whether duplicate CDIFMetaIdentifier error */
            IF query.MO_TYPE.i = tmpMO_Type THEN        /* O.K., of same type   */
            DO
               IF ( POS( tmpMO_Type, "ME MR AMO") > 0 ) THEN    /* really same MO ?     */
                  bReuse = ( query.LONGNAME.i = tmpObject ~ Longname )
               ELSE IF tmpMO_Type = "MA" THEN           /* oh, an MA-duplicate !        */
               DO       /* an error or related to same AMO (with same CDIFMetaIdentifier) ? */
                        /* get AMO this MA belongs to:  */
                   tmpAMO = .xfer_cdif ~ IsLocalMetaAttributeOf[ tmpObject ]
                   /* does MA exist already in the database and is it assigned to the same
                      AMO ?     */
                    sql_check_amo = ,
     "SELECT SURR, CDIFMetaIdentifier, Name, MO_Type, Longname "                          ,
     "FROM   AMO "                                                                       ,
     "WHERE      "                                                                       ,
     "       CDIFMETAIdentifier = ‘" || tmpAmo ~ CDIFMetaIdentifier || "’"               ,
     "       AND"                                                                        ,
     "       LONGNAME           = ‘" || tmpAmo ~ LongName           || "’"               ,
     "       AND"                                                                        ,
     "       SURR IN ( SELECT DESTINATION FROM   IsLocalMetaAttributeOf_ "               ,
     "                 WHERE  SOURCE IN ( SELECT SURR FROM MA"                           ,
     "                                    WHERE  CDIFMetaIdentifier = ‘" || tmpCID || "‘ AND" ,
     "                                           NAME = '" || tmpName || "' ) ) "

                   IF sqlcommand( "queryAMO", sql_check_amo ) < 0 THEN /* execute statement */
                      rxSQL.Abort('command: insertRow() -' sql_check_amo, , sqlca.)
                   bReuse = ( queryAMO.SURR.0 > 0 )     /* if AMO exists also, then reuse MA */
               END
               ELSE     /* an SA in hand        */
                  bReuse = ( query.NAME.i = tmpName )

               IF bReuse THEN           /* o.k., inferring: same MO, hence reuse it     */
               DO
                  tmpObject ~ surr = query.SURR.1  /* use SURR of MO as stored in RDBMS */
                  SAY "             (MO exists already, gets reused)"
                  RETURN .False         /* no insertion needed, object exists already   */
               END
            END
         END
```

```
      tmpString = ""
      IF ( POS( tmpMO_Type, "ME MR AMO") > 0 ) THEN
         tmpString = "LongName" pp( tmpObject ~ longname )
      .error ~ SAY

      IF tmpMO_Type = "MA" THEN
      DO
         IF \ VAR( "TMPAMO" ) THEN   /* AMO there, if not get it     */
                tmpAMO = .xfer_cdif ~ IsLocalMetaAttributeOf[ tmpObject ]

         tmpString = "MA" pp( tmpName ) pp( tmpObject ~ CDIFMetaIdentifier ),
                     "SURR" pp( tmpObject ~ surr ) "of AMO" pp( tmpAMO ~ LongName ),
                     pp( tmpObject ~ CDIFMetaIdentifier ) "SURR" pp( tmpAMO ~ surr )
      END
      ELSE
         tmpString = tmpMO_Type pp( tmpObject ~ LongName ) ,
                     pp( tmpObject ~ CDIFMetaIdentifier ) "SURR" pp( tmpObject ~ surr )
      .error ~ SAY( "..." "Duplicate CDIFMetaIdentifier:" pp( tmpCID ) )
      .error ~ SAY( "   " "    therefore" tmpString    )
      .error ~ SAY( "   " "is in conflict with:"        )

      DO i = 1 TO query.SURR.0       /* list all duplicate usages    */
         tmpString = pp( query.NAME.i )
         IF ( POS( query.MO_Type.i, "ME MR AMO") > 0 ) THEN
            tmpString = pp( query.LONGNAME.i )
         .error ~ SAY( "   " "    >> existing MO of type" pp( query.MO_TYPE.i ),
                tmpString pp( query.CDIFMetaIdentifier.i ) ,
                "SURR" pp( query.SURR.i )  )
      END
      .error ~ SAY( "   " "MO inserted with duplicate [CDIFMetaIdentifier] anyway." )
      .error ~ SAY
   END
END

stmt = clsObject ~ stmt_insert       /* get appropriate prepared insert-statement */
dt. = clsObject ~ dt_stem            /* assign datatype-stem to local stem   */
dv. = ""                             /* stem to contain values       */
i = 0                                /* fill value-stem              */
DO item OVER clsObject ~ attrib_list /* iterate in the attrib_list order     */
   i = i + 1
   dv.i = check4null( .message ~ new( tmpObject, item ) ~ send )   /* get value    */
END
dv.0 = i
IF sqlexecute( stmt , ".", "DV." ) < 0 THEN rxSQL.Abort('execute:' stmt, , sqlca.)
RETURN .True              /* indicate that insertions along the inheritance tree is o.k. */
```

```
/* routine to create SQL-statements according to available class attribute methods  */
:: ROUTINE make_sql_stmts
   USE ARG object
   column = ""                  /* columns of SQL-insert statement      */
   values = ""                  /* values of SQL-insert statement       */
   i = 0
   DO item OVER object ~ attrib_list    /* iterate over class attribute methods */
      i = i + 1                 /* create column-names                          */
      IF column = "" THEN column = item
                     ELSE column = column "," item
                                /* create number-(=standard)-placeholder        */
      IF values = "" THEN values = ":" || i
                     ELSE values = values ", :" || i
   END
                                     /* create and save insert-SQL-statement with object  */
   object ~ sql_insert = "INSERT INTO" object ~ table_name ,
                                   "(" column ") VALUES (" values ") "
                                     /* create and save select-SQL-statement with object  */
   object ~ sql_select = "SELECT" column "FROM" object ~ table_name "WHERE SURR = :1"
   RETURN

:: ROUTINE pp; RETURN "[" || ARG( 1 ) || "]"    /* "poor man's pretty-print ;-) */

/* return the next unique surrogate value       */
:: ROUTINE NEXT_SURR    PUBLIC

   IF .m3sql.surr = ".M3SQL.SURR" THEN      /* not set in .local as of yet  */
   DO
      /* get highest value of SURR out of table MO and all tables representing MMR */
      query = 'SELECT MAX( SURR ) "MAX" FROM mo                      UNION'  ,
              'SELECT MAX( SURR ) "MAX" FROM IsUsedIn_               UNION'  ,
              'SELECT MAX( SURR ) "MAX" FROM IsLocalMetaAttributeOf_ UNION'  ,
              'SELECT MAX( SURR ) "MAX" FROM HasSubtype_             UNION'  ,
              'SELECT MAX( SURR ) "MAX" FROM HasSource_              UNION'  ,
              'SELECT MAX( SURR ) "MAX" FROM HasDestination_              '  ,
              'ORDER BY 1 DESC                                            '

      IF sqlcommand( "tmp", query ) < 0 THEN    /* execute statement    */
         rxSQL.Abort('command: executing', , sqlca.)

      IF DATATYPE( tmp.max.1, "W" ) THEN
      DO
         .local ~ m3sql.surr = tmp.max.1 + 1    /* save to .local       */
      END
      ELSE                      /* no values so far, start out with "1" */
      DO
         .local ~ m3sql.surr = 1                /* save to .local       */
      END
   END
   ELSE
   DO
      .local ~ m3sql.surr = .m3sql.surr + 1     /* save to .local       */
   END
   RETURN .m3sql.surr                           /* return new highest value for "SURR" */

/* return an empty string, if .nil      */
:: ROUTINE check4null
   IF ARG( 1 ) = .nil THEN RETURN ""            /* return empty string  */
                      ELSE RETURN ARG( 1 )
```

```
/* insert into a table representing a relationship-type;
   note, if versioning is planned, the logic for checking for duplicates has to change ! */
:: ROUTINE INSERT_REL           PUBLIC
   USE ARG table_name, Source, Destination
   /* check whether it exists already, if so do not insert      */
   sql_check = "SELECT SURR FROM" table_name "WHERE SOURCE      =" Source      "AND",
                                             "      DESTINATION =" Destination
   IF sqlcommand( "QUERY" , sql_check ) < 0 THEN        /* execute statement            */
      rxSQL.Abort('command:' "INSERT_REL: sql_check", , sqlca.)
   IF query.surr.0 > 0 THEN RETURN                      /* exists already, no need to add */
        /* Insert into table    */
   sql_insert = "INSERT INTO" table_name "( SURR, SOURCE, DESTINATION ) VALUES" ,
                              "(" next_surr() "," Source "," Destination ")"
   IF sqlcommand( "QUERY" , sql_insert ) < 0 THEN       /* execute statement            */
      rxSQL.Abort('command:' "INSERT_REL: sql_insert", , sqlca.)
   RETURN

/* RETURN SURR for the given AMO-longName, if it does not exist, return .nil    */
:: ROUTINE GET_SURR_BY_LONGNAME PUBLIC
   USE ARG longname
   sql_find = "SELECT SURR FROM AMO WHERE LONGNAME = '" || longname || "'"
   IF sqlcommand( "QUERY" , sql_find ) < 0 THEN         /* execute statement            */
      rxSQL.Abort('command:' "GET_SURR_BY_LONGNAME: sql_find for" pp( longname ) , , sqlca.)
   IF query.surr.0 > 0 THEN RETURN query.surr.1 /* return SURR-value            */
   RETURN .nil                                  /* not found, return .nil       */
```

```
/* delete MO's of given SubjectArea     */
:: ROUTINE DELETE_SA    PUBLIC
   USE ARG SA_name
   query0 = "SELECT SURR FROM SA WHERE NAME = '" || SA_NAME || "'"
   IF sqlcommand( "query" , query0 ) < 0 THEN  /* execute statement     */
      rxSQL.Abort('command:' query0, , sqlca.)
   IF query.SURR.0 = 0 THEN RETURN      /* new SA, nothing to delete    */

   SA_SURR = query.surr.1               /* get SURR of SA               */
   SAY "Deleting SA" pp( SA_name ) "with SURR:" pp( SA_SURR ) ,
       "total SA's by that name" pp( query.SURR.0 )
        /* delete all relations of MO's belonging to that particular SA         */
        /* (a particular MO may belong to multiple SA's !)                      */
   query1 = "DELETE FROM IsUsedIn_ WHERE destination =" SA_SURR
   IF sqlcommand( "query" , query1 ) < 0 THEN           /* execute statement            */
      tmp = rxSQL.Abort('command:' query1, .True, sqlca.)

   SAY "... deleting" pp( "IsUsedIn" ) "affected" pp( sqlca.rowcount ) "rows."
        /* AMOs, die nicht in einer SA direkt benutzt werden und nicht Supertyp
           für welche sind, die in einer anderen SA benutzt werden */
   query2a = "DELETE FROM MO_ WHERE SURR IN (                              " ,
        "SELECT SURR                                                       " ,
        "FROM   AMO_                                                       " ,
        "WHERE  SURR NOT IN ( SELECT SOURCE FROM IsUsedIn_ ) AND           " ,
        "       SURR NOT IN ( SELECT SOURCE FROM HasSubtype_               " ,
        "                     WHERE Destination IN                         " ,
        "                           (SELECT SOURCE FROM IsUsedIn_ ) )      " ,
        ")                                                                 "
   IF sqlcommand( "query" , query2a ) < 0 THEN          /* execute statement            */
      tmp = rxSQL.Abort('command:' query2, .True, sqlca.)

   SAY "... deleting of leftover AMOs affected" pp( sqlca.rowcount ) "rows."
        /* MAs, die keinem AMO mehr zugeordnet sind                     */
   query2b = "DELETE FROM MO_ WHERE SURR IN (                            " ,
        "SELECT SURR FROM MA_ WHERE SURR NOT IN ( SELECT Source          " ,
        "                           FROM IsLocalMetaAttributeOf_ )       " ,
        ")                                                               "
   IF sqlcommand( "query" , query2b ) < 0 THEN          /* execute statement            */
      tmp = rxSQL.Abort('command:' query2, .True, sqlca.)
   SAY "... deleting of leftover MAs affected" pp( sqlca.rowcount ) "rows."
        /* delete SA itself     */
   query3 = "DELETE FROM MO_ WHERE SURR IN (SELECT SURR FROM SA_ WHERE MO_.SURR =" SA_SURR ")"
   IF sqlcommand( "query" , query3 ) < 0 THEN           /* execute statement            */
      tmp = rxSQL.Abort( 'prepare:' query3, .True, sqlca.)
   SAY "... deleting SA affected" pp( sqlca.rowcount ) "row."
   SAY

   CALL SQLCommit
   RETURN
```

Programmcode 6-12: Das Object-Rexx-Programm „M3SQL.CLS“ zur Definition von Klassen und Routinen zum Einfügen (und Abfragen) von Meta-Modelldaten in (aus) relationale(n) Datenbankverwaltungssysteme(n)

6.2.2.3 Definition eines Moduls für die vereinfachte Unterstützung von dynamischem SQL („`rxSQLutil.cmd`“)

In diesem Abschnitt werden die Routinendeklarationen des Object Rexx-Moduls „`rxSQLutil.cmd`“ im Programmcode 6-13 dokumentiert. Es handelt sich hierbei um Routinen, die aus Funktionen des Testprogrammes, „`tester.cmd`“, aus dem Rexx-Funktionspaket „RexxSQL“ des Australiers Mark Hessling stammen.[894, 895]

Die folgende Liste skizziert prägnant die Routinendeklarationen:

Routine	Beschreibung
`PP`	Liefert den als Argument übergebenen Wert in eckigen Klammern eingeschlossen zurück.
`rxSQL.connect`	Öffentliche Routine zum Etablieren einer Sitzung mit der gewünschten relationalen Datenbank, optional können die Werte für den Verbindungsaufbau in Umgebungsvariablen angegeben werden,[896] und zum Abfragen und Speichern – der grundlegenden in der Zieldatenbank verfügbaren Datentypen – in der Umgebung „`.local`“ unter dem Index „`rxSQL.desc`“.
`rxSQL.default`	Öffentliche Routine zum Festlegen der aktiven Datenbanksitzung (englisch: „default connection“), diese Sitzungsidentifikation kann anschließend mit dem Umgebungssymbol „`.RxSQL.default.conn`“ abgerufen werden.
`rxSQL.disconnect`	Öffentliche Routine zum Abmelden von der Datenbank.

894 In der Regel wurden die Funktionen und Prozeduren lediglich als öffentliche Routinen deklariert, damit sie von anderen Rexx-Programmen und Rexx-Modulen aus aufgerufen werden können.

895 Nachdem die Implementierungen bereits durchgeführt wurden, hat der Amerikaner John Blumel sein Object Rexx Framework für den allgemeinen Zugriff auf relationale Datenbankverwaltungssysteme mit Hilfe des RexxSQL-Funktionspaketes im Herbst 1997 vorgestellt. Wäre es etwas früher entstanden, würden die hier dokumentierten Implementierungen dieses Framework nutzen (vgl. hierzu [W3ORexx-SQL] sowie [W3Rexx-SQL]), zu dessen Entstehung der Autor in Form von Feedback ein wenig beitragen konnte.

896 Es handelt sich hierbei um die Umgebungsvariablen „`REXXSQL_USERNAME`“, „`REXXSQL_PASSWORD`“, „`REXXSQL_DATABASE`“ und „`REXXSQL_SERVER`“.

`rxSQL.initialise` Öffentliche Routine zum Laden des RexxSQL-Funktionspaketes und damit der Funktionen zum Interagieren mit relationalen Datenbanken über RexxSQL.

`rxSQL.finalise` Öffentliche Routine zum Entladen des RexxSQL-Funktionspaketes aus der Rexx-Sitzung.

`rxSQL.abort` Öffentliche Routine zum Erzeugen von Fehlermeldungen; die Object Rexx-Variante löst eine allgemeine Syntaxausnahme aus, die an den Aufrufer weitergeleitet wird.

Der folgende Programmcode 6-13 dokumentiert das Modul „`rxSQLutil.cmd`“.

```
/*
   author:      Mark Hessling, adapted for "M3SQL.CLS" by Rony G. Flatscher
   name:        rxSQLutil.cmd
   date:        97-09
   purpose:     Collect general routines from "Tester.cmd" into its own module.
*/

PARSE SOURCE os .

SELECT                  /* save name of op_sys-environment      */
  WHEN os = 'UNIX'  THEN .local ~ rxSQL.envname = 'SYSTEM'
  WHEN os = 'WIN32' THEN .local ~ rxSQL.envname = 'ENVIRONMENT'
  WHEN os = 'OS/2'  THEN .local ~ rxSQL.envname = 'OS2ENVIRONMENT'
  OTHERWISE SAY 'Unsupported platform'
END
.local ~ rxSQL.os = os  /* save operating system                */

:: ROUTINE PP; RETURN "[" || ARG( 1 ) || "]"

/* adapted from Mark Hessling's Connect() in Tester.cmd supplied with RexxSQL           */
:: ROUTINE rxSQL.connect      PUBLIC
   USE ARG id, sqlconnect
   IF VAR( "SQLCONNECT" ) & (sqlconnect ~ class = .stem ) THEN
      sqlconnect. = sqlconnect
   ELSE         /* try to retrieve logon-data from op_sys environment   */
   DO
      sqlconnect.1 = VALUE('REXXSQL_USERNAME',, .rxSQL.envname)
      sqlconnect.2 = VALUE('REXXSQL_PASSWORD',, .rxSQL.envname)
      sqlconnect.3 = VALUE('REXXSQL_DATABASE',, .rxSQL.envname)
      sqlconnect.4 = VALUE('REXXSQL_SERVER'  ,, .rxSQL.envname)
   END

   IF sqlconnect(id, sqlconnect.1, sqlconnect.2, sqlconnect.3, sqlconnect.4) < 0 THEN
      RxSQL.abort('connect', , sqlca.)
           /* query fundamental datatypes of database of this particular connection     */
   IF sqlgetinfo( id ,'DESCRIBECOLUMNS','desc.') < 0 Then       /* execute statement    */
      RxSQL.abort('rxSQL.connect: getting datatypes', , sqlca.)
   DO i = 1 TO desc.0
      width.i = LENGTH( desc.i )
   END
   widt.0 = desc.0
                               /* save description and widths for further use  */
   .local ~ rxSQL.desc = .array ~ of( desc., width. )
   RETURN

/* set default connection               */
:: ROUTINE RxSQL.default         PUBLIC
   PARSE ARG id
   IF SQLDefault( id ) < 0  THEN RxSQL.abort( "RxSQL.default:" pp( id ),  , sqlca. )
   .local ~ RxSQL.default.conn  = id    /* save id of default connection into .local */
   RETURN

/* adapted from Mark Hessling's Disconnect() in Tester.cmd supplied with RexxSQL        */
:: ROUTINE RxSQL.disconnect      PUBLIC
   PARSE ARG id
   IF sqldisconnect(id) < 0 THEN RxSQL.abort('disconnect', ,sqlca.)
   IF id = .rxSQL.default.conn THEN
      .local ~ setentry( "RXSQL.DEFAULT.CONN" ) /* remove default connection from .local     */
   RETURN
```

```
/* adapted from Mark Hessling's Initialise() in Tester.cmd supplied with RexxSQL        */
:: ROUTINE rxSQL.initialise      PUBLIC
   USE ARG bStandardPlaceMarkers
   .local ~ rxSQL.bStandardPlaceMarkers = ( bStandardPlaceMarkers = .True )
   dll.unix='./rexxsql.rxlib'
   dll.os2='REXXSQL'
   dll.win32='REXXSQL'
   dll.windowsnt='REXXSQL'
   dll.windows95='REXXSQL'
   PARSE SOURCE int_os method .
   IF int_os = 'OS/2' THEN int_os = 'OS2'; ELSE int_os = Translate(int_os)

   rc = RXFuncAdd('SQLLoadFuncs', dll.int_os, 'SQLLoadFuncs')
   CALL SqlLoadFuncs
   version = sqlvariable('VERSION')
   IF .rxSQL.bStandardPlaceMarkers THEN /* SQL-statements with standard placemarkers ?  */
   DO
      IF SQLVariable( "STANDARDPLACEMARKERS", 1 ) < 0 THEN
         RxSQL.abort( "rxSQL.initialise: standard placemarkers", , sqlca. )
   END
   RETURN 0

/* adapted from Mark Hessling's Finalise() in Tester.cmd supplied with RexxSQL          */
:: ROUTINE RxSQL.finalise        PUBLIC
   IF \ RxFuncQuery( "SqlDropFuncs" ) THEN      /* RexxSQL loaded ?                     */
   DO
      CALL SqlDropFuncs                /* make sure RexxSQL cleans up this instance    */
      .local ~ setentry( "RXSQL.DEFAULT.CONN" ) /* remove default connection string     */
   END
   RETURN

/* adapted from Mark Hessling's RxSQL.abort() in Tester.cmd supplied with RexxSQL       */
:: ROUTINE RxSQL.abort           PUBLIC
   USE ARG message, kontinue, sqlca.    /* note "USE" instead of "PARSE"                */
   .error ~ SAY( 'Error in' message )
   IF sqlca.intcode = -1 THEN
   DO
      .error ~ Say( 'SQLCODE:' sqlca.sqlcode )
      .error ~ Say( 'SQLERRM:' sqlca.sqlerrm )
      .error ~ Say( 'SQLTEXT:' sqlca.sqltext )
   END
   ELSE
   DO
      .error ~ Say( 'INTCODE:' sqlca.intcode )
      .error ~ Say( 'INTERRM:' sqlca.interrm )
   END

   IF kontinue = 1 THEN RETURN 1
   CALL RxSQL.finalise                  /* initiate interface clean-up  */
   SIGNAL ON SYNTAX                     /* activate signal trapping     */
   RAISE SYNTAX 40.1 ARRAY ( message )  /* "external routine failed"    */
   SYNTAX: RAISE PROPAGATE              /* re-issue in caller           */
```

Programmcode 6-13: Das Object-Rexx-Programm „`rxSQLutil.cmd`“

6.2.2.4 Überführung von Meta-Modelldaten aus dem CTI-Datenformat in ORACLE-Tabellen

In diesem Abschnitt wird ein Object Rexx-Programm dokumentiert, mit dessen Hilfe Meta-Modelldaten aus dem CTI-Austauschformat ausgelesen und direkt in die relationale Datenbank von ORACLE übertragen werden. Das CTI-Austauschformat, das dieses Object Rexx-Programm abarbeitet, wird im direkt folgenden Unterkapitel definiert.

6.2.2.4.1 BNF von dem „CTI“-Austauschformat für Meta-Modelldaten

Im Rahmen der Arbeiten mit EIA/CDIF wurde ein Austauschformat definiert, das ausschließlich für die vereinfachte Bearbeitung beziehungsweise den Austausch von EIA/CDIF-Meta-Modelldaten gedacht ist. Da die Definitionen der Meta-Meta-Attribute nicht alle EIA/CDIF-Datentypen in Anspruch nehmen, kann ein vereinfachtes, textbasiertes Austauschformat im Klartext festgelegt werden. Sämtliche Meta-Meta-Objektdefinitionen erfolgen hierbei genauso, wie sie in den EIA/CDIF-Standards textuell dargestellt sind.[897]

Die Definitionen für das CTI-Austauschformat[898] folgen der erweiterten BNF[899] von EIA/CDIF, wie sie in [CDIF94d] und [CDIF94e] gleichlautend festgelegt sind. Nach dem Studium der Konventionen und der Definitionen des CTI-Austauschformates, sollten die EIA/CDIF SYNTAX.1- und ENCODING.1-Festlegungen ohne Probleme verarbeitbar sein. EIA/CDIF legt für ihre BNF folgende Konventionen fest:[900]

---

897 Sämtliche Beschreibungen von Meta-Meta-Objekten in den EIA/CDIF-Meta-Modell-standards erfolgen textuell und können mit dem hier vorgeschlagenen Austauschformat abgebildet werden. So sind entsprechend für das Meta-Meta-Attribut „IsOptional“ die Zeichenketten „True“ und „False“ vorgesehen, die aus `<GeneralPrintableChar>`-Zeichen bestehen.

898 Das Akronym „CTI“ stammt aus der ursprünglichen Arbeitsbezeichnung des Autors: „**C**DIF **T**extbased **I**nterchange“ Format für EIA/CDIF-Meta-Modelle.

899 Das Akronym „BNF“ leitet sich bekanntlich aus den Anfangsbuchstaben von „**B**ackus-**N**aur-**F**orm“ ab.

900 Vgl. in diesem Zusammenhang auch [CDIF94d], Seiten fünf bis sechs, und [CDIF94e], Seiten vier bis fünf.

Symbol	Bedeutung
`< >`	Spitze Klammern umfassen die Bezeichner für syntaktische Elemente.
`::=`	Definitionsoperator, mit dessen Hilfe die auf der rechten Seite stehende Produktionsregel dem syntaktischen Element auf der linken Seite zugeordnet wird.
`[ ]`	Eckige Klammern umfassen optionale Ausdrücke.
`{ }`	Geschwungene Klammern umfassen Ausdrücke, die damit als ein syntaktisches Element aufgefaßt werden können.
`|`	Dieses Zeichen dient als Alternativoperator und erlaubt es, eine Liste von syntaktischen Elementen zu definieren, von denen eines ausgewählt wird.
`...`	Ellipse, die anzeigt, daß der unmittelbar davor stehende Ausdruck beliebig oft wiederholt werden kann.[901]
`! !`	Dies ist der Kommentaroperator, alles was rechts davon steht, dient zur Erläuterung der Produktionsregeln. Sofern es sinnvoll erscheint, wird die Produktionsregel vollständig von einem Kommentar (Erläuterung) ersetzt und legt diese damit fest.
`<TokenName>`	Ein in spitzen Klammern eingeschlossenes, fett dargestelltes syntaktisches Element wird nicht weiter zerlegt. Die detaillierte Verkodierung wird im Verkodierungsteil („Encoding“) angegeben.
`Terminal`	Fett dargestellte Zeichen, die nicht in spitze Klammern eingeschlossen sind, stellen Literale dar.

901 In [CDIF98b], Seite 11 oben, wird mit „+“ ein zusätzlicher Operator eingeführt, der aus SGML stammt und in weiterer Folge auch in XML enthalten ist. Der Ausdruck, der vor diesem Operator steht, *muß* mindestens einmal und kann beliebig oft vorkommen. Die Elllipse „...“ dagegen ist optional und kann beliebig oft vorkommen und würde in SGML (XML) durch einen Stern „*“ repräsentiert werden.

x..y Bereichsanzeige, die sämtliche Zeichen umfaßt, die in den Bereich mit „x“ als Untergrenze und „y“ als Obergrenze fallen.

Die Syntaxdefinitionen für das CTI-Austauschformat findet sich in der folgenden Abbildung 6-2.

```
<CTI_ExchangeFormat>     ::= {   <Whitespace> <EOL> | <Comment> | <CTI_Line> } ...
<CTI_Line>               ::= {   <MO_Type_Def> <EOL> { <MetaMetaAttributeDef> <EOL> }... }
                                 ...
<MO_Type_Def>            ::= <AMO_Type_Def> | <MA> | <SA>
<AMO_Type_Def>           ::= {   <AMO> | <ME> | <MR> } <delimiter> <Is_Used_In> <delimiter>
                                 <Fully_Qualified_Name>
<MetaMetaAttributeDef>   ::= <Delimiter> <AttributeName> <Delimiter> <AttributeValue>
<AttributeName>          ::= <Identifier>
<AttributeValue>         ::= <GeneralPrintableChar>...
```

Abbildung 6-2: Die Definition der Syntax für das CTI-Austauschformat in EIA/CDIFs BNF-Version

Die Definition der Verkodierung für das CTI-Austauschformat findet sich in der folgenden Abbildung 6-3.

```
<AMO>                    ::= AMO
<Comment>                ::= <SemiColon> { <GeneralPrintableChar> | <Whitespace> }... <EOL>
<CR>                     ::= !!   carriage return character as defined in [ISO/IEC-6429]
<delimiter>              ::= <TAB>
<EOF>                    ::= !!   end-of-file character (CTL-Z)
<EOL>                    ::= <CR> <LF> | <LF> | <EOF>
<Fully_Qualified_Name>   ::= !!   AMO, ME: value of Meta-Meta-Attribute "Name";
                                  MR: concatenation of source ME-Name, MR-Name and target ME-
                                  Name, concatenation symbol: point character
<Digit>[902]             ::= 0..9
<GeneralPrintableChar>   ::= <UpperOrLowerCaseAlphabeticChar> | <Digit> | <GeneralSymbolChar>
<GeneralSymbolChar>      ::= !! any[903] character from [ISO/IEC-8859-1]
<Hyphen>                 ::= -
<Identifier>[904]        ::= {   <UpperOrLowerCaseAlphabeticChar> | <Digit> }
                                  [<UpperOrLowerCaseAlphabeticChar> | <Digit> | <UnderScore> |
                                  <Hyphen> ] ...
<Is_Used_In>             ::= 0 | 1            !! 0: Referenced, 1: Explicitly Used in SA
<LF>                     ::= !!   linefeed character as defined in [ISO/IEC-6429]
<LowerCaseAlphabeticChar> ::= a..z  !! English lower case letters
<MA>                     ::= MA
<ME>                     ::= ME
<MR>                     ::= MR
```

902 Vgl. Produktion in [CDIF94e], Seite 19, wo statt des Bereichsoperators sämtliche Ziffern von null bis neun angeführt sind.

903 Im Unterschied zur entsprechenden Definition in [CDIF94e], Seite 20, werden hier *alle* Zeichen aus dem [ISO/IEC-8859-1]-Zeichensatz erlaubt. (In [CDIF94e] werden die Zeichen \, **“**, | und # als Escapezeichen für die Klartext-Verkodierung benutzt und müssen daher dort selbst kodiert werden.)

904 Definition entnommen aus [CDIF94e], Seite 20 unten.

```
<SA>                        ::= SA
<SemiColon>                 ::= ;
<TAB>                       ::= !!   horizontal tab char as defined in [ISO/IEC-6429]
<UnderScore>                ::= _
<UpperCaseAlphabeticChar> ::= A..Z  !! English upper case letters
<UpperOrLowerCaseAlphabeticChar> ::= <LowerCaseAlphabeticChar> | <UpperCaseAlphabeticChar>
<Whitespace>                ::= !!   Space, horizontal tab, vertical tab, carriage return, line
                                     feed or form feed character as defined in [ISO/IEC-6429]
```

Abbildung 6-3: Die Definition der Verkodierung für das CTI-Austauschformat in EIA/CDIFs BNF-Version

6.2.2.4.2 Das Object Rexx-Programm „`CTI2SQL.cmd`“

In diesem Abschnitt werden die Routinendeklarationen des Object Rexx-Programmes „`CTI2SQLl.cmd`“ im Programmcode 6-14 dokumentiert. Die Aufgabe dieses Programmes ist es, aus CTI-Austauschdateien die Meta-Objektdefinitionen zu extrahieren und in die das EIA/CDIF-Meta-Meta-Modell repräsentierenden relationalen Tabellen einzufügen. Hierzu wird das Object Rexx-Modul „`M3SQL.CLS`“ mit Hilfe der „`::REQUIRES`“-Direktive eingebunden, sodaß in weiterer Folge auf die entsprechenden Klassen- und Routinendefinitionen zugegegriffen werden kann.

Die folgende Liste skizziert prägnant die Routinendeklarationen:

Routine	Beschreibung
`PP`	Liefert den als Argument übergebenen Wert in eckigen Klammern eingeschlossen zurück.
`initializeXferCDIF`	Öffentliche Routine zum Errichten einer „CDIF-Umgebung“, das eine Verzeichnisinstanz ist, die weitere Objekte beinhaltet, beispielsweise zum Speichern von Meta-Meta-Entitäten und Meta-Meta-Beziehungen.
`Parse_File`	Öffentliche Routine zum Auslesen und Zerlegen von CTI-Austauschdateien, wobei die ausgelesenen Meta-Objektdefinitionen mit Hilfe der Routine „`Create_MO_Object`“ zu Instanzen der entsprechenden Object Rexx-Klasse werden.
`Create_MO_Object`	Öffentliche Routine zur Instanziierung von Meta-Objekten aus den entsprechenden das EIA/CDIF-Meta-

Meta-Modell repräsentierenden Klassen, die in weiterer Folge im entsprechenden Sammelobjekt der CDIF-Umgebung gespeichert wird.

`Insert_MOs` Private Routine, die mit Hilfe der für die Klassen definierten Methoden die Meta-Objektdefinitionen als Datensätze in die entsprechenden relationalen Tabellen einfügt und anschließend entsprechend der in der CDIF-Umgebung gespeicherten Informationen auch die Meta-Meta-Beziehungen in den entsprechenden relationalen Tabellen vermerkt.

`cti2sql.init` Private Routine, die die Benutzerkennung und das Kennwort für die Etablierung der Datenbanksitzung festlegt, sowie ein kleines Verzeichnis der EIA/CDIF-Meta-Modelle definiert.

Der folgende Programmcode 6-14 dokumentiert das Modul „`CTI2SQLl.cmd`“.

```
/*
   author:        Rony G. Flatscher
   date:          97-09
   purpose:       read from RGF-CDIF-Interchange-file format and put meta-model data
                  into Database
   needs:         Mark Hessling's "RexxSQL" interface (available for OS/2, Unix, Windows95/NT)
   pre-requisite: needs an error free x2i-file (processed e.g. with x2i.cmd)
   usage:         "cti2sql x2i-file"
*/

.local ~ cti2sql.debug = .False /* set debug option     */

CALL cti2sql.init                      /* initialise data      */
CALL rxSQL.initialise                  /* initialise RexxSQL   */
CALL rxSQL.connect .cti2sql.conn, .cti2sql.sqlconnect          /* connect      */
CALL rxSQL.default .cti2sql.conn       /* define default connection           */

xfer_cdif = parse_file( ARG( 1 ) )     /* here, because NEXT_SURR() needs access to RDBMS */

SA_Obj      = xfer_cdif ~ SA_Object    /* get SA-object from XFER_CDIF-directory          */
SA_New_Name = SA_Obj ~ name            /* get name                            */
SA_Obj ~ ShortHand  = ""               /* default value for shorthand of SA    */
SA_Obj ~ CDIFNumber = ""
DO item OVER .cti2sql.shorthand        /* try to assign ShortName, CDIFNumber  */
   IF SA_New_Name = item[ 1 ] THEN
   DO
      SA_Obj ~ ShortHand  = item[ 2 ]  /* assign pre-defined short-hand        */
      SA_Obj ~ CDIFNumber = item[ 3 ]  /* official CDIF-Number for ordering    */
      LEAVE
   END
END

SAY "CTI ---> SQL:" pp( SA_Obj ~ CDIFNumber ) pp( SA_Obj ~ name ) pp( SA_Obj ~ ShortHand )
SAY
CALL delete_SA SA_Obj ~ name           /* if SA exists, delete it      */
all_classes = .array ~ of( .mo, .sa, .cmo, .ma, .amo, .me, .mr )
DO item OVER all_classes
   item ~ prep_insert                  /* prepare the INSERT- and the SELECT-statements */
END
        /* do the work          */
CALL Insert_MOs xfer_cdif, SA_Obj      /* insert MOs into RDBMS        */
CALL SQLCommit                         /* commit to database           */
        /* clean up            */
DO item OVER all_classes
   item ~ dispose           /* dispose of the working-areas for the prepared SQL-statements */
END
CALL rxSQL.finalise                    /* de-initialize RexxSQL        */
CALL rxSQL.disconnect .cti2sql.conn    /* disconnect                   */
SAY

:: REQUIRES m3sql.cls           /* get class-definitions for M3/SQL interface   */

:: ROUTINE pp; RETURN "[" || ARG( 1 ) || "]"
```

```
:: ROUTINE initializeXferCDIF   PUBLIC  /* create and initialize the xfer_cdif environment    */
IF .xfer_cdif <> ".XFER_CDIF" THEN      /* already defined              */
   RETURN .xfer_cdif

xfer_cdif = .directory ~ new            /* store results in Xfer_CDIF   */
     /* MetaMetaRelationships        */
               /* index is *first* CMO, value is *second* CMO e.g.
                        CMO.IsUsedIn.SA ...     IsUsedIn[ CMO ] = SA */
xfer_cdif ~    IsUsedIn               = .relation ~ new   /* CMO.iui.SA   */
xfer_cdif ~    IsLocalMetaAttributeOf  = .relation ~ new   /* MA.ilmao.AMO */
xfer_cdif ~    HasSubtype             = .relation ~ new   /* AMO.hs.AMO   */
xfer_cdif ~    HasSource              = .relation ~ new   /* MR.hs.ME     */
xfer_cdif ~    HasDestination         = .relation ~ new   /* MR.hd.ME     */
    /* lookup relations                    */
               /* ALL NAME2MO-entries! nameRel[ name ] = object */
xfer_cdif ~       nameRel   = .relation  ~ new
    /* index: CDIFMetaIdentifier 4 SA, ME, MR, MA       */
               /* ALL CDIF2MO-entries! cdifRel[ cdifMetaIdentifier ] = object  */
xfer_cdif ~       cdifRel   = .relation  ~ new
    /* sets                         */
xfer_cdif ~    SA_set       = .set ~ new        /* SubjectArea                  */
xfer_cdif ~    AMO_set      = .set ~ new        /* AttributableMetaObject (RootObject only)  */
xfer_cdif ~    ME_set       = .set ~ new        /* MetaEntity                   */
xfer_cdif ~    MR_set       = .set ~ new        /* MetaRelationship             */
xfer_cdif ~    MA_set       = .set ~ new        /* MetaAttribute                */
    /* additional objects needed for making life easier */
xfer_cdif ~    roots        = .set ~ new        /* set of roots (should have 1 entry only !) */
xfer_cdif ~    leaves       = .set ~ new        /* set of leaves                */
xfer_cdif ~    SA_object    = .nil      /* object to contain SA, this all is defined for     */
.local ~ xfer_cdif = xfer_cdif                  /* save directory with .local   */
RETURN xfer_cdif
```

```
/* parse file and build classes and fill in relations                          */
:: ROUTINE Parse_File           PUBLIC  /* read ASCII-file and setup MOs       */
   USE ARG inputFile
   IF STREAM( inputFile, "C", "QUERY EXISTS") = "" THEN
   DO
      CALL sayerror pp( inputFile ) "does not exist, aborting ..."
      RETURN .nil
   END
   inFile = STREAM( inputFile, "C", "OPEN READ" )       /* open input file     */
   inFile = inputFile
   .error ~ SAY( "parsing" pp( inFile ) "... " )
   /* get xfer_cdif environment    */
   xfer_cdif = initializeXferCDIF()     /* create and get a XFER_CDIF-environment-directory  */
   tabChar   = "09"x                    /* tab-character                       */
   selector = .nil                      /* abbreviation of MetaObject in hand  */
   tmpDir    = .directory ~ new         /* temporary object to address         */
   SAObject  = .nil                     /* keep SA-Object                      */
   tmpAMO    = .nil                     /* AMO for MA's to attach to           */
   DO WHILE CHARS( inFile ) > 0
      tmpLine = LINEIN( inFile )        /* read line                           */
      IF LEFT( tmpLine, 1 ) = ";" | tmpLine = "" THEN ITERATE   /* skip        */
      PARSE VAR tmpLine selector ( tabChar ) .
      IF selector <> "" THEN            /* a (new) selector in hand            */
      DO
         IF tmpDir ~ items > 0 THEN
         DO
             tmpAMO = create_MO_object( xfer_cdif, tmpDir, tmpAMO )
             DROP tmpDir
             tmpDir = .directory ~ new
         END
      END

      SELECT
         WHEN selector = "SA" THEN                      /* SubjectArea         */
                 tmpDir ~ Selector = selector           /* save selector       */

         WHEN POS( selector, "AMO ME MR" ) > 0 THEN     /* AMO, ME, MR in hand */
              DO
                 tmpDir ~ Selector = selector   /* save selector        */
                 PARSE VAR tmpLine . ( tabChar ) bDefinedInSA ( tabChar ) name
                 tmpDir ~ bDefinedInSA   = bDefinedInSA /* defined in SA ?     */

                 IF selector = "MR" THEN
                    tmpDir ~ LongName = name            /* concatenated name   */
              END

         WHEN selector = "MA" THEN                      /* MetaAttribute       */
                 tmpDir ~ Selector = selector           /* save selector       */

         OTHERWISE                                      /* a (M)MA-definition in hand  */
              DO
                 PARSE VAR tmpLine ( tabChar ) attribute ( tabChar ) value
                 IF value <> "" THEN
                    tmpDir ~ setentry( attribute, value )
              END
      END
   END
```

```
IF tmpDir ~ items > 0 THEN                  /* directory contains an entry? */
    tmpAMO = create_MO_object( xfer_cdif, tmpDir, tmpAMO )
CALL STREAM inFile, "C", "CLOSE"            /* close input file             */
.error ~ SAY( "parsing" pp( inFile ) "... ended." )
.error ~ SAY()
RETURN xfer_cdif                            /* return directory             */
```

```
/* create appropriate object and set attributes according to tmpDir             */
/* tmpAMO = last known ME, MR                                                   */
:: ROUTINE Create_MO_Object     PUBLIC          /* create a MO and check it     */
   USE ARG xfer_cdif, tmpDir, tmpAMO
   selector  = tmpDir ~ remove( "SELECTOR" )    /* get selector                 */
   IF      selector = "SA"  THEN    clsObject = .SA
   ELSE IF selector = "ME"  THEN    clsObject = .ME
   ELSE IF selector = "MR"  THEN    clsObject = .MR
                                    /* no AMO to attach MA to */
   ELSE IF selector = "MA"  THEN DO
                                  clsObject = .MA
                                  IF tmpAMO = .nil THEN RETURN .nil
                                      /* if MA has to be used as its AMO      */
                                  tmpDir ~ bDefinedInSA = ( tmpAmo ~ bDefinedInSA )
                                END
   ELSE IF selector = "AMO" THEN    clsObject = .AMO
   ELSE /* unknown, i.e. illegal selector received ! */
   DO
      SIGNAL ON SYNTAX
      RAISE SYNTAX 40.1 ARRAY ("CREATE_MO_Object; unknown selector" PP( selector ) )
   END
   tmpDir ~ surr = next_surr()                  /* create and assign new surrogate */
   tmpObject = clsObject ~ new( tmpDir )        /* now create instance */
   IF selector <> "MA" THEN tmpAMO = tmpObject  /* MO to attach MAs to  */

   IF xfer_cdif ~ SA_Object = .nil THEN
   DO
      IF selector <> "SA" THEN                  /* oops, no SA defined as of yet ! */
      DO
         SIGNAL ON SYNTAX                       /* activate SYNTAX-trapping     */
                                                /* raise a SYNTAX error         */
         RAISE SYNTAX 40.1 ARRAY ( "Missing SA-Object (must be first!); received instead" ,
                                   PP( selector ) )
      END
   END

   SELECT
      WHEN POS( selector, "ME MR AMO" ) > 0 THEN
           DO
              tmpAMO    = tmpObject             /* for possible MA's to attach to       */
              SELECT
                 WHEN SELECTOR = "ME" THEN xfer_cdif ~ ME_set  ~ put( tmpObject )
                 WHEN SELECTOR = "MR" THEN xfer_cdif ~ MR_set  ~ put( tmpObject )
                 OTHERWISE                 xfer_cdif ~ AMO_set ~ put( tmpObject )
              END
           END

      WHEN selector = "MA" THEN                 /* MetaAttribute                        */
           DO
              xfer_cdif ~   MA_set ~ put( tmpObject )
              xfer_cdif ~ IsLocalMetaAttributeOf[ tmpObject ] = tmpAMO
              name = tmpObject ~ name                   /* get attribute name           */
              tmpAMO ~ LocalMA[ Name ] = tmpObject      /* save with AMO                */
              tmpObject ~ BelongsTo = tmpAmo            /* point to AMO, MA belongs to  */
                /* Check and set IsOptional, default to .True */
              IF ( TRANSLATE( tmpObject ~ IsOptional ) <> "FALSE" ) THEN
                 tmpObject ~ IsOptional = "True"
              ELSE
                 tmpObject ~ IsOptional = "False"
           END
```

```
      WHEN selector = "SA" THEN                /* SubjectArea                          */
           DO
              tmpAMO   = tmpObject      /* not an AMO, but needed for Create_SAObject() */
              xfer_cdif ~   SA_set ~ put( tmpObject )
              IF xfer_cdif ~ SA_object = .nil THEN      /* save SA-object              */
                 xfer_cdif ~ SA_object = tmpObject
           END

      OTHERWISE NOP
   END

   IF selector <> "SA" THEN                    /* determine locality of MO             */
   DO
      IF tmpObject ~ bDefinedInSA THEN         /* explicitly used                      */
         xfer_cdif ~ IsUsedIn[ tmpObject ] = xfer_cdif ~ SA_object
   END
        /* add to directory CDIFMetaIdentifier --> object       */
   xfer_cdif ~ cdifRel ~ put( tmpObject, tmpObject ~ CDIFMetaIdentifier )
        /* add to directory Name/LongName      --> object       */
   IF POS( selector, "MA SA" ) = 0 THEN
      xfer_cdif ~ nameRel ~ put( tmpObject, tmpObject ~ LongName )
   RETURN tmpAMO

SYNTAX : RAISE PROPAGATE                       /* raise error in caller                */
```

```
/* insert MOs into tables, check whether they exist or not     */
:: ROUTINE Insert_MOs
   USE ARG xfer_cdif, SA_Obj
   SA_SURR = SA_Obj ~ SURR             /* get surrogate value for SA           */
                                       /* build array of MO-object sets        */
   workArr = .array ~ of( xfer_cdif ~ SA_set,  .sa ,,
                          xfer_cdif ~ AMO_set, .amo ,,
                          xfer_cdif ~ ME_set,  .me ,,
                          xfer_cdif ~ MR_set,  .mr ,,
                          xfer_cdif ~ MA_set,  .ma )
   DO i = 1 TO workArr ~ items BY 2
      clsObj = workArr[ i + 1 ]        /* get appropriate class object         */
      SAY "Inserting item(s) of type" pp( clsObj ~ ID )
               /* sort MOs     */
      sortArray = SortCollection( workArr[ i ], "SORTVALUE" )
      totItems = WorkArr[ i ] ~ items
      totWidth = LENGTH( totItems )
      DO j = 1 TO  totItems
         SAY "---" RIGHT( j, totWidth) "of" totItems,
                   pp( sortArray[ j, 1 ] )      /* show name                   */
         clsObj ~ insert( sortArray[ j, 2 ] )   /* insert object               */
         IF clsObj <> .sa THEN                  /* Insert into relation "IsUsedIn"      */
         DO
            IF sortArray[ j, 2 ] ~ bDefinedInSA THEN    /* only, if explicitly defined  */
               CALL insert_rel "IsUsedIn_", sortArray[ j, 2 ] ~ SURR, SA_SURR
         END
      END
      SAY
   END
```

```
.output ~ SAY( "Setting up relations ... " )
     /* set-up IsLocalMetaAttributeOf, HasSubtype, HasSource, HasDestination */
DO i = 1 TO workArr ~ items BY 2
   clsObj = workArr[ i + 1 ]         /* get appropriate class object                */
   IF clsObj = .SA THEN ITERATE      /* don't handle SA                             */

   DO tmpObj OVER workArr[ i ]
      SELECT
         WHEN clsObj = .ma then      /* attach MA to AMO via "IsLocalMetaAttributeOf" */
              DO
                 CALL insert_rel "IsLocalMetaAttributeOf_", tmpObj ~ SURR, ,
                                 tmpObj ~ belongsTo ~ SURR
              END

         OTHERWISE                   /* AMO, ME, MR  */
              DO
                 /* populate HasSubtype  */
                 tmpString = tmpObj ~ SubtypeOf
                 IF tmpString <> .nil THEN
                 DO WHILE tmpString <> ""
                    PARSE VAR tmpString supertype tmpString
                    CALL insert_rel "HasSubtype_",  get_SURR_by_longName( supertype ),,
                                    tmpObj ~ SURR
                 END
                 /* populate HasSource and HasDestination */
                 IF clsObj = .mr THEN
                 DO
                    PARSE VALUE tmpObj ~ LongName WITH source "." . "." destination
                    CALL insert_rel "HasSource_",      tmpObj ~ SURR,,
                                    get_SURR_by_longName( source )
                    CALL insert_rel "HasDestination_", tmpObj ~ SURR,,
                                    get_SURR_by_longName( destination )
                 END
              END
      END
   END
END
.output ~ SAY( "Setting up relations ... ended." )
SAY
RETURN
```

```
/* initialise CTI2SQL-variables, use data from op_sys environment, if available */
:: ROUTINE cti2sql.init
   .local ~ cti2sql.conn = "CDIF_CONNECT"       /* define connection name       */
        /* logon-data from op_sys-environment, else defaults to: test/test1     */
   sqlconnect.1 = VALUE('REXXSQL_USERNAME',, .rxSQL.envname)
   IF sqlconnect.1 = "" THEN sqlconnect.1 = "test"             /* default      */
   sqlconnect.2 = VALUE('REXXSQL_PASSWORD',, .rxSQL.envname)
   IF sqlconnect.2 = "" THEN sqlconnect.2 = "test1"            /* default      */
   sqlconnect.3 = VALUE('REXXSQL_DATABASE',, .rxSQL.envname)
   sqlconnect.4 = VALUE('REXXSQL_SERVER'  ,, .rxSQL.envname)
   .local ~ cti2sql.sqlconnect = sqlconnect.                   /* save in .local */
        /* EIA/CDIF abbreviations, EIA-publishing number        */
   .local ~ cti2sql.shorthand = .array ~ of(,
      .array ~ of( "Foundation"                          , "FND"  , "EIA/IS-111" ) ,,
      .array ~ of( "Common"                              , "CMMN" , "EIA/IS-112" ) ,,
      .array ~ of( "DataModeling"                        , "DMOD" , "EIA/IS-114" ) ,,
      .array ~ of( "DataFlowModel"                       , "DFM"  , "EIA/IS-115" ) ,,
      .array ~ of( "PresentationLocationAndConnectivity" , "PLAC" , "EIA/IS-118" ) ,,
      .array ~ of( "StateEventModeling"                  , "STEV" , "EIA/IS-116" ) ,,
      .array ~ of( "DataDefinition"                      , "DDEF" , "EIA/IS-113" ) ,,
      .array ~ of( "BusinessProcessModeling"             , "BPM"  , "EIA/IS-???" ) ,,
      .array ~ of( "IntegratedMetaModel_RGF"             , "IMM"  , "EIA/IS-???" ) ,,
      .array ~ of( "PlanningAndScheduling"               , "PMPS" , "EIA/IS-121" )    )
   RETURN
```

Programmcode 6-14: Das Object-Rexx-Programm „CTI2SQL.cmd“

6.3 Verzeichnisse

In diesem Abschnitt werden verschiedene Verzeichnisse dargestellt, die entweder das Verständnis für die Inhalte dieser Arbeit erleichtern sollen oder prägnante Auflistungen der erarbeiteten Inhalte darstellen.

6.3.1 Abkürzungsverzeichnis

Im Rahmen dieser Arbeit wurden zahlreiche Abkürzungen benutzt, um das Erarbeiten des Inhalts zu erleichtern. Die folgende Tabelle 6-4 stellt in der ersten Spalte die Abkürzung und in der zweiten Spalte die ausgeschriebene Form dar.

ADTF	Analysis and Design Task Force (OMG)
AKA	Also Known As
AMO	AttributableMetaObject, deutsch: AttribuierbaresMetaObjekt
ANSI	American National Standards Institute
ARIS	Architektur integrierter Informationssysteme
ASM	Allgemeiner Strukturierungsmechanismus
BNF	Backus-Naur-Form
BPM	CDIF Integrated Meta-model: Business Process Modeling Subject Area
CACSD	CDIF Integrated Meta-model: Computer Aided Control Systems Design Modeling Subject Area
CASE	Computer Aided Software Engineering, manchmal auch: Computer Aided Systems Engineering
CCIR	Consultative Committee on International Radio
CD	Committee Draft
CDIF	CASE Data Interchange Format, ab 1998 auch: Common Data Interchange Facility, Concept Data Interchange Facility/Format
CDV	Committee Draft for Vote (IEC)
CMMN	CDIF Integrated Meta-model: Common Subject Area
CMO	CollectableMetaObject, deutsch: SammelbaresMetaObjekt
CORBA	Common Object Request Broker
DDEF	CDIF Integrated Meta-model: Data Definition Subject Area
DFM	CDIF Integrated Meta-model: Data Flow Model Subject Area
DIN	Deutsches Institut für Normung
DIS	Draft International Standard (ISO)
DMOD	CDIF Integrated Meta-model: Data Modeling Subject Area
DTD	Document Type Definition (SGML, XML)

ECMA	European Computer Manufacturers Association, seit 1994: ECMA – European Association for Standardizing Information and Communication Systems
EERM	Extended (Erweitertes) ERM
EIA	Electronic Industries Association, seit 1998: Electronic Industries Alliance
EIG	Electronic Information Group (EIA)
ER	Entity Relationship
ERM	Entity Relationship Modell beziehungsweise Modellierung
EXPR	CDIF Integrated Meta-model: Expression Modeling Subject Area
FCD	Final Committee Draft
FCO	First Class Object
FDIS	Final Draft International Standard
FND	CDIF Integrated Meta-model: Foundation Subject Area
FZI	Forschungszentrum Informatik (Universität Karlsruhe)
GOPPR	Graph-Object-Property-Role-Relationship
GRAL	Graph Specification Language
GSM	General Structuring Mechanism
ICAM	Integrated Computer Aided Manufacturing
IDEF	Integration Definition
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
IGES	Initial Graphics Exchange Specification
IKS	Informations- und Kommunikationssysteme
IMM	Integriertes Meta-Modell
IRDS	Information Resource Dictionary System
IS	Informationssystem
ISO	International Organization for Standardization
ICT	Information and Communication Technology (ECMA)
JTC	Joint Technical Committee
JTPC	Joint Technical Programming Committee
KOGGE	Koblenzer Generator für Graphische Entwurfsumgebungen
M0	Benutzerdaten (aus null "M" bestehend); Instanzdaten von M1-Typen, selbst nicht mehr instanziierbar
M1	Modell (aus einem "M" bestehend); Instanzen von M2-Typen
M2	Meta-Modell (aus zwei "M" bestehend); Instanzen von M3-Typen
M3	Meta-Meta-Modell (aus drei "M" bestehend)
MA	MetaAttribute, deutsch: MetaAttribut
ME	MetaEntity, deutsch: MetaEntität
MetaPHOR	Metamodeling: Principles, Hypertext, Objects and Repositories
MO	MetaObject, deutsch: MetaObjekt
MOF	Meta-Object Facility

MR	MetaRelationship, deutsch: MetaBeziehung
NIAM	Nijssen's Information Analysis Method
NP	New work item proposal
OMG	Object Management Group
ON	Österreichisches Normungsinstitut
ÖNORM	Österreichische Norm
OOAD	CDIF Integrated Meta-model: Object-Oriented Analysis and Design Modeling Subject Area
PCTE	Portable Common Tool Environment (ECMA)
PIO	PresentationInformationObject
PLAC	CDIF Integrated Meta-model: Presentation Location and Connectivity Subject Area
PMPS	CDIF Integrated Meta-model: Project Management Planning and Scheduling Modeling Subject Area
PORD	CDIF Integrated Meta-model: Physical (Object) Relational Database Modeling Subject Area
PROLOG	Programming in Logic
PWI	Preliminary work item
RE	RootEntity
RE.IR2.RE	RootEntity.IsRelatedTo.RootEntity
RE.IRT.RE	RootEntity.IsRelatedTo.RootEntity
RFI	Request for Information
RFP	Request for Proposal
SA	SubjectArea, deutsch: GegenstandsBereich
SC	Subcommittee
SDAI	Standard data access interface specification (STEP)
SDS	Schema Definition Sets (ECMA-PCTE)
SEDDI	Software Engineering Data Description and Interchange (ISO/CDIF)
SERM	Strukturiertes ERM
SGML	Standard Generalized Markup Language
SIO	SemanticInformationObject
SMIF	Stream-based Model Interchange Format (OMG)
SQL	Structured Query Language
STEP	Standard for the Exchange of Product Data
STEV	CDIF Integrated Meta-model: State Event Subject Area
TC	Technical Committee
UDM	Unternehmensdatenmodell
UML	Unified Modeling Language
URL	Uniform Resource Locator
UTC	französische Abkürzung für den englischen Begriff "Universal Coordinated Time" der CCIR
WD	Working draft

WG	Working Group
XML	Extensible Markup Language

Tabelle 6-4: Verzeichnis der Abkürzungen

6.3.2 Kleines Englisch/Deutsch- und Deutsch/Englisch-Lexikon

Im Rahmen dieser Arbeit wurden englische Bezeichnungen ins Deutsche übersetzt. Damit sollen zum einen normierend die deutschen Begriffe und Bedeutungen für EIA/CDIF festgelegt werden[905], zum anderen kann als Folge in dieser Arbeit abwechselnd der englische oder der deutsche Begriff eingesetzt werden. Beispielsweise kann für die englische Bezeichnung „AttributableMetaObject“ synonym der deutsche Begriff „AttribuierbaresMetaObjekt“ benutzt werden.

Die deutschen – nach den Regeln von EIA/CDIF – zusammengesetzten Bezeichner werden in dieser Arbeit entsprechend den Regeln der deutschen Grammatik konjugiert, beispielsweise in dem Satzausschnitt: „... in einem Attribuierbare***n***MetaObjekt werden ...“.

6.3.2.1 Übersetzungen: Englisch-Deutsch

Die folgende Tabelle 6-5 beinhaltet in der ersten Spalte die englischen Begriffe und in der zweiten ihre deutschen Entsprechungen.

AbsolutePoint	AbsoluterPunkt
AbstractionLevel	AbstraktionsEbene
AccessPath	ZugriffsPfad
ActsAs	BenimmtSichAls
AggregationalMetaRelationship	AggregierungsMetaBeziehung
AlternateName	AliasName
Annotation	Beschreibung, Anmerkung
AnnotationArgument	BeschreibungsArgument
AppearsOn	ErscheintAuf
AttributableMetaObject	AttribuierbaresMetaObjekt
Attribute	Attribut
BelongsTo	GehörtZu
CASCADES	Kaskadiert (Fortpflanzen)
CandidateKey	PrimärSchlüsselKandidat
Cluster	Haufen, Gruppe

905 Durch diese Festlegungen und Dokumentation im Zusammenhang mit dieser Arbeit sollen die Übersetzungen selbst einer Diskussion zugeführt werden.

CollectableMetaObject	SammelbaresMetaObjekt
Collects	Sammelt
ComponentObject	KomponentenObjekt, BestandteilObjekt
ComponentObjectReference	KomponentenObjektReferenz
ConsistsOf	BestehtAus
Constrains	Beschränkt
ConstraintPort	EinschränkungsÖffnung
Consumes	Konsumiert
Contains	Beinhaltet
ContextIdentifier	KontextIdentifikator
ControlPort	KontrollÖffnung
CreatedBy	AngelegtVon
DFMProcess	DFMProzeß
DFMProcessDefinition	DFMProzeßDefinition
DataFlow	Datenfluß
DataFlowModel	DatenFlußModell
DataModel	DatenModell
DataModelObject	DatenModellObjekt
DataModelSubset	DatenModellTeilMenge
DataObject	DatenObjekt
DefinesPath	DefiniertPfad
DefinitionObject	DefinitionsObjekt
DefinitionObjectReference	DefinitionsObjektReferenz
DeleteEffect	LöschenEffekt
Derivation	Ableitung
Diagram	Diagramm
Edge	Kante
EdgeElement	KantenElement
Entity	Entität
EquivalenceSet	ÄquivalenzMenge
Exception	Ausnahmebedingung
Excludes	NimmtAus
Extensibility	Erweiterung
ExternalAgent	Externer Agent
ExternalAgentDefinition	ExterneAgentenDefinition
Flow	Fluß
FlowDefinition	FlußDefinition
FlowInputPort	FlußEingabeÖffnung
FlowOutputPort	FlußAusgabeÖffnung
FlowPort	FlußÖffnung

FlowProducerConsumer	FlußProduzentKonsument
ForBuildingStructureOf	ZurBildungDerStrukturVon
ForeignKey	FremdSchlüssel
from	von
General Structuring Mechanism	Allgemeiner Strukturierungsmechanismus
GraphicalElement	GraphischesElement
Has	Hat
HasCenter	HatMittelpunkt
HasDestination	HatAlsZiel
HasEnd	HatEnde
HasMember	HatElement
HasRoot	HatWurzel
HasSource	HatAlsQuelle
HasSubtype	BesitztUntergeordnetes
Incorporates	NimmtAuf
InheritableDataModelObject	VererbbaresDatenModellObjekt
InsertEffect	EinfügeEffekt
Instantiates	Instanziiert
Integrated Meta-Model	Integriertes Meta-Modell
Interface	Schnittstelle
IsAccessedUsing	WirdZugegriffenMitHilfeVon
IsAttachedTo	IstBefestigtAn
IsCategorizedIn	IstKategorisiertIn
IsConstraintOn	IstEinschränkungFür
IsConstructedWith	WirdAufgebautMit
IsDependentUpon	IstAbhängigVon
IsDiscriminatorOf	IstDiskriminatorFür
IsFullProjectionOf	IstVolleAbleitungVon
IsIdentifiedBy	WirdIdentifiziertMit
IsInheritedFrom	IstVererbtVon
IsLocalMetaAttributeOf	IstLokalesMetaAttributVon
IsLocatedOn	BefindetSichAuf
IsMemberOf	IstMitgliedVon
IsPartOfIdentityOf	IstTeilDerIdentitätVon
IsProjectionOf	IstAbleitungVon
IsRelatedTo	IstBezogenAuf
IsRelativeTo	IstRelativZu
IsSubsetOf	IstTeilmengeVon
IsSupertypeFor	IstSupertypFür
IsSupportedBy	WirdUnterstütztVon

IsUsedIn	WirdBenutztIm
Key	Schlüssel
Language	Sprache
LastUpdatedBy	ZuletztGeändertVon
M2LevelObject	M2SchichtObjekt
mandatory	zwingend
MetaAttribute	MetaAttribut
MetaEntity	MetaEntität
MetaEntityReference	MetaEntitätsReferenz
MetaObject	MetaObjekt
MetaObjectReference	MetaObjektReferenz
MetaRelationship	MetaBeziehung
MetaRelationshipReference	MetaBeziehungsReferenz
NULL	Nichts (kein Wert vorhanden)
Namespace	Namensraum, Gültigkeitsbereich
Node	Knoten
Point	Punkt
Port	Pforte, Einlaß-, Auslaßöffnung
PositionedElement	PositioniertesElement
PresentationInformationObject	DarstellungsInformationsObjekt
PresentationInformationObject	PräsentationsInformationsObjekt
ProcessObject	ProzeßObjekt
Produces	Erzeugt, Produziert
ProducesOrConsumes	KonsumiertOderErzeugt
ProjectedAttribute	AbgeleitetesAttribut
ProjectionComponent	AbleitungsKomponente
RESTRICTS	beschränken
ReferencedElement	ReferenziertesElement, ReferenzierterBestandteil
References	Referenziert
Refines	Verfeinert
Relationship	Beziehung
RelativePoint	RelativerPunkt
Represents	Repräsentiert
Role	Rolle
RoleConstraint	RollenBeschränkung
RolePlayer	RollenSpieler
RootEntity	WurzelEntität
SETDEFAULT	auf Standardwert setzen
SETNULL	Wert entfernen
Selects	WähltAus

SemanticInformationObject	SemantischesInformationsObjekt
SemanticObjectReference	SemantischeObjektReferenz
SequenceNumber	SequenzNummer
SetOfExclusiveMetaRelationships	MengeEinanderAusschließenderMetaBeziehungen
SetOfValidComponents	MengeGültigerKomponenten
Specifies	Spezifiziert
Store	Speicher
StoreDefinition	SpeicherDefinition
SubjectArea	GegenstandsBereich
SubtypeSet	SubtypMenge
SubtypeSetMembershipCriterion	SubtypMengenKriterium
SupportPort	UnterstützungsÖffnung
TextualConstraint	TextuelleEinschränkung
ToolUser	WerkzeugBenutzer
Traversal	Navigation, Abschreiten
UpdateEffect	ÄnderungsEffekt
Uses	Benutzt
W3	World-Wide-Web
Working Meta-Model	Arbeits-Meta-Modell
WWW	World-Wide-Web

Tabelle 6-5: Begriffsübersetzungen aus dem Englischen ins Deutsche

6.3.2.2 Übersetzungen: Deutsch-Englisch

Die folgende Tabelle 6-6 beinhaltet in der ersten Spalte die deutschen Begriffe und in der zweiten ihre englischen Entsprechungen.

AbgeleitetesAttribut	ProjectedAttribute
Ableitung	Derivation
AbleitungsKomponente	ProjectionComponent
Abschreiten	Traversal
AbsoluterPunkt	AbsolutePoint
AbstraktionsEbene	AbstractionLevel
AggregierungsMetaBeziehung	AggregationalMetaRelationship
AliasName	AlternateName
Allgemeiner Strukturierungsmechanismus	General Structuring Mechanism
AngelegtVon	CreatedBy
ÄnderungsEffekt	UpdateEffect
Anmerkung	Annotation
ÄquivalenzMenge	EquivalenceSet
Arbeits-Meta-Modell	Working Meta-Model
AttribuierbaresMetaObjekt	AttributableMetaObject
Attribut	Attribute
Auslaßöffnung	Port
Ausnahmebedingung	Exception
BefindetSichAuf	IsLocatedOn
Beinhaltet	Contains
BenimmtSichAls	ActsAs
Benutzt	Uses
Beschreibung	Annotation
BeschreibungsArgument	AnnotationArgument
Beschränkt	Constrains
BesitztUntergeordnetes	HasSubtype
BestandteilObjekt	ComponentObject
BestehtAus	ConsistsOf
Beziehung	Relationship
DFMProzeß	DFMProcess
DFMProzeßDefinition	DFMProcessDefinition
DarstellungsInformationsObjekt	PresentationInformationObject
DatenFlußModell	DataFlowModel
DatenModell	DataModel
DatenModellObjekt	DataModelObject

DatenModellTeilMenge	DataModelSubset
DatenObjekt	DataObject
Datenfluß	DataFlow
DefiniertPfad	DefinesPath
DefinitionsObjekt	DefinitionObject
DefinitionsObjektReferenz	DefinitionObjectReference
Diagramm	Diagram
EinfügeEffekt	InsertEffect
Einlaßöffnung	Port
EinschränkungsÖffnung	ConstraintPort
Entität	Entity
ErscheintAuf	AppearsOn
Erzeugt	Produces
Externe AgentenDefinition	ExternalAgentDefinition
Externer Agent	ExternalAgent
Fluß	Flow
FlußAusgabeÖffnung	FlowOutputPort
FlußDefinition	FlowDefinition
FlußEingabeÖffnung	FlowInputPort
FlußProduzentKonsument	FlowProducerConsumer
FlußÖffnung	FlowPort
FremdSchlüssel	ForeignKey
GegenstandsBereich	SubjectArea
GehörtZu	BelongsTo
GraphischesElement	GraphicalElement
Gruppe	Cluster
Gültigkeitsbereich	Namespace
Hat	Has
HatAlsQuelle	HasSource
HatAlsZiel	HasDestination
HatElement	HasMember
HatEnde	HasEnd
HatMittelpunkt	HasCenter
HatWurzel	HasRoot
Haufen	Cluster
Instanziiert	Instantiates
Integriertes Meta-Modell	Integrated Meta-Model
IstAbhängigVon	IsDependentUpon
IstAbleitungVon	IsProjectionOf
IstBefestigtAn	IsAttachedTo

IstBezogenAuf	IsRelatedTo
IstDiskriminatorFür	IsDiscriminatorOf
IstEinschränkungFür	IsConstraintOn
IstKategorisiertIn	IsCategorizedIn
IstLokalesMetaAttributVon	IsLocalMetaAttributeOf
IstMitgliedVon	IsMemberOf
IstRelativZu	IsRelativeTo
IstSupertypFür	IsSupertypeFor
IstTeilDerIdentitätVon	IsPartOfIdentityOf
IstTeilmengeVon	IsSubsetOf
IstVererbtVon	IsInheritedFrom
IstVolleAbleitungVon	IsFullProjectionOf
Kante	Edge
KantenElement	EdgeElement
Knoten	Node
KomponentenObjekt	ComponentObject
KomponentenObjektReferenz	ComponentObjectReference
Konsumiert	Consumes
KonsumiertOderErzeugt	ProducesOrConsumes
KontextIdentifikator	ContextIdentifier
KontrollÖffnung	ControlPort
LöschenEffekt	DeleteEffect
M2SchichtObjekt	M2LevelObject
MengeEinanderAusschließenderMetaBeziehungen	SetOfExclusiveMetaRelationships
MengeGültigerKomponenten	SetOfValidComponents
MetaAttribut	MetaAttribute
MetaBeziehung	MetaRelationship
MetaBeziehungsReferenz	MetaRelationshipReference
MetaEntität	MetaEntity
MetaEntitätsReferenz	MetaEntityReference
MetaObjekt	MetaObject
MetaObjektReferenz	MetaObjectReference
Namensraum	Namespace
Navigation	Navigation, Traversal
NimmtAuf	Incorporates
NimmtAus	Excludes
Pforte	Port
PositioniertesElement	PositionedElement
PrimärSchlüsselKandidat	CandidateKey
Produziert	Produces

ProzeßObjekt	ProcessObject
PräsentationsInformationsObjekt	PresentationInformationObject
Punkt	Point
Referenziert	References
ReferenzierterBestandteil	ReferencedElement
ReferenziertesElement	ReferencedElement
RelativerPunkt	RelativePoint
Repräsentiert	Represents
Rolle	Role
RollenBeschränkung	RoleConstraint
RollenSpieler	RolePlayer
SammelbaresMetaObjekt	CollectableMetaObject
Sammelt	Collects
Schlüssel	Key
Schnittstelle	Inteface
SemantischeObjektReferenz	SemanticObjectReference
SemantischesInformationsObjekt	SemanticInformationObject
SequenzNummer	SequenceNumber
Speicher	Store
SpeicherDefinition	StoreDefinition
Spezifiziert	Specifies
Sprache	Language
SubtypMenge	SubtypeSet
SubtypMengenKriterium	SubtypeSetMembershipCriterion
TextuelleEinschränkung	TextualConstraint
UnterstützungsÖffnung	SupportPort
VererbbaresDatenModellObjekt	InheritableDataModelObject
Verfeinert	Refines
von	from
WerkzeugBenutzer	ToolUser
WirdAufgebautMit	IsConstructedWith
WirdBenutztIn	IsUsedIn
WirdIdentifiziertMit	IsIdentifiedBy
WirdUnterstütztVon	IsSupportedBy
WirdZugegriffenMitHilfeVon	IsAccessedUsing
WurzelEntität	RootEntity
WähltAus	Selects
ZugriffsPfad	AccessPath
ZuletztGeändertVon	LastUpdatedBy

ZurBildungDerStrukturVon	ForBuildingStructureOf
zwingend	mandatory

Tabelle 6-6: Begriffsübersetzungen aus dem Deutschen ins Englische

6.3.3 Abbildungsverzeichnis

6.3.4 SQL-Tabellendefinitionsverzeichnis

6.3.5 SQL-Viewdefinitionsverzeichnis

6.3.6 Programmcode-Verzeichnis

6.3.7 Verzeichnis der Tabellen für das Integrierte EIA/CDIF-Meta-Modell

6.3.8 Tabellenverzeichnis

6.3.9 Literaturverzeichnis

[AmbRau95] Amberg M., Raue H.: „Eine Beschreibungsform für Informationssystem-Architekturen“, in: [GI52_95a].

[Ande86] Anderson R.: „Management, Information Systems and Computers“, Macmillan Education Ltd., London 1986.

[AnAlSi89] Angeli A., Alandt K., Siromach-Kostov L.: „Relationale Datenbanksysteme für Softwareentwickler“, Addison-Wesley, Bonn 1989.

[ANSI96] „Information Technology – Programming Language REXX“, ANSI X3.274-1996.

[AO96] Arbeitskreis Organisation: „Kontaktstudium: Restrukturierung – Organisation im Umbruch“, in: ZfbF, 48. Jg. (1996) 6, S. 621 – 665.

[BaCeNa92] Batini C., Ceri S., Navathe S.: „Conceptual Database Design“, The Benjamin/Cummings Publishing, Redwood City 1992.

[Balz82] Balzert H.: „Die Entwicklung von Software-Systemen“, Bibliographisches Institut, Mannheim 1982.

[Balz91] Balzert H. (Hrsg.): „CASE – Systeme und Werkzeuge“, 3. Auflage, BI-Wissenschaftsverlag, Zürich 1991.

[Bark90a] Barker R.: „CASE*Method: Tasks and Deliverables“, Addison Wesley, Reading 1990.

[Bark90b] Barker R.: „CASE*Method: Entity Relationship Modelling“, Addison Wesley, Reading 1990.

[BarLon92] Barker R., Longman C.: „CASE*Method: Function and Process Modelling“, Addison Wesley, Reading 1990.

[Ben95] Ben-Natan R.: „CORBA – a guide to common object request broker architecture“, MacGraw-Hill, New York 1995.

[BiMuRu97] Biethan J., Mucksch H., Ruf W.: „Ganzheitliches Informationsmanagement, Band 2: Entwicklungsmanagement“, 2. Auflage, Oldenbourg Verlag, München 1997.

[BinRex95] Binstock A., Rex J.: „Practical Algorithms for Programmers“, Addison-Wesley, Reading 1995.

[BlDiTuW90] Bleimann U., Dippel D., Turetschek G., Wente K.: „Betriebsinformatik – Informationsverarbeitungssysteme in Unternehmen und Verwaltungen“, Carl Hanser Verlag, München/Wien 90.

[Blei81] Bleicher K.: „Organisation – Formen und Modelle“, Verlag Dr. Th. Gabler, Wiesbaden 1981.

[BloBlo87] Blokdijk A., Blokdijk P.: „Planning and Design of Information Systems“, Academic Press, London 1987.

[Blum91] Blum E.: „Betriebsorganisation: Methoden und Techniken“, 3. Auflage, Gabler, Wiesbaden 1991.

[Booc94a] Booch G.: „Object-Oriented Analysis and Design with Applications“, Second Edition, Benjamin/Cummings Publishing Comp., Redwood City: California 1994.

[Booc94b] Booch G.: „Objektorientierte Analyse und Design“, Addison-Wesley, Bonn et.al. 1994.

[Booc96] Booch G.: „Object Solutions – Managing the Object-Oriented Project“, Addison-Wesley, Menlo Park 1996.

[BraBra88] Brassard G., Bratley P.: „Algorithms – Theory and Practice“, Prentice-Hall, Englewood Cliffs: New Jersey 1988.

[Bran97] Brandtweiner R.: „Wissenschaftstheorie und wissenschaftliche Methodik für Wirtschaftswissenschafter“, Skriptum, Servicebetriebe der ÖH-WU, Wien 1997.

[BrLyWe96] Brinkkemper S., Lyytinen K., Welke R.J. (Hrsg.): „Method Engineering – Principles of method construction and tool support“, Chapman & Hall, London et.al. 1996.

[Burk97] Burkhardt R.: „UML – Unified Modeling Language – Objektorientierte Modellierung für die Praxis“, Addison-Wesley, Bonn et.al. 1997.

[Busc83] Busch U.: „Konzeption betrieblicher Informations- und Kommunikationssysteme (IKS)“, Erich Schmidt Verlag, Berlin 1983.

[CarEbe94] Carstensen M., Ebert J.: „Ansatz und Architektur von KOGGE“, Interner Projektbericht 2/94, Universität Koblenz-Landau, Institut für Softwaretechnik, Koblenz, 1994. URL (Stand: 1998-05-30): „`http://www.uni-oblenz.de/~ist/retrieve/ansatz_architektur.ps.gz`“.

[CDIF94a] „CDIF – CASE Data Interchange Format – Overview“, Interim Standard, EIA/IS-106, EIA 1994.

[CDIF94b] „CDIF – Framework for Modeling and Extensibility“, Interim Standard, EIA/IS-107, EIA 1994.

[CDIF94c] „CDIF – Transfer Format – General Rules for Syntaxes“, Interim Standard, EIA/IS-108, EIA 1994.

[CDIF94d] „CDIF – Transfer Format – Transfer Format Syntax – SYNTAX.1“, Interim Standard, EIA/IS-109, EIA 1994.

[CDIF94e] „CDIF – Transfer Format – Transfer Format Encoding – ENCODING.1“, Interim Standard, EIA/IS-110, EIA 1994.

[CDIF94f] „CDIF – Integrated Meta-model, Foundation Subject Area“, Interim Standard, EIA/IS-111, EIA 1994.

[CDIF95] „CDIF – Integrated Meta-model, Common Subject Area“, Interim Standard, EIA/IS-112, EIA 1995.

[CDIF96a] Ernst J. (Editor): „State/Event Model Subject Area“, CDIF-DRAFT-STEV-V8, 1996. URL (Stand: 1998-05-14): „ftp://www.cdif.org/pub/drafts/StateEvent/CDIF-DRAFT-STEV-V8.pdf“.

[CDIF96b] „CDIF – Integrated Meta-model, Data Modeling Subject Area“, Interim Standard, EIA/IS-114, EIA 1996.

[CDIF96c] „CDIF – Integrated Meta-model, Data Flow Subject Area“, Interim Standard, EIA/IS-115, EIA 1996.

[CDIF96d] „CDIF – Integrated Meta-model, Presentation Location and Connectivity Subject Area“, Interim Standard, EIA/IS-118, EIA 1996.

[CDIF96e] Pidcock W. (Editor): „CDIF – Integrated Meta-model, Business Process Modeling Subject Area“, CDIF-DRAFT-BPM-V02, 1996. URL (Stand: 1997-03-10): „ftp://ftp.cdif.org/cdif/documents/BPM-Boeing.doc.zip“.

[CDIF96f] Ernst J. (Editor): „CDIF – Integrated Meta-model, Object-oriented Analysis and Design Core Subject Area“, CDIF-DRAFT-OOAD-V01, 1996. URL (Stand: 1998-05-14): „ftp://ftp.cdif.org/cdif/drafts/ObjectAnalysisDesign/CDIF-DRAFT-OOAD-V1.pdf“.

[CDIF96g] Navarro T. (Editor): „CDIF – Integrated Meta-model, Project Management Planning And Scheduling Subject Area“, CDIF-DRAFT-PMPS-V4, 1996. URL (Stand: 1998-05-14): „ftp://ftp.cdif.org/cdif/drafts/ProjectManagement/CDIF-DRAFT-PMPS-V4.doc“.

[CDIF96h] Hill R. (Editor): „CDIF – Integrated Meta-model, Data Definition Subject Area“, CDIF-DRAFT-DDEF-V13R13, 1996. URL (Stand: 1998-05-14): „ftp://ftp.cdif.org/cdif/drafts/DataDefinition/CDIF-DRAFT-DDEF-V13R13.pdf“.

[CDIF97a] Ernst J.: „Using the CDIF Transfer Format for Exchanging UML Models“, in EIA/CDIF committee working document: „CDIF-JE-N34-V2“, EIA/CDIF 1997. URL (Stand: 1998-03-21): „http://www.cdif.org/liaisons/CDIF-JE-N34-V2.pdf“.

[CDIF97b] „CDIF Transfer Format – OMG IDL Bindings“, EIA/IS-734, EIA 1997.

[CDIF97c] Ernst J.: „Introduction to CDIF“, URL (Stand: 1997-03-20): „http://www.cdif.org/intro.html“.

[CDIF97d] Ernst J.: „Evolution of the CDIF Meta-meta-model“, in EIA/CDIF committee working document: „CDIF-JE-N28-V1“, EIA/CDIF 1997.

[CDIF98a] Ernst J. (Editor): „Computer Aided Control Systems Design Subject Area“, CDIF-DRAFT-CACSD-V7, 1998. URL (Stand: 1998-05-17): „ftp://ftp.cdif.org/cdif/drafts/ControlSystemsDesign/CDIF-DRAFT-CACSD-V7.pdf“.

[CDIF98b] Ernst J. (Editor): „CDIF-XML-based Transfer Format“, CDIF-DRAFT-XML-V2, 1998. URL (Stand: 1998-06-13): „`http://www.cdif.org/drafts/XmlSyntax/CDIF-DRAFT-XML-V2.pdf`“.

[Chen76] Chen P.P.: „The Entity Relationship Model – Toward a Unified View of Data“, in: ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976.

[Codd70] Codd E.F.: „A Relational Model of Data for Large Shared Banks“, in: Communications of the ACM 13, No. 6, June 1970.

[Codd79] Codd E.F.: „Extending the Database Relational Model to Capture More Meaning“, in: ACM Transactions on Database Systems 4, No. 4, December 1979.

[Codd90] Codd E.F.: „The Relational Model for Databse Management – Version 2“, Addison-Wesley, Reading: 1990.

[CoLeRi91] Cormen H., Leiserson C., Rivest R.: „Introduction to Algorithms“, Third Printing, The MIT Press, Cambridge: Massachutes 1991.

[CORBA98] „The Common Object Request Broker: Architecture and Specification“, version 2.2, OMG standard, OMG 1998. URL (Stand: 1998-02-28): „`ftp://ftp.omg.org/pub/docs/formal/98-02-01.pdf`“.

[CotPot95] Cotter S., Potel M.: „Inside Taligent Technology“, Addison-Wesley, Reading: Massachusetts 1995.

[Cowl90] Cowlishaw M.F.: „The Rexx Language“, Second Edition, Prentice-Hall, Englewood Cliffs 1990.

[DahMat93] Dahlbom B., Mathiassen L.: „Computers in Context“, Blackwell Publishers, Cambridge: Massachusetts 1993.

[DatDar97] Date C.J., Darwen H.: „A Guide to the SQL Standard“, Addison-Wesley, 4th edition, Readings 1997.

[Date85] Date C.J.: „An Introduction to Database Systems – Volume II“, Addison-Wesley, 4th edition, Reading 1985.

[Date86] Date C.J.: „An Introduction to Database Systems – Volume I“, Addison-Wesley, 4th edition, Reading 1986.

[Date87] Date C.J.: „A Guide to the SQL Standard“, Addison-Wesley, 4th edition, Reading 1987.

[DatWhi88] Date C.J., White C.J.: „A Guide to DB2“, Addison-Wesley, 2nd edition, Reading 1988.

[DeMar87] De Marco T.: „Structured Analysis and System Specification“, Yourdon Inc., Revised, New York, December1978.

[DeRDur94] DeRose S., Durand D.: „Making Hypermedia Work“, Kluwer Academic Publishers, Norwell:Massachusetts 1994.

[Dude83] „DUDEN – Deutsches Universalwörterbuch“, Bibliographisches Institut, Mannheim 1983.

[Ebe97] Ebert J.: „MetaCASE: Generierung und Anpassung von CASE-Werkzeugen“ In Gens W.(Hrsg.): „3. Fachkongress Smalltalk und Java in Industrie und Ausbildung (STJA'97)“, Technische Universität Ilmenau, 1997.

[EbSuUh97] Ebert J., Süttenbach R., Uhe I.: „Meta-CASE in Practice: a Case for KOGGE“, in A. Olive, J. A. Pastor (Hrsg.): „Advanced Information Systems Engineering“, Proceedings of the 9th International Conference, CAiSE'97, Barcelona, Catalonia, Spain, June 16-20, 1997 LNCS 1250, S. 203-216, Berlin, 1997. URL (Stand: 1998-05-30): „`http://www.uni-koblenz.de/~ist/retrieve/ caise.cr.ps.gz`“.

[EbWiDa96a] Ebert J., Winter A., Dahm P., Franzke A., Süttenbach R.: „Graph Based Modeling and Implementation with EER/GRAL“, in Thalheim B. (Hrsg.): „15th International Conference on Conceptual Modeling (ER'96)“, Proceedings Springer, LNCS 1157, pg. 163-178, Berlin, 1996. URL (Stand: 1998-05-30): „`http://www.uni-koblenz.de/~ist/retrieve/AW.ER96LNCS.ps.gz`“.

[EbWiDa96b] Ebert J., Winter A., Dahm P., Franzke A., Süttenbach R.: „Graph Based Modeling and Implementation with EER/GRAL (Extended Version)“, Fachbericht Informatik 11/96, Universität Koblenz-Landau, Institut für Informatik, Koblenz, 1996. URL (Stand: 1998-05-30): „`http://www.uni-koblenz.de/fb4/publikationen/gelbereihe/RR-11-96.ps.gz`“.

[ECMA-149] „Portable Common Tool Environment (PCTE) – Abstract Specification“, 4th edition, ECMA-149, ECMA 1997. URL (Stand: 1998-03-01): „`http://www.ecma.ch/stand/Ecma-149.htm`“.

[ECMA-270] „Portable Common Tool Environment (PCTE) – Mapping from CASE Data Interchange Format (CDIF) to PCTE“, ECMA-270, ECMA 1997. URL (Stand: 1998-03-01): „`http://www.ecma.ch/stand/Ecma-270.htm`“.

[EdKaTj86] Eder J., Kappel G., Tjoa A.M., Wagner R.R.: „BIER – the Behaviour Integrated Entity-Relationship Approach“, in: [Spac87].

[Eich94] Eichler B.: „Informations- und Vermittlungsdienste in offenen verteilten Systemen“, ADV, Wien 1994.

[EinJon85] Ein-Dor P., Jones C.R.: „Information Systems Management: Analytical Tools and Techniques“, Elsevier Science Publishing, New York 1985.

[ElmNav89] Elmasri R., BaCeNa9Navathe S.: „Fundamentals of Database Systems“, The Benjamin/Cummings Publishing, Redwood City 1989.

[Ende97] Ender T.: „Object-Oriented Programming with REXX“, John Wiley & Sons, New York et.al. 1997.

[Ernst98] 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.

[FeJoMi94] Feijs L., Jonkers H., Middelburg C.: „Notations for Software Design“, Springer, London 1994.

[FerSin94] Ferstl O.K., Sinz E.J.: „Grundlagen der Wirtschaftsinformatik, Band 1“, 2. Auflage, Oldenbourg, München/Wien 1994.

[FerSin95] Ferstl O.K., Sinz E.J.: „Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen“, in: Wirtschaftsinformatik 3/95, 37. Jg., 1995.

[FerSin98] Ferstl O.K., Sinz E.J.: „Grundlagen der Wirtschaftsinformatik, Band 1“, 3. Auflage, Oldenbourg, München/Wien 1998.

[Flat90] Flatscher R.G.: „Design relationaler Datenbanken – Daten einer betrieblichen Auftragsabwicklung relational organisieren“, IWT-Verlag, München 1990.

[Flat94] Flatscher R.G.: „Informationssystementwicklung mit CASE an der Wirtschaftsuniversität Wien – Erfahrungen mit ORACLE*CASE“, in: ZfB-Ergänzungsheft, 1994.

[Flat95] Flatscher R.G.: „The Workplace Shell: Objects to the Core“, in: OS/2 Developer: Vol. 7, Summer 1995 a.k.a. „REXX Report, Summer 1995“.

[Flat96a] Flatscher R.G.: „An Overview of the Architecture of EIA's CASE Data Interchange Format (CDIF)“, in: [GI52_96].

[Flat96b] Flatscher R.G.: „Grundlagen der Datenmodellierung und relationaler Datenbanken“, Skriptum zu den Lehrveranstaltungen „Geschäftsprozeßmodellierung“ und „Entwicklung von Informationssystemen (mit ORACLE-CASE)“, Servicebetriebe der ÖH-WU, Wien 1996.

[Flat96c] Flatscher R.G.: „Local Environment and Scopes in Object REXX“, in: Proceedings of the „7th International REXX Symposium, May 12-15, Texas/Austin 1996“, The Rexx Language Association, Raleigh N.C. 1996. Eine Postscriptversion davon findet sich unter der Bezeichnung „LOCAL.PS“ in einem ZIP-Archiv, das aus dem WWW geladen werden kann: URL (Stand: 1998-06-08): „`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx7doc.zip`“.

[Flat96d] Flatscher R.G.: „Object Classes, Meta Classes and Method Resolution in Object REXX“, in: Proceedings of the „7th International REXX Symposium, May 12-15, Texas/Austin 1996“, The Rexx Language Association, Raleigh N.C. 1996. Eine Postscriptversion davon findet sich unter der Bezeichnung „CLASS.PS“ in einem ZIP-Archiv, das aus dem WWW geladen werden kann: URL (Stand: 1998-06-08): „`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx7doc.zip`“.

[Flat96e] Flatscher R.G.: „ORX_ANALYZE.CMD – a Program for Analyzing Directives and Signatures of Object REXX Programs“, in: Proceedings of the „7th International REXX Symposium, May 12-15, Texas/Austin 1996“, The Rexx Language Association, Raleigh N.C. 1996. Eine Postscriptversion davon findet sich unter

der Bezeichnung „ANALYZE.PS“ in einem ZIP-Archiv, das aus dem WWW geladen werden kann: URL (Stand: 1998-06-08): „`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx7doc.zip`“.

[Flat97a] 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.

[Flat97b] Flatscher R.G.: „Utility Routines and Utility Classes for Object REXX“, in: Proceedings of the „8th International REXX Symposium“, April 22-24, Heidelberg/Deutschland 1997. Eine Postscriptversion davon findet sich unter der Bezeichnung „PART1.PS“ in einem ZIP-Archiv, das aus dem WWW geladen werden kann: URL (Stand: 1998-06-08): „`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx8doc.zip`“.

[Flat97c] Flatscher R.G.: „Utility Routines and Utility Classes for Object REXX, Part II“, in: Proceedings of the „8th International REXX Symposium“, April 22-24, Heidelberg/Deutschland 1997. Eine Postscriptversion davon findet sich unter der Bezeichnung „PART2.PS“ in einem ZIP-Archiv, das aus dem WWW geladen werden kann: URL (Stand: 1998-06-08): „`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx8doc.zip`“.

[Flat98a] 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.

[Flat98b] Flatscher R.G.: „Konzeption von Glossaren für HTML-Browser“, in: Czap H., Ohly H.P., Pribbenow S.: „Herausforderungen an die Wissensorganisation: Visualisierung, multimediale Dokumente, Internetstrukturen“, Tagungsband zur 5. Tagung der Deutschen Sektion der Internationalen Gesellschaft für Wissensorganisation, Berlin 1997, ERGON Verlag, Würzburg 1998.

[Fran97] Franzke A.: „GRAL: A Reference Manual“, Fachbericht Informatik 3/97, Universität Koblenz-Landau, Fachbereich Informatik, Koblenz, 1997. URL (Stand: 1998-05-30): „`http://www.uni-koblenz.de/fb4/publikationen/ gelbe-reihe/RR-3-97.ps.gz`“.

[Fran98] Frank U.: „Object-Oriented Modeling Languages: State of the Art and Open Research Questions“, in: Schader M., Korthaus A. (Hrsg.): „The Unified Modeling Language – Technical Aspects and Applications“, Physica-Verlag, Heidelberg 1998.

[GanSar79] Gane C., Sarson T.: „Structured Systems Analysis: Tools and Techniques“, Prentice-Hall, Englewood Cliffs, USA 1979.

[GI52_94a] Rundbrief des GI-Fachausschusses 5.2 Informationssystem Architekturen, 1. Jg. (1994), Heft 1.

[GI52_94b] Rundbrief des GI-Fachausschusses 5.2 Informationssystem Architekturen, 1. Jg. (1994), Heft 2.

[GI52_95a] Rundbrief des GI-Fachausschusses 5.2 Informationssystem Architekturen, 2. Jg. (1995), Heft 1.

[GI52_95b] Rundbrief des GI-Fachausschusses 5.2 Informationssystem Architekturen, 2. Jg. (1995), Heft 2.

[GI52_96] Rundbrief des GI-Fachausschusses 5.2 Informationssystem Architekturen, 3. Jg. (1996), Heft 1.

[Gold84] Goldberg A.: „Smalltalk-80 – The Interactive Programming environment“, Addison-Wesley, Reading: Massachusetts 1984.

[Gold90] Goldfarb C.F.: „The SGML Handbook“, Oxford University Press, Oxford 1990.

[GolRob83] Goldberg A., Robson D.: „Smalltalk-80 – The Language and Its Implementation“, Addison-Wesley, Reading: Massachusetts 1983.

[Groc74] Grochla E. u. Mitarbeiter: „Integrierte Gesamtmodelle der Datenverarbeitung. Entwicklung und Anwendung des Köner Integrationsmodells (KIM)“, Köln 1974.

[HabLey93] Habermann H.-J., Leymann F.: „Repository – Eine Einführung“, R. Oldenbourg Verlag, München/Wien 1993.

[HanRie88] Hansen H.R., Riedl R.: „Strategische langfristige Informationssystemplanung (SISP)“, in: Kurbel K., Strunz H. (Hrsg.): „Handbuch Wirtschaftsinformatik“, Verlag C.E. Poeschel, Stuttgart 1989.

[Hans93] Hansen H.R.: „Beurteilungsmöglichkeiten des Erfolges der Informationsverarbeitung“, in: Journal für Betriebswirtschaft, 43. Jg., Heft 5, 1993.

[Hans96] Hansen H.R.: „Wirtschaftsinformatik I“, UTB 802, 7. Auflage, Stuttgart 1996.

[Hars94] Hars A.: „Referenzdatenmodelle“, Gabler, Wiesbaden 1994.

[Henn80] Henne H.: „Lexikographie“, in: Althaus H.P., Henne H., Wiegand H.E. (Hrsg.): „Lexikon der Germanistischen Linguistik, Studienausgabe IV“, 2. Auflage, Max Niemeyer Verlag, Tübingen 1980.

[HePoSc91] Heinrich L.J., Pomberger G., Schauer R. (Hrsg.): „Die Informationswirtschaft im Unternehmen“, Universitätsverlag Rudolf Trauner, Linz 1991.

[Hess96] Hess Th.: „Entwurf betrieblicher Prozesse“, Deutscher Universitätsverlag, Wiesbaden 1996.

[Hild95] Hildebrand K.: „Informationsmanagement: wettbewerbsorientierte Informationsverarbeitung“, Oldenbourg Verlag, München 1995.

[Hoff87] Hoffmann H.-J. (Hrsg.): „SMALLTALK verstehen und anwenden“, Carl Hanser Verlag, München/Wien 1987.

[HofRoc92] Hofman F.J., Rockardt D.J.: „The Emerging Use of Application Templates“, MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 250, Cambridge: Massachusets 1992.

[Hrus91] Hruschka P. (Hrsg.): „CASE in der Anwendung: Erfahrungen bei der Einführung von CASE“, Carl Hanser Verlag, München/Wien 1991.

[IBM74] „HIPO – A Design Aid and Documentation Technique“, IBM-Order No. GC20-1851, 1974.

[IBM82] „Enterprise Analysis“, IBM Systems Journal, Volume 21, No. 1, 1982.

[IBM85] „Fachausdrücke der Informationsverarbeitung – Wörterbuch und Glossar – Englisch-Deutsch“, IBM Deutschland 1985.

[ISO/IEC-6429] „Information technology – Control functions for coded character sets“, Edition 3, JTC 1/SC 2, International Standard ISO/IEC 6429:1992.

[ISO/IEC-8859-1] „Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No.1“, JTC 1/SC 2, International Standard ISO/IEC 8859-1:1998.

[ISO97a] „Information Technology (IT), Conceptual Modelling Facilities (CSMF)“, Committee Draft SC21 WG3 N2039 for International Standard ISO/IEC 14481, January 1997.

[ISOCDIF98a] „Information Technology – CDIF Framework – Part 1: Overview“, committee draft CD 15474-1, ISO/IEC JTC 1/SC 7 N, 1540R, ISO/IEC 1998.

[ISOCDIF98b] „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.

[ISOCDIF98c] „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.

[ISOCDIF98d] „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.

[ISOCDIF98e] „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.

[ISOCDIF98f] „Information Technology – CDIF Semantic Metamodel – Part 1: Foundation“, committee draft CD 15476-1, ISO/IEC JTC 1/SC 7 N, 1545R, ISO/IEC 1998.

[ISOCDIF98g] „Information Technology – CDIF Semantic Metamodel – Part 2: Common“, committee draft CD 15476-2, ISO/IEC JTC 1/SC 7 N, 1546R, ISO/IEC 1998.

[ISOCDIF98h] „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.

[ISOCDIF98i] „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.

[Jank93] Janko W.: „Informationswirtschaft 1“, Springer-Verlag, Berlin/Heidelberg/New York/Tokyo 1993.

[Jard78] Jardin D.A.: „The ANSI/SPARC DBMS Model“, North-Holland Publishing Company, 2. Auflage, Amsterdam/New York/Oxford, New York 1978.

[Jehl75] Jehle E. (Hrsg.): „Systemforschung in der Betriebswirtschaftslehre“, C.E. Poeschel Verlag, Stuttgart 1975.

[Kami79] Kamitz R. (Hrsg.): „Logik und Wirtschaftswissenschaft“, Duncker & Humblot, Berlin 1979.

[KapSch96] Kappel G., Schrefl M.: „Objektorientierte Informationssysteme – Konzepte, Darstellungsmittel, Methoden“, Springer Verlag, Wien/New York 1996.

[KeLyRo96] Kelly S., Lyytinen K., Rossi M.: „MetaEdit+: A Fully configurable Multi-User and Multi-Tool CASE and CAME Environment“, in: Vassiliou Y., Mylopoulos J. (Hrsg.): „Proceedings of the 8th Conference on Advanced Information Systems Engineering“, Springer Verlag, 1996.

[Klot93] Klotz M.: „Integrierte Anwendungssoftware und Unternehmensorganisation“, Erich Schmidt Verlag, Berlin 1993.

[Köni95] König W. (Hrsg.): „Wirtschaftsinformatik '95“, Physica-Verlag, Heidelberg 1995.

[Kras83] Krasner G. (Hrsg.): „Smalltalk-80 – Bits of History, Words of Advice“, Addison-Wesley, Reading: Massachusetts 1983.

[Krcm97] Krcmar H.: „Informationsmanagement“, Springer, Berlin 1997.

[Kric94] Krickl O.C. (Hrsg.): „Geschäftsprozeßmanagement“, Physica-Verlag, Heidelberg 1994.

[Küff94] Küffmann K.: „Software-Wiederverwendung“, Friedr. Vieweg & Sohn Verlagsgesellschaft, Braunschweig/Wiesbaden 1994.

[KuMeSch89] Kurbel K., Mertens P., Scheer A.-W. (Hrsg.): „Interaktive betriebswirtschaftliche Informations- und Steuerungssysteme“, de Gruyter, Berlin 1989.

[Lang89a] „Langenscheidts Großwörterbuch, 'Der Kleine Muret-Sanders', Deutsch-Englisch“, 5. Auflage, Langenscheidt, Berlin 1989.

[Lang89b] „Langenscheidts Großwörterbuch, 'Der Kleine Muret-Sanders', Englisch-Deutsch“, 4. Auflage, Langenscheidt, Berlin 1989.

[LauLau94] Laudon K., Laudon J.: „Management Information Systems – Organization and Technology“, Third Edition, Macmillan College Publishing Company, New York 1994.

[LeAuBa91] Lehner F., Auer-Rizzi W., Bauer R., Breit K., Lehner J., Reber G.: „Organisationslehre für Wirtschaftsinformatiker“, Carl Hanser Verlag, München/Wien 1991.

[LeMaHi95] Lehner F., Maier R., Hildebrand K.: „Wirtschaftsinformatik – Theoretische Grundlagen“, Carl Hanser Verlag, Wien 1995.

[Lewi95] Lewis T. (Hrsg.): „Object Oriented Application Frameworks“, Manning Publications Co., Greenwich: USA (CT) 1995.

[LuGoNi81] Lundeberg M., Goldkuhl G., Nilsson A.: „Information Systems Development – A Systematic Approach“, Prentice Hall, Englewood-Cliffs, USA 1981.

[Lund82] Lundeberg M.: „The ISAC Approach to Specification of Information Systems and its Application to the Organization of an IFP Working Conference“, in: [OlSoTu82].

[LyyKer94] Lyytinen K., Kerola P.; Kaipala J., Kelly S., Lehto J., Liu H., Marttiin , Oinas-Kukkonen H., Pirhonen J., Rossi M., Smolander K., Tahvanainen V.-P., Tolvanen J.-P.: „MetaPHOR: Metamodeling, Principles, Hypertext, Objects and Repositories“, Technical Report TR-7 (Abschlußbericht), University of Jyväskylä (Department of Computer Science and Information Systems) und University of Oulu (Department of Information Processing Science), Jyväskylä/Oulu 1994. URL (Stand: 1998-05-30): „`http://www.cs.jyu.fi/raportit/metaphor/frapuwww.ps`“.

[Maie96] Maier R.: „Qualität von Datenmodellen“, Deutscher Universitäts Verlag, Wiesbaden 1996.

[Maie98] Maier R.: „Nutzen und Qualität der Datenmodellierung – Ergebnisse einer empirischen Studie“, in: Wirtschaftsinformatik 2/98, 40. Jg., 1998.

[Mar82] Martin J.: „Strategic Data-Planning Methodologies“, Prentice-Hall, Englewood-Cliffs, USA 1982.

[Mare95] Marent C.: „Branchenspezifische Referenzmodelle für betriebswirtschaftsliche Anwendungsbereiche“, in: Wirtschaftsinformatik 3/95, 37. Jg., 1995.

[Mart89] Martin J.: „Information Engineering, Book I: Introduction“, Prentice-Hall, Englewood Cliffs: New Jersey 1989.

[Mart90a] Martin J.: „Information Engineering, Book II: Planning and Analysis“, Prentice-Hall, Englewood Cliffs: New Jersey 1990.

[Mart90b] Martin J.: „Information Engineering, Book III: Design and Construction“, Prentice-Hall, Englewood Cliffs: New Jersey 1990.

[MDIS97] „Meta Data Interchange Specification (MDIS Version 1.1)“, URL (Stand: 1998-04-08): „`http://www.he.net/~metadata/standards/toc.html`“; URL der PDF-Datei: „`http://www.he.net/~metadata/standards/MDIS-11.pdf`“; URL der PostScript-Datei: „`http://www.he.net/~metadata/standards/MDIS-11.ps`“,

[MeBoKö95] Mertens P., Bodendorf F., König W., Picot A., Schumann M.: „Grundzüge der Wirtschaftsinformatik“, 3. Auflage, Springer Verlag, Berlin 1995.

[Mehl96] Mehler-Bicher A.: „Spezifische Architekturmodelle: Zielsetzung, Abgrenzung, Ableitungsmöglichkeiten“, in: [GI52_96].

[MerGri91] Mertens P., Griese J.: „Integrierte Informationsverarbeitung 2, Planungs- und Kontrollsysteme in der Industrie“, 6. Auflage, Dr. Th. Gabler, Wiesbaden 1991.

[MerHol92] Mertens P., Holzner J.: „Eine Gegenüberstellung von Integrationsansätzen der Wirtschaftsinformatik“, in: Wirtschaftsinformatik 1/92, 34. Jg., 1992.

[Mert95] Mertens P.: „Integrierte Informationsverarbeitung 1 – Administrations- und Dispositionssysteme in der Industrie“, 10. Auflage, Verlag Dr. Th. Gabler, Wiesbaden 1995.

[MFaHof91] MacFadden F., Hoffer J.: „Database management“, Third Edition, The Benjamin/Cummings Publishing Comp., Redwood City 1991.

[Mik86] Miksch G.: „Strategische Informationssystemplanung dargestellt am Beispiel der zentralen Verwaltung der WUW“, Service-Fachverlag, Wirtschaftsuniversität, Wien 1986.

[MOF97a] „Meta Object Facility (MOF) – Specification“, version 1.0, OMG standard. URL (Stand: 1998-02-10): „ftp://ftp.omg.org/pub/docs/ad/97-10-02.pdf“.

[MOF97b] „Meta Object Facility – Appendices“, version 1.0, OMG standard. URL (Stand: 1998-02-10): „ftp://ftp.omg.org/pub/docs/ad/97-10-03.pdf“.

[MPHOR94] „MetaPHOR: Metamodeling, Principles, Hypertext, Objects and Repositories“, Computer Science and Information Systems Reports, Technical Reports TR-7, University of jyväskylä, 1994. URL (Stand: 1998-4-09): „http://www.cs.jyu.fi/raportit/metaphor/frapuwww.ps“.

[Neum94] Neumann G.: „Datenmodellierung mit deduktiven Techniken“, Physica-Verlag, Heidelberg 1994.

[ODMG97] „Object Data Management Group: Object Definition Language (ODL), Object Query Language (OQL) – Summary“, submission to OMG, 1997. URL (Stand: 1998-02-20): „ftp://ftp.omg.org/pub/docs/ad/97-01-06.pdf“.

[OlSoTu82] Olle T.W., Sol H.G., Tully V.J. (Hrsg.): „Information System Design Methodologies: A Comparative Review“, North-Holland, Amsterdam 1982.

[OMG97] „Stream-based Model Interchange Format RFP“, OMG RFP, OMG 1997. URL (Stand: 1998-02-28): „ftp://ftp.omg.org/pub/docs/ad/97-12-03.pdf“.

[ORAC92] „CASE*Dictionary Meta-model Version 5.0“, ORACLE Part No. 0258-50-0292, 1992.

[ORAC97] Ohne Autor: „Microsoft's Repository and Oracle's Repository July'97 – The Microsoft Repository and the Oracle® Repository -- Which One Meets Your Needs? A comparison...“, ORACLE-Dokument, URL (Stand: 1998-05-31): „http://www.oracle.com/products/tools/des2k/collateral/ final_ms_repos2.doc“.

[OrfHar97] Orfali R., Harkey D., Edwards J.: „Client/Server Programming with Java and CORBA“, John Wiley & Sons, New York/Chichester/Brisbane/Toronto/Singapore 1997.

[OrHaEd96] Orfali R., Harkey D., Edwards J.: „The Essential Distributed Objects Survival Guide“, John Wiley & Sons, New York/Chichester/Brisbane/Toronto/Singapore 1996.

[Orr77] Orr K.T.: „Structured Systems Development“, Yourdon Inc., New York 1977.

[Ortn83] Ortner E.: „Aspekte einer Konstruktionssprache für den Datenbankentwurf“, S. Toeche-Mittler Verlag, Darmstadt 1993.

[Ortn97] Ortner E.: „Methodenneutraler Fachentwurf: zu den Grundlagen einer anwendungsorientierten Informatik“, Teubner, Stuttgart 1997.

[ÖsBrGa95] Österle H., Brenner C., Gaßner Ch., Gutzwiller T., Hess T.: „Business Engineering, Prozeß- und Systementwicklung – Band 2: Fallbeispiel“, Springer Verlag, Berlin/Heidelberg 1995.

[ÖsBrHi92] Österle H., Brenner W., Hilbers K.: „Unternehmensführung und Informationssystem“, B.G. Teubner, Stuttgart 1992.

[ÖsGuSp94] Österle H., Gutzwiller Th., Sprenger U.: „Referenz-Metamodell für den Entwurf betrieblicher Informationssysteme“, in: [GI52_94a].

[ÖsRiVo96] Österle H., Riehm R., Vogler P.: „Middleware“, Verlag Vieweg, Braunschweig/Wiesbaden 1996.

[Öste95a] Österle H.: „Business Engineering, Prozeß- und Systementwicklung – Band 1: Entwurfstechniken“, Springer Verlag, Berlin/Heidelberg 1995.

[ÖstGut92a] Österle H., Gutzwiller Th.: „Konzepte angewandter Analyse- und Design-Methoden – Band 1: Ein Referenz-Metamodell für die Analyse und das System-Design“, Verlag Angewandte Informationstechnik, Hallbergmoos 1992.

[ÖstGut92b] Österle H., Gutzwiller Th.: „Konzepte angewandter Analyse- und Design-Methoden – Band 2: Ein Beispiel für die Analyse und das System-Design“, Verlag Angewandte Informationstechnik, Hallbergmoos 1992.

[ÖstSan94] Österle H., Sanche J.: „Systementwicklung mit Applikationsplattformen – Erfahrungen bei der Lufthansa und der Schweizerischen Kreditanstalt“, in: Wirtschaftsinformatik, 36. Jg. (1994) 2, S. 145-154.

[PanTau98] Panny W., Taudes A.: „SQL – Eine Einfuehrung in den Sprachstandard und dessen Erweiterungen“ (Arbeitstitel), WU-Wien, in Vorbereitung.

[PicMai93] Picot A., Maier M.: „Interdependenzen zwischen betriebswirtschaftlichen Organisationsmodellen und Informationsmodellen“, in: Information Management, S. 6-15, August 1993.

[PiReWi96] Picot A., Reichwald R., Wigand R.: „Die grenzenlose Unternehmung: Information, Organisation und Management“, Verlag Gabler, Wiesbaden 1996.

[PopKel96] Popp K., Keller G.: „Komposition betrieblicher Anwendungssysteme aus Komponenten – ein zentrales Architektur-Problem des Componentware-Ansatzes“, in: [GI52_96].

[Raue96a] Raue H.: „Wiederverwendbare betriebliche Anwendungssysteme“, Deutscher Universitätsverlag, Wiesbaden 1996.

[Raue96b] Raue H.: „Objektorientierte Entwicklung von wiederverwendbaren betrieblichen Anwendungssystemen auf der Grundlage von Geschäftsprozessen“, in: [GI52_96].

[Rieg95] Rieger W.: „SGML für die Praxis“, Springer, Berlin/Heidelberg 1995.

[Rohl96] Rohloff M.: „Prozeßorientierte Entwicklung von Informationssystem-Architekturen“, in: [GI52_96].

[Ross77] Ross D.T.: „Structured Analysis (SA): A Language for Communicating Ideas“, IEEE Transactions of Software Engineering, SE-3, No. 1, 1977.

[RuBlPr91] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W.: „Object-Oriented Modeling and Design“, Prentice Hall, Englewood Cliffs: N.J. 1991.

[Schad97] Schader M.: „Objektorientierte Datenbanken – Die C++-Anbindung des ODMG-Standards“, Springer, Berlin 1997.

[Sche90] Scheer A.-W.: „EDV-orientierte Betriebswirtschaftslehre“, 4. Auflage, Springer, Berlin 1990.

[Sche91] Scheer A.-W.: „Architektur integrierter Informationssysteme – Grundlagen der Unternehmensmodellierung“, Springer Verlag, Berlin/Heidelberg 1991.

[Sche95] Scheer A.-W.: „Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse“, 6. Auflage, Springer, Berlin/Heidelberg/New York 1995.

[Sche97] Scheer A.-W.: „Unternehmensdatenmodell“, in: Mertens P. et.al. (Hrsg.): „Lexikon der Wirtschaftsinformatik“, Springer, München 1997.

[Sche98a] Scheer A.-W.: „ARIS – Vom Geschäftsprozeß zum Anwendungssystem“, 3. Auflage, Springer, Berlin 1998.

[Sche98b] Scheer A.-W.: „ARIS – Modellierungsmethoden, Metamodelle, Anwendungen“, 3. Auflage, Springer, Berlin 1998.

[Schol90] Scholz-Reiter B.: „CIM – Informations- und Kommunikationssysteme“, R. Oldenbourg Verlag, München/Wien 1990.

[Schr97b] Schreier U.: „Data Dictionary“, in: Mertens P. et.al. (Hrsg.): „Lexikon der Wirtschaftsinformatik“, Springer, München 1997.

[Schr97] Schreier U.: „Entity-Relationship-Darstellung“, in: Mertens P. et.al. (Hrsg.): „Lexikon der Wirtschaftsinformatik“, Springer, München 1997.

[Schul90] Schulz A.: „Software-Entwurf: Methoden und Werkzeuge“, 2. Auflage, R. Oldenbourg Verlag, München 1990.

[ScWeWö94] Scholz-Reiter B., Weichhardt F., Wöhrle G.: „Ansatz zur Integration der Funktionalität von Werkzeugen der Organisationsmodellierung und der Softwareentwicklung (CASE)“, in: [GI52_94a].

[SIEM90] „GRAPES – Referenzmanual“, Siemens AG, Bereich Daten- und Informationstechnik, Ausgabe Juli 1990.

[Sinz88] Sinz E.J.: „Das Strukturierte Entity-Relationship-Modell (SER-Modell)“, in: Angewandte Informatik, Heft 5, Mai 1998.

[Sinz95] Sinz E.J.: „Geschäftsprozesse und Workflow-Systeme – Modelle und Architekturen“, in: [GI52_95b].

[Sinz96] Sinz E.J.: „IS-Architekturen: Aktuelle Anforderungen und Entwicklungen“, in: [GI52_96].

[SowZac92] Sowa F.J., Zachman A.J.: „Extending and Formalizing the Framework for Information Systems Archtitecture“, in: IBM Systems Journal 31 (1992) 3, S. 590-616.

[Spac87] Spaccapietra S. (Hrsg.): „Entity Relationship Approach“, Proceedings of the Fifth International Conference on Entity Relationship Approach, Dijon 1986, North-Holland, Amsterdam 1987.

[Spiv88] Spivey J.M.: „Understanding Z: a specification language and its formal semantics“, Cambridge University Press, Cambridge 1988.

[StaHas97] Stahlknecht P., Hasenkamp U.: „Einführung in die Wirtschaftsinformatik“, 8. Auflage, Springer, Berlin 1997.

[Stec93] Stecher P.: „Building Business Application Systems with the Retail Application Architecture“, in: IBM Systems Journal 32 (1993) 2, S. 278-306.

[Stro92] Stroustrup B.: „Die C++ Programmiersprache“, 2. Auflage, Addison-Wesley, Bonn/München 1992.

[Szil95] Szillat H.: „SGML: Eine Einführung“, Internat. Thomson Publ., Bonn 1995.

[ThoHuf96] Thome R., Hufgard A.: „Continuous System Engineering“, Vogel Verlag, Würzburg 1996.

[TurWah97] Turton T., Wahli U.: „Object Rexx for OS/2 Warp“, Prentice-Hall, London 1997.

[UML97a] „Unified Modeling Language – UML-Compliant Interchange Format“, version 1.0, submission to OMG, 1997. URL (Stand: 1997-03-10): „`ftp://ftp.omg.org/pub/docs/ad/97-01-13.pdf`“.

[UML97b] „Unified Modeling Language – Appendix M3: UML Meta-Metamodel Alignment with MOF and CDIF“, version 1.0, submission to OMG, 1997. URL (Stand: 1997-03-10): „`ftp://ftp.omg.org/pub/docs/ad/97-01-06.pdf`“.

[UML97c] „Unified Modeling Language – UML Summary“, version 1.1, OMG, 1997. URL (Stand: 1998-02-20): „`ftp://ftp.omg.org/pub/docs/ad/97-08-03.pdf`“.

[UML97d] „Unified Modeling Language – UML Semantics“, version 1.1, OMG, 1997. URL (Stand: 1998-02-20): „ftp://ftp.omg.org/pub/docs/ad/97-08-04.pdf“.

[UML97e] „Unified Modeling Language – UML Notation Guide“, version 1.1, OMG, 1997. URL (Stand: 1998-02-20): „ftp://ftp.omg.org/pub/docs/ad/97-08-05.pdf“.

[UML97f] „Appendix M2: UML Meta-Metamodel“, version 1.0, submission to OMG, 1997. URL (Stand: 1997-11-22): „ftp://ftp.omg.org/pub/docs/ad/97-01-06.pdf“.

[Urm96] Urman S.: „ORACLE PL/SQL Programming – The Essential Guide for Every Oracle Programmer“, Osborne McGraw-Hill, Berkeley 1996.

[Vatt93] Vatteroth H.-C.: „PPS und computergestützte Personalarbeit“, Datakontext-Verlag, Köln 1993.

[Vern96] Vernadat F.: „Enterprise Modeling and Integration“, Chapman & Hall, London 1996.

[VeTrUr96] Veneskey G., Trosky W., Urbaniak J.: „Object Rexx by Example“, Aviar, Pittsburgh 1996.

[Vett88] Vetter M.: „Strategie der Anwendungssoftware-Entwicklung – Planung, Prinzipien, Konzepte“, Teubner Verlag, Stuttgart 1988.

[Vett90] Vetter M.: „Strategie der Anwendungssoftware-Entwicklung“, 2. Auflage, B.G. Teubner, Stuttgart 1990.

[Vett91] Vetter M.: „Aufbau betrieblicher Informationssystem mittels objektorientierter, konzeptioneller Datenmodellierung“, 7. Auflage, B.G. Teubner, Stuttgart 1991.

[W3CASETOOLS] „CASE Tools and Manufacturers“, Department of Computer Science and Information Systems, University of Jyväskylä , URL (Stand: 1998-05-30): „http://www.jyu.fi/~kelly/meta/caselist.html“.

[W3CDIF] WWW-Homepage der „Electronic Industries Association (EIA), CASE Data Interchange Format (CDIF) committee“, URL (Stand: 1998-02-18): „http://www.cdif.org“.

[W3CDIF_MA] WWW-Seite der aktiven Mitglieder von EIA/CDIF, URL (Stand: 1998-05-31): „http://www.cdif.org/activemembers.html“.

[W3CDIF_ME] WWW-Seite der ehemals aktiven Mitglieder von EIA/CDIF, URL (Stand: 1998-05-31): „http://www.cdif.org/formermembers.html“.

[W3CDIF_STANDARDS] WWW-Seite mit der Preisliste für die verfügbaren EIA/CDIF-Standards, URL (Stand: 1998-05-31): „http://www.cdif.org/how-to-obtain-standards.html“.

[W3ECMA] WWW-Homepage der „European Computer Manufacturers Association (ECMA)“, URL (Stand: 1998-03-01): „http://www.ecma.ch/“.

[W3EIA] WWW-Homepage der „Electronic Industries Alliance (EIA), CASE Data Interchange Format (CDIF) committee“, URL (Stand: 1998-05-31): „http://www.eia.org“.

[W3FZI] WWW-Homepage des „Forschungszentrums für Informatik“, URL (Stand: 1998-05-31): „http://www.fzi.de/“.

[W3GRAL] WWW-Homepage von: „IST – Graphentechnologie“, Institut für Softwaretechnik (IST), Universität Koblenz/Landau, URL (Stand: 1998-05-30): „http://www.uni-koblenz.de/~ist/gralab.html“.

[W3IDEF0] „Integrated Computer-Aided Manufacturing (ICAM) Function Modeling Manual (IDEF0)“, US Air Force, June 1981, URL (Stand: 1998-04-05): „http://nemo.ncsl.nist.gov/standsp/toc.html“; sowie: „Integration definition for function modeling (IDEF0)“, FIPS Pub 183, NIST 1993. URL(STand: 1998-04-05): „http://nemo.ncsl.nist.gov/idef/standsp/idef0.html“.

[W3ISO] WWW-Homepage der „International Organisation for Standardisation (ISO)“, URL (Stand: 1998-05-31): „http://www.iso.ch“.

[W3KOGGE] WWW-Homepage von: „Metacase (KOGGE)“, Institut für Softwaretechnik (IST), Universität Koblenz/Landau, URL (Stand: 1998-05-30): „http://www.uni-koblenz.de/~ist/kogge.html“.

[W3MDC] WWW-Homepage der „Meta Data Coalition (MDC)“, URL (Stand: 1998-02-17): „http://www.hw.net/~metadata/“.

[W3METAEDIT] „MetaEdit and MetaEdit+“, Department of Computer Science and Information Systems, University of Jyväskylä , URL (Stand: 1998-05-30): „http://www.jyu.fi/~kelly/meta/tools.html“.

[W3METAPHOR] WWW-Homepage von: „MetaPHOR Project“, Department of Computer Science and Information Systems, University of Jyväskylä , URL (Stand: 1998-05-30): „http://www.jyu.fi/~kelly/meta/metaphor.html“.

[W3NetRexx] WWW-Homepage von NetRexx: „NetRexx Home Page“, URL (Stand: 1998-03-10): „http://www2.hursley.ibm.com/netrexx/“.

[W3ODMG] WWW-Homepage der „Object Database Management Group (ODMG)“, URL (Stand: 1998-02-21): „http://www.odmg.org“.

[W3OMG] WWW-Homepage der Object Management Group (abgekürzt: „OMG“), URL (Stand: 1998-05-30): „http://www.omg.org“.

[W3ORACLE] WWW-Homepage der Firma ORACLE, URL (Stand: 1998-05-31): „http://www.oracle.com“.

[W3ORexx-SQL] Blumel J.: WWW-Seite für das Übertragen des Object Rexx Frameworks für [W3Rexx-SQL], URL (Stand: 1998-06-08): „http://www.tmc.tulane.edu/jblumel/orexxsql.zip“ oder „http://hobbes.nmsu.edu/pub/os2/dev/orexx/orexxsql.zip“.

[W3ORexx] WWW-Homepage von Object Rexx: „Object Rexx Home Page“, URL (Stand: 1998-03-10): „http://www2.hursley.ibm.com/orexx/“.

[W3ORX-LINUX] WWW-Seite für das Übertragen von „Object Rexx für Linux“, URL (Stand: 1998-06-08): „http://service2.boulder.ibm.com/dl/rexx/orexxlinux-d“.

[W3ORX-OS2] WWW-Seite für das Übertragen von „Object Rexx für OS/2 Warp 3 (Update für OS/2 Warp 4)“, URL (Stand: 1998-06-08): „http://service.software.ibm.com/dl/rexx/orexx30-d“.

[W3ORX7] WWW-Seite für das Übertragen des Analysetools für Object Rexx Programme, URL (Stand: 1998-03-10): „ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx7.zip“.

[W3ORX8] WWW-Seite für das Übertragen der Object Rexx Utilities, URL (Stand: 1998-03-10): „ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx8.zip“.

[W3PARADIGM] WWW-Homepage „PLATINUM Paradigm Plus“, URL (Stand: 1998-05-31): „http://www.platinum.com/products/appdev/pplus_ps.htm“.

[W3PLATINUM] WWW-Homepage von der Firma „Platinum Technology“, URL (Stand: 1998-05-30): „http://www.platinum.com/“.

[W3Rexx-SQL] Hessling M.: WWW-Homepage für „RexxSQL“, URL (Stand: 1998-06-08): „http://www.lightlink.com/hessling/#REXXSQL“.

[W3Rexx] WWW-Homepage von Rexx: „Rexx Home Page“, URL (Stand: 1998-03-10): „http://www2.hursley.ibm.com/rexx/“.

[W3UML] WWW-Homepage von Rational („Unified Modeling Language“), URL (Stand: 1998-05-30): „http://www.rational.com/uml“.

[W3VHB] WWW-Homepage des Verband der Hochschullehrer für Betriebswirtschaft e.V., URL(Stand: 1998-05-27): „http://www.bwl.uni-muenchen.de/vhb/ index.html“.

[W3WI] „Gegenstand der Wirtschaftsinformatik“, Definition der Wissenschaftlichen Kommission Wirtschaftsinformatik im Verband der Hochschullehrer für Betriebswirtschaft e.V., URL (Stand: 1998-05-27): „http://isw.wiwi.uni-frankfurt.de/wi/gegensta.html“.

[WahHolTur97] Wahli U., Holder I., Turton T.: „Object REXX for Windows 95/NT, With OODialog“, Prentice Hall, London 1997.

[Wede76] Wedekind H.: „Systemanalyse – Die Entwicklung von Anwendungssystemen für Datenverarbeitungsanlagen“, 2. Auflage, Carl Hanser Verlag, München/Wien, München 1976.

[Wede92] Wedekind H.: „Objektorientierte Schema-Entwicklung“, BI Wissenschaftsverlag, Mannheim/Wien/Zürich 1992.

[Wede97] Wedekind H.: „Datenmodell“, in: Mertens P. et.al. (Hrsg.): „Lexikon der Wirtschaftsinformatik“, Springer, München 1997.

[Wend93] Wendt S.: „Defizite im Software Engineering“, in: Informatik-Spektrum 16 (1993), 1, S. 34-38.

[Wern95] Werner D.: „Taschenbuch der Informatik“, Fachbuchverlag Leipzig, Leipzig 1995.

[WinEbe94] Winter A., Ebert J.: „Ein Referenz-Schema zur Organisationsbeschreibung“, in: [GI52_94b].

[Wint95] Winter R.: „Wiederverwendung von Schemawissen auf der Grundlage eines erweiterten konzeptionellen Modells betrieblicher Informationssysteme“, in: [GI52_95a].

[Wöhe90] Wöhe G.: „Einführung in die Allgemeine Betriebswirtschaftslehre“, 17. Auflage, Verlag Franz Vahlen, München 1990.

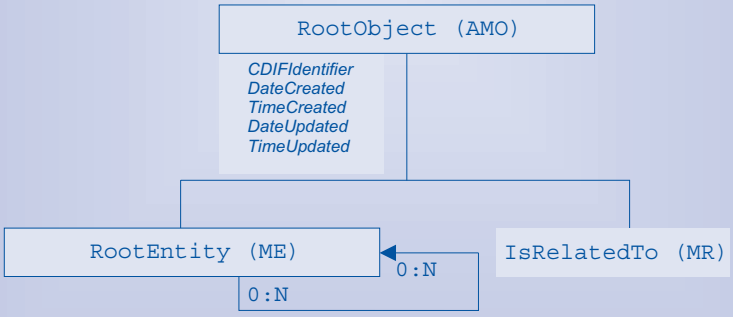
[Wües96] Wüest G.: „Kennzahlen und Kennzahlensysteme“, in: Journal für Betriebswirtschaft, 46. Jg., Heft 2, 1996.

[YouCon79] Yourdon E., Constantine L.L.: „Structured Design – Fundamentals of Computer Program and Systems Design“, Prentice-Hall, Englewood Cliffs 1979.

[Your92] Yourdon E.: „Moderne strukturierte Analyse“, Wolfram's Fachverlag, Attenkirchen 1992.

EIA/CDIF wurde ursprünglich als amerikanischer Industriestandard für den Austausch von CASE-Modelldaten erstellt. Die Arbeiten, die 1987 begannen, führten 1994 zu einem Satz von EIA/CDIF-Standards, die von der internationalen Standardisierungsorganisation ISO/IETC (JTC1 SC7/WG11) übernommen und nach Überarbeitungen Ende 1998 zu einem eigenen, internationalen Standard (flISO/CDIFf führten. Aufgrund der erwarteten Bedeutung von ISO/CDIF wird in diesem Buch immer wieder auf die wenigen Unterschiede zu EIA/CDIF eingegangen, sodaß es auch für die schnelle Einarbeitung in ISO/CDIF hervorragend geeignet ist.
Eine wichtige Rolle spielt EIA/CDIF im Zusammenhang mit der Definition von Modelldatenaustauschregeln für die OMG (flObjec Managment Groupfl) indem die Unterstützung von EIA/CDIF im Rahmen des fiRFP (fiReques for Proposalfl für fiSMIF (fl tream-based Interchange Formatfl zwingend vorgeschrieben wird.

Diese Arbeit stellt das gesamte EIA/CDIF-Standardwerk erstmalig in einer systematischen und zusammenhängenden Form so vor, daß der Leser in die Lage versetzt wird, die bis 1998 standardisierten EIA/CDIF-Meta-Modelle selbst für den Austausch von Modelldaten zu benutzen. Darüber hinaus soll nach dem Studium dieses Buches auch der Entwurf von eigenen EIA/CDIF-konformen Meta-Modellen möglich werden.
Für weitergehende Implementierungen von Meta-Modellen in relationalen Datenbanksystemen und für objektorientierte Programmiersprachen wurden Spezifikationen entworfen und beispielhaft in Oracle und Object Rexx umgesetzt.



Für diese Arbeit wurden sämtliche Definitionen der bis Anfang 1998 standardisierten EIA/CDIF-Meta-Modelle analysiert und überprüft sowie erstmals das hypothetische fiIntegriert EIA/CDIF-Meta-Modellf physisch erarbeitet und dokumentiert.
Die Analyseergebnisse wurden unter anderem auch in Form von unterschiedlichen Referenztabellen so aufbereitet, daß dieses Buch auch als ein einmaliges Nachschlagwerk für CDIF-Meta-Modelle im allgemeinen (EIA und ISO) dienen kann.
Für Meta-Modellierer ist der dokumentierte Entwurf eines EIA/CDIF-konformen Meta-Modells (fiM2Levelf interessant, das es erlaubt, Meta-Meta-Modellobjekte semantisch ausführlicher zu beschreiben, als dies mit dem EIA-Meta-Meta-Modell selbst möglich ist.

ISBN 3-901198-09-1
9 783901 198090

ADV-Verlag
Trattnerhof 2
A-1010 Wien/Vienna