



TECHNICAL REPORT	EIA-IS-114™	
	Issued 1996-12 Stabilized 2015-07	
CDIF - Integrated Meta-model Data Modeling Subject Area		

NOTICE

This document has been taken directly from the original TechAmerica document and contains only minor editorial and format changes required to bring it into conformance with the publishing requirements of SAE Technical Standards. The release of this document is intended to replace the original with the SAE International document. Any numbers established by the original document remain unchanged.

The original document was adopted as an SAE publication under the provisions of the SAE Technical Standards Board (TSB) Rules and Regulations (TSB 001) pertaining to accelerated adoption of specifications and standards. TSB rules provide for (a) the publication of portions of unrevised specifications and standards without consensus voting at the SAE committee level, and (b) the use of the existing specification or standard format.

SAE Technical Standards Board Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2016 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
<http://www.sae.org>

SAE WEB ADDRESS:

Full document details: [SAE EIAIS114](#)

SAE values your input. To provide feedback on this Technical Report, please visit <http://www.sae.org/technical/standards/EIAIS114>

Documents
Visit Document Sharing Hub

Interim Standard

EIA/IS-114

CDIF

CASE Data Interchange Format
A Division of Electronic Industries Association

CDIF
Integrated Meta-model
Data Modeling Subject Area

December 1996

Electronic Industries Association
CDIF Technical Committee

NOTICE

EIA Engineering standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such standards and publications shall not in any respect preclude any member or nonmember of EIA from manufacturing or selling products not conforming to such standards and publications, nor shall the existence of such standards and publications preclude their voluntary use by those other than EIA members, whether the standard is to be used either domestically or internationally.

Recommended standards and publications are adopted by EIA without regard to whether or not their adoption may involve patents on articles, materials or processes. By such action, EIA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standard or publication.

EIA INTERIM STANDARDS

EIA Interim Standards contain information deemed to be of technical value to the industry, and are published at the request of the originating Committee without necessarily following the rigorous public review and resolution of comments which is a procedural part of the development of an EIA Recommended Standard.

EIA Interim Standards should be reviewed on an annual basis by the formulating Committee and a decision made on whether to proceed to develop an EIA Recommended Standard on this subject. EIA Interim Standards must be canceled by the Committee and removed from the EIA Standards Catalog before the end of their third year of existence.

This EIA standard is considered to have international standardization implication, but the International Standardization Organization activity has not progressed to the point where a valid comparison between the EIA standard and the ISO work can be made.

COPYRIGHT 1996

Published by
ELECTRONIC INDUSTRIES ASSOCIATION
Engineering Department
2500 Wilson Blvd.
Arlington, VA 22201

PRICE: Please refer to the current
Catalog of EIA & JEDEC STANDARDS & ENGINEERING PUBLICATIONS
or contact EIA Electronic Information Group
Phone: (703) 907-2545, FAX: (703) 907-7501

All rights reserved
Printed in U.S.A.

ISBN 0-7908-0020-9

PREFACE

This standard will assist the vendors and users of CASE tools in developing mechanisms for interchanging information between CASE tools. This standard specifies an element of a family of related standards. When used together, these standards specify a mechanism for transferring information between CASE tools.

***EIA/IS-106 CDIF - CASE Data Interchange Format - Overview and EIA/IS-107 CDIF - Framework for Modeling and Extensibility* should be read first when initially exploring CDIF.** The first explains the overall CDIF Architecture and how the family of standards fits together. The second explains the scope, and modeling approach in CDIF. The CDIF Meta-model and extensibility mechanism are also defined in that document.

This standard, *EIA/IS-114 CDIF - Integrated Meta-model - Data Modeling Subject Area*, explains the Data Modeling subject area. This is one of the subject areas of the CDIF Integrated Meta-model, which is used to ensure that the information held by tools communicating using CDIF is transferred with an agreed meaning. It covers the information required to express Entity-Relationship-Attribute modeling and Logical Database Design concepts.

This standard has been developed with the wide support and participation of vendors, users, academia and government involved in or familiar with the CASE industry, its products and the general requirements associated with interchanging information between these products.

The material contained in this publication was formulated under the cognizance of the CDIF Technical Committee.

CONTENTS

1. SCOPE	1
2. INTRODUCTION.....	3
2.1 Audience.....	3
2.2 Structure of this Document	3
2.3 Conventions	6
3. STRUCTURE OF SUBJECT AREA DEFINITION.....	9
3.1 Introduction.....	9
3.2 Structure of Subject Area Overview	9
3.2.1 Summary	9
3.2.2 Description of Major Concepts	9
3.2.3 Diagram(s)	9
3.2.4 AttributableMetaObject Hierarchy	12
3.2.5 MetaEntity and MetaRelationship Summaries	13
3.3 Structure of Subject Area Detailed Definitions.....	14
3.3.1 Summary	14
3.3.2 Subject Area Definition	14
3.3.3 MetaEntity Definitions.....	14
3.4 Structure of Referenced Meta-object Definitions	18
4. SUBJECT AREA OVERVIEW	20
4.1 Description.....	20
4.1.1 Introduction.....	20
4.1.2 Entities	21
4.1.3 Relationships.....	21
4.1.3.1 Binary Relationship	21
4.1.3.2 N-ary Relationship	22
4.1.3.3 Simple Relationship	22
4.1.3.4 Complex Relationship.....	22
4.1.4 Roles	25
4.1.5 Cardinalities	27
4.1.5.1 Outer Cardinalities	28
4.1.5.2 Inner Cardinalities.....	28
4.1.6 Multi-player Roles	31
4.1.7 Role Constraints.....	33
4.1.8 Attributes.....	36

4.1.9 Keys and Access Paths.....	38
4.1.10 Referential Integrity	43
4.1.11 Supertypes and Subtypes	46
4.1.11.1 Coverage	47
4.1.11.2 Exclusivity	48
4.1.11.3 Orthogonality	49
4.1.11.4 Inheritance.....	51
4.1.12 Refinement of Inherited Characteristics	53
4.1.12.1 Refinement of Attribute Characteristics	53
4.1.12.2 Refinement of Relationship Characteristics.....	53
4.1.13 Subsets	56
4.1.14 Clusters	57
4.1.15 Projections.....	58
4.1.16 Computable Languages.....	62
4.1.17 Formats for Computable Values	62
4.2 Diagrams	63
4.3 AttributableMetaObject Hierarchy	69
4.4 MetaEntity Summary	71
4.5 MetaRelationship Summary	81
 5. SUBJECT AREA DETAILED DEFINITIONS.....	 88
5.1 Introduction.....	88
5.2 SubjectArea Definition	88
5.3 MetaEntity Definitions	89
5.3.1 AccessPath	89
5.3.2 Attribute	92
5.3.3 CandidateKey.....	96
5.3.4 Cluster	98
5.3.5 ComponentObject	100
5.3.6 DataModel.....	101
5.3.7 DataModelObject.....	104
5.3.8 DataModelSubset.....	106
5.3.9 DefinitionObject	108
5.3.10 Entity.....	112
5.3.11 ForeignKey	125
5.3.12 InheritableDataModelObject.....	127
5.3.13 Key	129
5.3.14 ProjectedAttribute	132
5.3.15 ProjectionComponent	135
5.3.16 Relationship	138
5.3.17 Role	141
5.3.18 RoleConstraint	145

5.3.19 RolePlayer.....	148
5.3.20 SemanticInformationObject.....	168
5.3.21 SubtypeSet	171
5.3.22 SubtypeSetMembershipCriterion.....	174
5.4 MetaRelationship Definitions	177
5.4.1 AccessPath.Incorporates.Attribute.....	177
5.4.2 AccessPath.Instantiates.Key	179
5.4.3 Attribute.IsDiscriminatorFor.SubtypeSetMembershipCriterion.....	180
5.4.4 Attribute.IsInheritedFrom.Attribute.....	181
5.4.5 CandidateKey.Incorporates.ForeignKey.....	182
5.4.6 Cluster.Collects.DataModelObject	183
5.4.7 DataModel.Collects.DataModelObject.....	184
5.4.8 DataModelObject.ActsAs.RolePlayer	185
5.4.9 DataModelObject.IsMemberOf.DataModelSubset.....	186
5.4.10 DataModelSubset.Excludes.Attribute.....	187
5.4.11 DataModelSubset.IsSubsetOf.DataModel	189
5.4.12 DefinitionObject.Contains.ComponentObject.....	190
5.4.13 DefinitionObject.IsConstructedWith.ProjectionComponent	192
5.4.14 Entity.IsAccessedUsing.AccessPath.....	194
5.4.15 Entity.IsIdentifiedBy.Key	195
5.4.16 ForeignKey.Incorporates.RolePlayer.....	196
5.4.17 ForeignKey.References.CandidateKey	197
5.4.18 InheritableDataModelObject.IsSubtypeIn.SubtypeSet	198
5.4.19 InheritableDataModelObject.IsSupertypeFor.SubtypeSet.....	201
5.4.20 Key.Incorporates.Attribute.....	202
5.4.21 Key.Incorporates.SemanticInformationObject	204
5.4.22 ProjectedAttribute.IsProjectionOf.Attribute	206
5.4.23 ProjectionComponent.IsFullProjectionOf.DefinitionObject	207
5.4.24 ProjectionComponent.IsProjectionOf.Attribute	208
5.4.25 Role.BelongsTo.Relationship	210
5.4.26 RoleConstraint.Incorporates.RoleConstraint	211
5.4.27 RoleConstraint.Incorporates.RolePlayer.....	212
5.4.28 RoleConstraint.Incorporates.SemanticInformationObject.....	213
5.4.29 RolePlayer.IsSupportedBy.Key	214
5.4.30 RolePlayer.Plays.Role.....	215
5.4.31 RolePlayer.Refines.RolePlayer.....	216
5.4.32 RolePlayer.RefinesForSubtype.DataModelObject	217
5.4.33 SubtypeSet.Specifies.SubtypeSetMembershipCriterion.....	218
5.4.34 SubtypeSetMembershipCriterion.Selects.InheritableDataModelObject ..	219
6. REFERENCED OBJECT DEFINITIONS	220

6.1 AttributableMetaObject Definitions	220
6.1.1 RootObject	220
6.2 MetaEntity Definitions	224
6.2.1 RootEntity	224
6.3 MetaRelationship Definitions	226
6.3.1 ComponentObject.References.DefinitionObject	226
6.3.2 RootEntity.IsRelatedTo.RootEntity	227
 APPENDIX A EXAMPLE	 230
A1. Introduction	230
A2. Details of Example	230
A3. CDIF Objects for Example.....	232
A3.1 Data Model.....	232
A3.2 Entities	232
A3.3 Subtyping	234
A3.4 Relationships	235
A3.5 Mutual Exclusivity	241
A3.6 Attributes.....	242
 APPENDIX B QUESTIONS AND ANSWERS	 246
 GLOSSARY	 250
 INDEX.....	 258
 RELATED EIA STANDARDS.....	 INSIDE BACK COVER

FIGURES

Figure 1 Position in the CDIF Family of Standards	1
Figure 2 Conventions for Representing Meta-entities	10
Figure 3 Conventions for Representing Meta-relationships	10
Figure 4 Conceptual Diagram and an Instance Diagram	12
Figure 5 Diagrammatic Conventions - Keys	20
Figure 6 Binary Relationship	21
Figure 7 Ternary Relationship	22
Figure 8 Complex Binary Relationship	23
Figure 9 Instance Diagram showing a Complex Relationship	24
Figure 10 Roles	25
Figure 11 Reflexive or Recursive Relationship	26
Figure 12 Examples of Types of Binary Relationships	27
Figure 13 Outer Cardinalities	28
Figure 14 Inner Cardinalities	29
Figure 15 Inner Cardinalities and Attributed Binary Relationships	30
Figure 16 Multi-player Roles and Heterogeneous Collections	31
Figure 17 Multi-player Roles and Common Outer Cardinality	32
Figure 18 Role Constraint Network	33
Figure 19 Constraints on Role Participation	34
Figure 20 Instance Diagram showing Role Constraints	35
Figure 21 Attributed <i>Relationships</i> , <i>Roles</i> and <i>RolePlayers</i>	37
Figure 22 Foreign Keys	39
Figure 23 <i>RolePlayer</i> as a component of a <i>ForeignKey</i>	40
Figure 24 <i>Key</i> support of <i>RolePlayer</i>	41
Figure 25 <i>Key</i> support for <i>RolePlayer</i> in a Reflexive <i>Relationship</i>	42
Figure 26 Subtype Set Discriminator	47
Figure 27 Abstract Supertype	48
Figure 28 Concrete Supertype	48
Figure 29 Non-exclusive Subtype Set	49
Figure 30 Orthogonal Subtype Sets	50
Figure 31 Instance Diagram showing Orthogonal Subtype Sets	50
Figure 32 Overlapping Subtype Sets	51
Figure 33 Inheritance	52
Figure 34 Refinement of Cardinalities	54
Figure 35 Instance Diagram showing refinement of Cardinalities.	55
Figure 36 Subsets	56
Figure 37 <i>Clusters</i>	58
Figure 38 Instance Diagram showing Simple Projection	59
Figure 39 Instance Diagram showing Complex Projection	60
Figure 40 Instance Diagram showing <i>Attribute</i> Renaming	61

Figure 41 Data Modeling Subject Area Overview.....	63
Figure 42 Partitioning/View Mechanisms	64
Figure 43 <i>Keys</i> and <i>AccessPath</i>	65
Figure 44 <i>Roles</i> , <i>RolePlayers</i> and <i>RoleConstraints</i>	66
Figure 45 Connection between Data Modeling and Common Subject Areas	67
Figure 46 Connection between Data Modeling and Data Definition Subject Areas	68
Figure A1 Entity-Relationship Example.....	231
Figure B1 Binary relationship in CDIF modeling technique notation.....	246
Figure B2 Binary relationship with assigned 1:1 outer cardinality values	247
Figure B3 Binary relationship with inner and outer cardinality values	247

TABLES

Table 1 Contents of Subject Area Definition.....	14
Table 2 Contents of Meta-entity Definition.....	15
Table 3 Contents of Local Meta-attribute Definition.....	16
Table 4 Contents of Meta-relationship Definition	17
Table 5 Referential Integrity Rules.....	43
Table A1 Attribute List for Example	231

1. SCOPE

The CDIF Family of Standards is primarily designed to be used as a description of a mechanism for transferring information between CASE tools. It facilitates a successful transfer when the authors of the importing and exporting tools have nothing in common except an agreement to conform to CDIF. The language that is defined for the Transfer Format also has applicability as a general language for Import/Export from repositories. The CDIF Integrated Meta-model defined for CASE also has applicability as the basis of standard definitions for use in repositories.

The standards that form the complete family of CDIF Standards are documented in *EIA/IS-106 CDIF - CASE Data Interchange Format - Overview*. These standards cover the overall framework, the transfer format and the CDIF Integrated Meta-model.

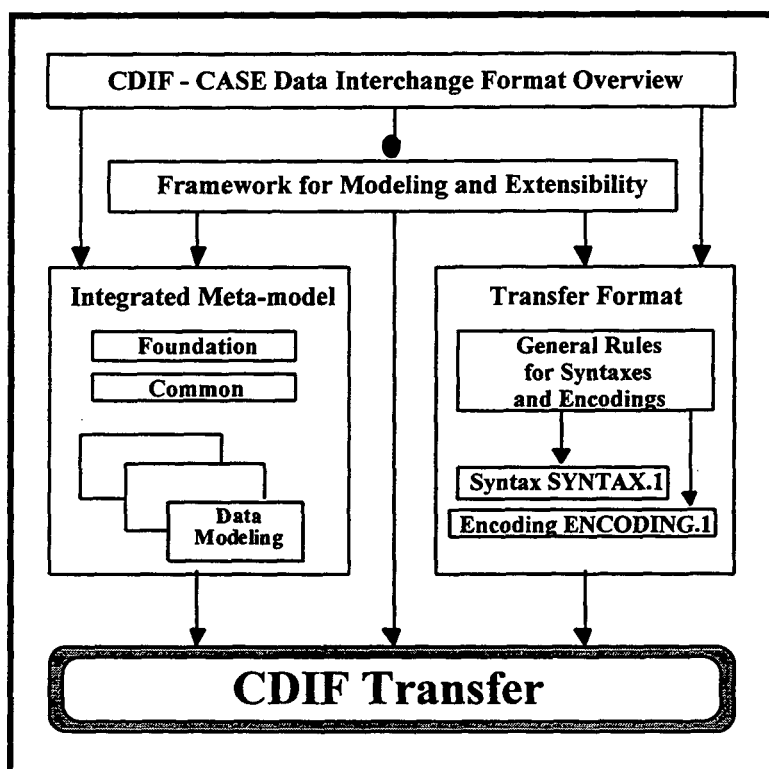


Figure 1
Position in the CDIF Family of Standards

The diagram in Figure 1 depicts the various standards that comprise the CDIF Family of Standards. The shaded box depicts this Standard and its position in the CDIF Family of Standards.

This standard, *EIA/IS-114 CDIF - Integrated Meta-model - Data Modeling Subject Area*, defines the portion of the CDIF Integrated Meta-model that supports general Data Modeling. This includes Entity-Relationship-Attribute Modeling as well as Logical Database Design.

Scope

2. INTRODUCTION

2.1 Audience

This document is intended to be used by anyone wishing to understand and/or use CDIF. This document provides a definition of a single subject area of the CDIF Integrated Meta-model. It is suitable for:

- those evaluating CDIF
- those who wish to understand the principles and concepts of a CDIF transfer
- those developing importers and exporters.

The document *EIA/IS-106 CDIF - CASE Data Interchange Format - Overview* and the document *EIA/IS-107 CDIF - Framework for Modeling and Extensibility* should be read first when initially exploring CDIF and before attempting to read other documents in the CDIF Family of Standards.

While there are no specific prerequisites for reading this document, it will be helpful for the reader to have familiarity with the following:

- Entity-Relationship-Attribute modeling
- CASE tools
- Information repositories
- Data dictionaries
- Multiple meta-layer modeling.

2.2 Structure of this Document

This document is organized into the following sections:

Section 1 'Scope'

This describes the CDIF Family of Standards and positions the current document within the set of documents.

Section 2 'Introduction'

This describes the intended audience, structure and conventions used in the document.

Section 3 'Structure of Subject Area Definition'

This describes in detail the structure of the subsequent sections of this standard including the notation and layout of the diagrams and the detailed definitions.

Section 4 'Subject Area Overview'

This gives an overview of the coverage of this subject area.

Section 5 'Subject Area Detailed Definitions'

This gives the detailed definitions of all the objects used in the subject area.

Section 6 'Referenced Meta-object Definitions'

This gives the detailed definitions of objects that are not used in the subject area defined in this Standard, but are referenced by objects that are. These definitions are included to avoid the need to reference multiple standards to obtain the complete definition of the current subject area.

Appendix A 'Examples'

This gives one or more examples of how information is represented using this CDIF Subject Area. Where there are different technique variants used within the coverage of a single Subject Area, the examples attempt to show how some of these different variants are represented in the CDIF Integrated Meta-model.

Appendix B 'Questions and Answers'

This includes detailed responses to questions that have been raised by those reviewing this document. It is intended that the inclusion of these will aid the reader in understanding this document, and related issues concerning CDIF.

Glossary

The Glossary contains definitions of terms used within this Standard.

Index

A comprehensive index is provided.

References

The References section contains references to other standards and documents used within this Standard.

2.3 Conventions

The typographical and naming conventions defined below are used throughout the CDIF Family of Standards.

- Double quotes are used to introduce new terms (e.g., "model layer")
- Bold type is used for emphasis (e.g., this is **important**)
- All meta-objects and meta-meta-objects in CDIF (in meta-models and meta-meta-models) are named by concatenating all the words that name the meta-object or meta-meta-object; the first letter of each word is upper-case, the rest are lower-case. These names are shown in italic (e.g., *MetaAttribute*, *AttributeDerivation*, *IsDrawnUsing*, *IsOptional*)
- The names of objects and values for meta-attributes that appear in examples and instance diagrams are shown in bold italic (e.g., ***Customer***, ***True***, ***100***)

Full details of the CDIF Graphical Notation used in the Meta-model and Meta-meta-model can be found in *EIA/IS-107 CDIF - Framework for Modeling and Extensibility*. A summary of the major concepts is given in Section 3.2.3, Diagram(s).

This page is intentionally blank.

3. STRUCTURE OF SUBJECT AREA DEFINITION

3.1 Introduction

The following subsections describe how the subject area is defined in the remainder of this document. The subject area is first introduced in the Subject Area Overview Section, and the detailed definitions of the objects contained in the subject area are given in the Subject Area Definitions Section.

3.2 Structure of Subject Area Overview

3.2.1 Summary

The Subject Area Overview Section consists of the following sub-sections:

- Description of Major Concepts
- Diagram(s)
- AttributableMetaObject Hierarchy
- MetaEntity Summary
- MetaRelationship Summary

3.2.2 Description of Major Concepts

This provides a textual description of the major concepts in the subject area. It explains the coverage of this subject area and outlines what it is capable of representing.

3.2.3 Diagram(s)

Included in the Subject Area Overview are various diagrams. These diagrams are intended to graphically present an overview of the meta-entities and meta-relationships used in this subject area. All CDIF subject area diagrams use the following diagramming notation conventions.

Meta-entities used in the subject area are shown as rectangular boxes, with the name of the meta-entity on a white background. An example is the meta-entity *A* in Figure 2. An additional special notation may be used to represent meta-entities on diagrams of meta-models. If the meta-entity is not used in the subject area, but is used in another subject area and is included for clarity, the box shall be shaded. An example is the meta-entity *B* in Figure 2.

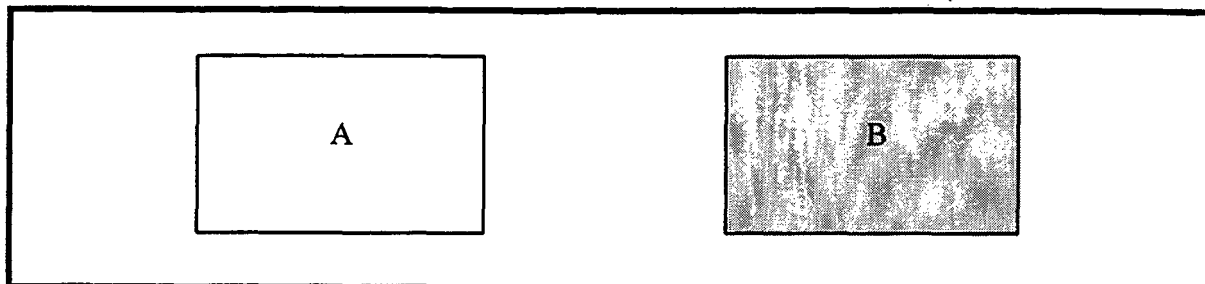


Figure 2
Conventions for Representing Meta-entities

Meta-relationships are normally represented by a line joining the source and destination entities, with the name of the meta-relationship given next to the line. An arrow on the line indicates the direction in which to read the name. The meta-relationship *C* in Figure 3 is an example of this. Additionally, two special notations may be used to represent meta-relationships on diagrams of meta-models.

If a meta-relationship is included in the diagram for clarification, but it is not used in the subject area, the text of the meta-relationship name shall be back-shaded. This is shown for the meta-relationship *D* in Figure 3.

If a supertype meta-entity that participates in a meta-relationship is not included in the diagram, the inherited meta-relationship shall be shown on the diagram as a broken line to/from the inheriting subtype meta-entity. This is shown for the meta-relationship *E* in Figure 3.

Subtype meta-relationships are shown with the meta-relationship name in italics. This is shown for the meta-relationship *F* in Figure 3.

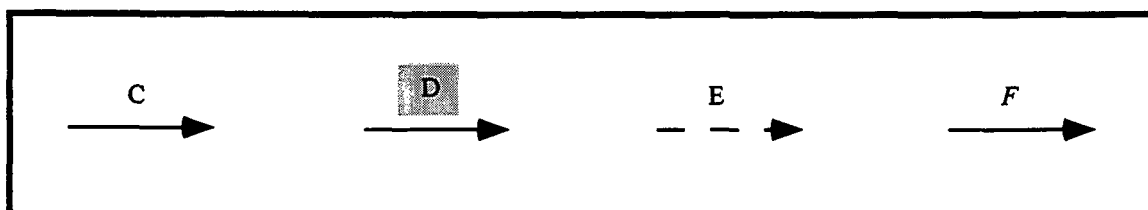


Figure 3
Conventions for Representing Meta-relationships

Full details of the CDIF Graphical Notation used in the Meta-model and the Meta-meta-model can be found in *EIA/IS-107 CDIF - Framework for Modeling and Extensibility*.

To aid in the understanding of some of the concepts modeled in this subject area, instance diagrams have been used. These diagrams differ from those described above in that an instance diagram depicts one or more occurrences or instances of meta-entities and meta-relationships

depicted in the diagrams previously described. Where one of the diagrams previously described may show that the meta-model includes an *Automobile* meta-entity, an instance diagram might depict *Honda*, *Ford*, *Mercedes*, and *Fiat* as instances of *Automobile*.

In an instance diagram, instances of meta-entities are depicted as boxes with a horizontal divider. Above the divider is the name of the meta-entity that is instantiated. Below the divider may be a name or a value assigned to a significant meta-attribute of that meta-entity. Where the name is inherited, inferred, or otherwise not significant to the example (e.g., some associative meta-entity instances), no name or value may be present.

Meta-relationships are depicted as arrows, just as in the previous diagrams described. They do not, however, have cardinality or exclusivity indicators, since none are necessary.

Meta-attributes of both meta-entities and meta-relationships are represented as:

[meta-attribute name =] *value*

where the name of the meta-attribute may be omitted depending on the ease of understanding the example with or without the name (e.g., *Name* might be omitted, *MinLength* might not).

When an instance of a meta-entity or meta-relationship not in this Subject Area is included for clarity, the box/arrow shall be shaded just as in the E-R Diagrams.

Figure 4 shows both the conceptual diagram and an instance diagram.

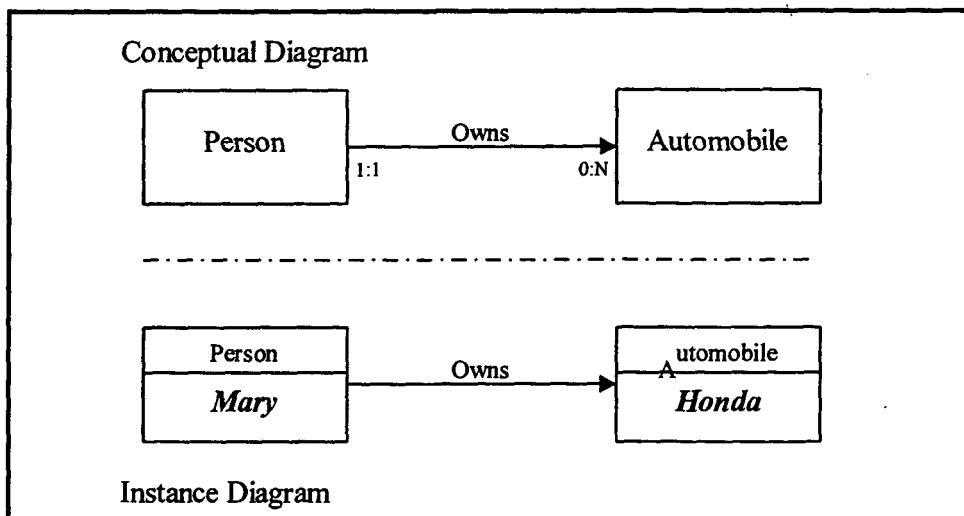


Figure 4
Conceptual Diagram and an Instance Diagram

3.2.4 AttributableMetaObject Hierarchy

The AttributableMetaObject Hierarchy in the Subject Area Overview section contains all the meta-entities and meta-relationships used in the subject area. It also identifies all supertypes required to relate the meta-objects used back to *RootObject*. Other subtypes of objects shown in the hierarchy, where neither they nor any of their subtypes are used in this subject area, are not shown.

The AttributableMetaObject Hierarchy is presented as an indented list of meta-object names. The indentation indicates the subtyping; all names indented below a given name are subtypes of that meta-object. For example, if *Car* and *Truck* are both subtypes of *Vehicle*, this would be shown as follows:

```
Vehicle
  Car
  Truck
```

A meta-entity or meta-relationship and all its subtypes can be duplicated in the hierarchy due to multiple inheritance. Where this occurs, the duplicate entries are shown in *italics*.

The overall AttributableMetaObject supertype, *RootObject*, defined in *EIA/IS-111 CDIF - Integrated Meta-model - Foundation Subject Area*, is shown at the top of the hierarchy.

Where a page break occurs in the list, all the supertypes of the first meta-entity or meta-relationship on the new page are repeated for clarity with a shaded background.

3.2.5 MetaEntity and MetaRelationship Summaries

Summaries are given of all the meta-entities and meta-relationships used in the Subject Area. The meta-entity definitions are given first, in alphabetical order, followed by the meta-relationship definitions, also in alphabetical order.

For each meta-entity used in the subject area, all the meta-attributes, both inherited and local, are listed. The inherited meta-attributes are shown first, in *italics*, sequenced alphabetically within parent meta-entity. The local meta-attributes are then listed alphabetically in normal font. The optionality, whether mandatory or optional, of each meta-attribute is given.

For each meta-relationship all the meta-attributes are listed sequenced alphabetically. Inherited and local meta-attributes are shown in the same way as for meta-entities. The optionality of each meta-attribute is given.

3.3 Structure of Subject Area Detailed Definitions

3.3.1 Summary

The Subject Area Detailed Definitions section consists of the following sub-sections:

- Subject Area Definition
- MetaEntity Definitions
- MetaRelationship Definitions

3.3.2 Subject Area Definition

This is the formal definition of the instance of the meta-entity called *SubjectArea* (SA). It contains the information listed in Table 1.

Table 1
Contents of Subject Area Definition

5.2 Subject Area Definition
NAME subject area name
CDIFMETAIDENTIFIER
DESCRIPTION
USAGE
ALIASES
CONSTRAINTS
VERSIONNUMBER

3.3.3 MetaEntity Definitions

The definition of each meta-entity used in the subject area is structured according to the classes of information listed in Table 2.

For each meta-entity:

- Inherited meta-attributes are listed by supertype meta-entity in a top-down hierarchical order. Within each sub-list, meta-attributes are listed alphabetically.

Table 2
Contents of Meta-entity Definition

5.N.N	Meta-entity Name
NAME.....	meta-entity name
CDIFMETAIDENTIFIER.....	
SUBTYPEOF	List of direct supertypes of the meta-entity meta-entity name [not] in this SA
SUPERTYPEOF.....	List of direct subtypes of the meta-entity within this subject area... meta-entity name [not] in this SA
DESCRIPTION	
USAGE	
ALIASES.....	
CONSTRAINTS	
TYPE	
INHERITED META-ATTRIBUTES	
	List of meta-attributes inherited from all supertypes of the meta-entity, with the name of the meta-object for which the meta-attribute is defined...
	meta-attribute name from meta-entity name [not] in this SA
LOCAL META-ATTRIBUTES	
	List of meta-attributes defined for the meta-entity...
	meta-attribute name
INHERITED META-RELATIONSHIPS	
	List of instantiable meta-relationships inherited from all supertypes of the meta-entity...
	meta-relationship name [not] in this SA
LOCAL META-RELATIONSHIPS	
	List of meta-relationships defined for the meta-entity...
	meta-relationship name
Full definitions of all local meta-attributes appear at this point. See Table 3 for format of local meta-attribute definitions.	

- Inherited meta-relationships are grouped by source meta-entity (listed in alphabetical order). Within each group, the inherited meta-relationships are sorted alphabetically by "verb" name and then alphabetically by destination meta-entity name. Inherited abstract meta-relationships are not listed. Local meta-attributes are listed alphabetically.
- Detailed local meta-attribute descriptions are listed alphabetically. These definitions are structured according to the classes of information listed in Table 3

Table 3
Contents of Local Meta-attribute Definition

META-ATTRIBUTE NAME.....meta-attribute name
CDIFMETAIDENTIFIER.....
DESCRIPTION.....
USAGE.....
ALIASES
CONSTRAINTS.....
DATA TYPE
DOMAIN
LENGTH.....
ISOPTIONAL.....

3.3.4 MetaRelationship Definitions

The definition of each meta-relationship used in the subject area is structured according to the classes of information listed in Table 4.

For each meta-relationship:

- Inherited meta-attributes are listed by supertype meta-relationship in a top-down hierarchical order. Within each sub-list, meta-attributes are listed alphabetically.
- Local meta-attributes are listed alphabetically.
- Detailed local meta-attribute descriptions are listed alphabetically.

Table 4
Contents of Meta-relationship Definition

5.N.N	Full Meta-relationship Name
NAME	meta-relationship name
CDIFMETAIDENTIFIER	
SUBTYPEOF	List of direct supertypes of the meta-relationship meta-relationship name [not] in this SA
SUPERTYPEOF	List of direct subtypes of the meta-relationship within this subject area... meta-relationship name
MINSOURCECARD	Minimum source meta-entity cardinality
MAXSOURCECARD	Maximum source meta-entity cardinality
MINDESTCARD	Minimum destination meta-entity cardinality
MAXDESTCARD	Maximum destination meta-entity cardinality
DESCRIPTION	
USAGE	
ALIASES	
CONSTRAINTS	
INHERITED META-ATTRIBUTES	
	List of meta-attributes inherited from all supertypes of the meta-relationship, with the name of the meta-object for which the meta-attribute is defined...
	meta-attribute name from meta-relationship name [not] in this SA
LOCAL META-ATTRIBUTES	
	List of meta-attributes defined for the meta-relationship...
	meta-attribute name
Full definitions of all local meta-attributes appear at this point. See Table 3 for format of local meta-attribute definitions.	

3.4 Structure of Referenced Meta-object Definitions

This section contains the formal definitions of meta-objects referenced from the definitions of meta-objects used in the subject area. These will include the *AttributableMetaObject RootObject*, since this is the root of the *AttributableMetaObject* Hierarchy, and it may also include other meta-entities and meta-relationships.

The definitions are given in the following order: *AttributableMetaObject*, meta-entities, then meta-relationships. Within each section, they are listed in alphabetical order. The definition layout for the *AttributableMetaObject* is the same as that of the meta-entity; both meta-entity and meta-relationship definitions follow the same layout as that defined for the Subject Area Detailed Definitions section.

This page is intentionally blank.

Structure of Subject Area Definition

4. SUBJECT AREA OVERVIEW

4.1 Description

4.1.1 Introduction

The Data Modeling Subject Area addresses the major forms of entity-relationship-attribute modeling and logical database design. This subject area is technique independent but offers broad coverage, striving for a superset of the concepts represented in various techniques.

The following sub-sections explain the major concepts supported in this subject area. The instance diagrams appearing in section 4 use various techniques to depict relationships. Relationships are shown as lines and arrow-heads, or as diamonds. Keys and key components are shown as in Figure 5.

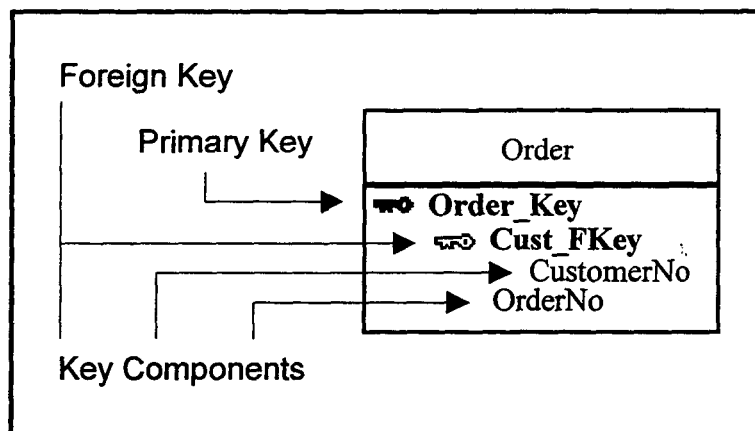


Figure 5
Diagrammatic Conventions - Keys

Solid keys are primary keys; clear keys are foreign keys and indented items are key components. For example, Figure 5 reads, "*Orders* have a primary key called *Order_Key* that consists of a foreign key called *Cust_Fkey* and an attribute *OrderNo*. *Cust_Fkey* consists of a single attribute called *CustomerNo*."

4.1.2 Entities

The concept of an "Entity," or object to be modeled, is represented directly by the meta-entity called *Entity*. A relational "Table" is also represented by *Entity*. A Table is a more concrete representation of an *Entity*.

4.1.3 Relationships

A relationship is an association between two or more *DataModelObjects*, expressed by the meta-entity *Relationship*. There are two orthogonal classifications of relationship, each with two subtypes:

- ARITY

Arity expresses the degree or number of participants in a relationship. The two types are:

- Binary
- N-ary

- INTRICACY

Intricacy categorizes the complexity of relationship association. The two types are:

- Simple
- Complex

4.1.3.1 Binary Relationship

This is an association between two *DataModelObjects*. Figure 6 depicts a binary relationship between *Orders* and *LineItems* and reads, "**Order** contains **LineItem**."

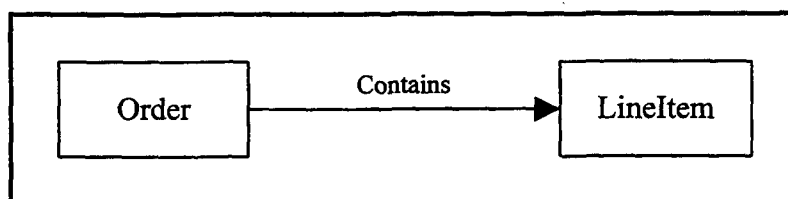


Figure 6
Binary Relationship

4.1.3.2 N-ary Relationship

This is an association between three or more *DataModelObjects*. Figure 7 depicts a ternary relationship which reads "An *Employee* is *Recruited* by an *Employer* using a *Recruiter*."

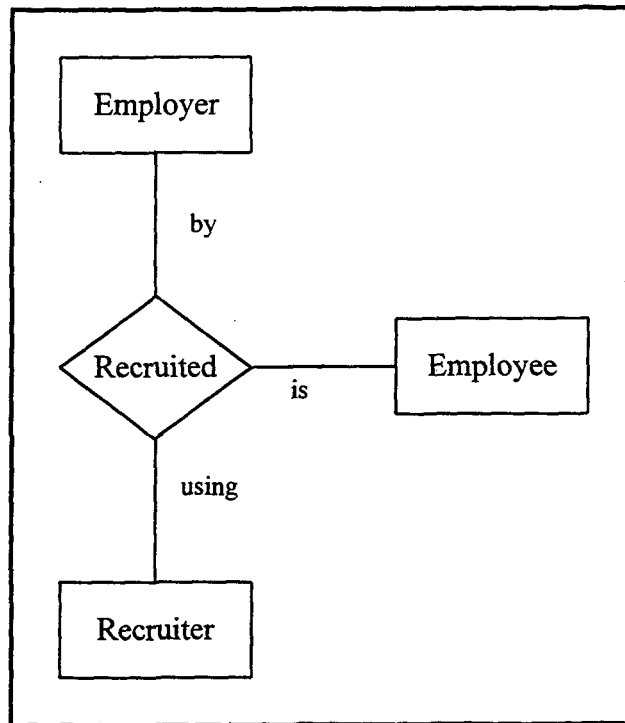


Figure 7
Ternary Relationship

4.1.3.3 Simple Relationship

This is an association between *Entities* or between *Clusters* or between *Entities* and *Clusters*. Figure 6 is an example of a Simple Binary Relationship and Figure 7 is an example of a Simple N-ary Relationship.

4.1.3.4 Complex Relationship

This is an association between a *Relationship* and one or more *DataModelObjects*. Figure 8 depicts the association of an entity called *Student* and a binary relationship called *Class* using another binary relationship called *Assigned*.

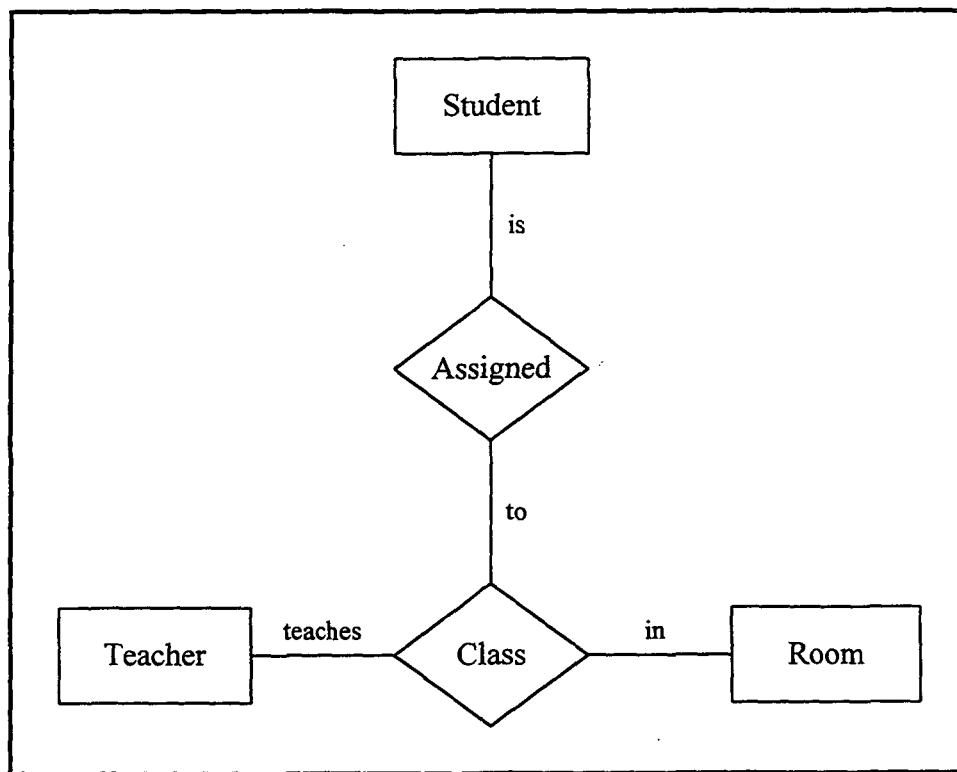


Figure 8
Complex Binary Relationship

Figure 9 is an instance diagram for Figure 8, illustrating this concept. *Captain Bligh* is a student assigned to the class taught by the teacher *Fletcher Christian* in the room named *Bounty*.

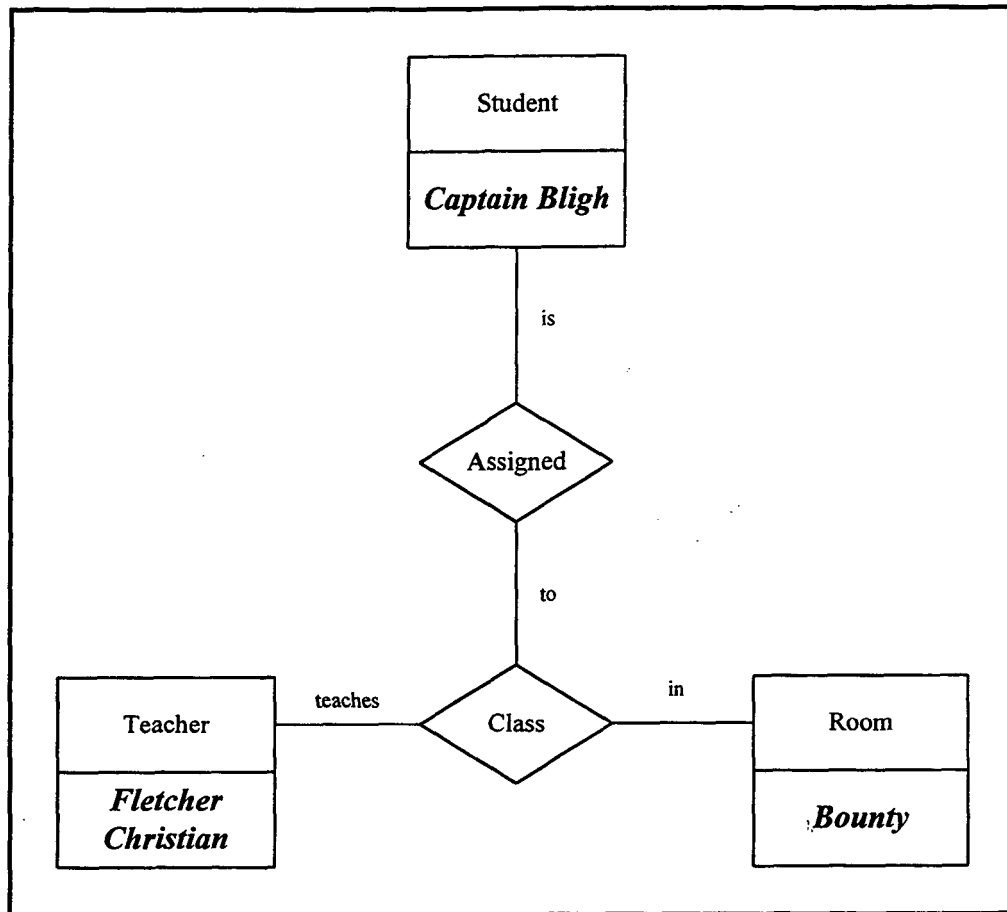


Figure 9
Instance Diagram showing a Complex Relationship

4.1.4 Roles

A "Role" is the expression of an object playing a part in a relationship. The degree or arity of the relationship is a representation of the number of roles in the relationship. In a binary relationship there are two roles (Figure 10); in a ternary relationship there are three roles (Figure 7); and in an n-ary relationship there are "n" roles. In Figure 10, **Order** plays the role of *set* and **LineItem** plays the role of *member*.

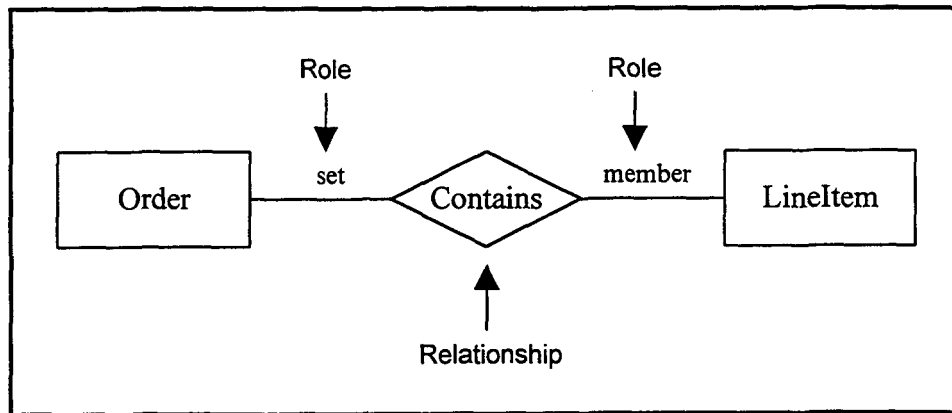


Figure 10
Roles

The object playing a role in the relationship is represented by *DataModelObject*. The *DataModelObject* can be another *Relationship*, an *Entity* or a *Cluster*. A relationship must have at least two roles to be meaningful (a binary relationship). Incomplete relationship definitions are not supported.

The same *DataModelObject* can play both (binary) or many (n-ary) *Roles* in a *Relationship* as depicted in Figure 11. This is called a "Reflexive" or "Recursive" relationship. In this example, the relationship being modeled is that of a **Person** managing another **Person**. However, each **Person** instance plays only one role with respect to another **Person** instance - either taking the role of *Manager* or that of *Subordinate*.

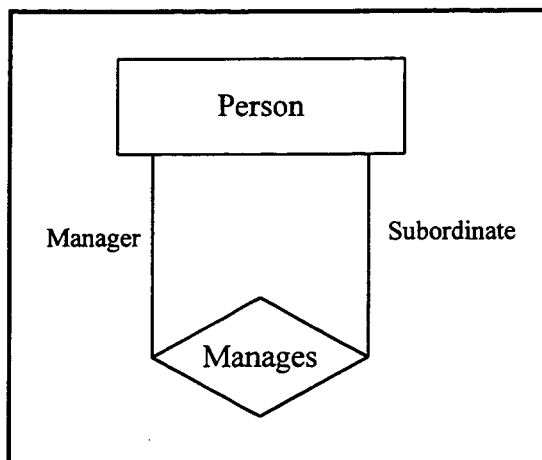


Figure 11
Reflexive or Recursive Relationship

4.1.5 Cardinalities

Cardinalities are used to define static limits on the number of *DataModelObject* instances that can be associated using a *Relationship*. The limits are generally expressed as a range in the form "m:x", where "m" expresses the minimum number of object instances, and "x" expresses the maximum number of object instances. These pairings are visually associated with the object at each end of the relationship. For example, "0:N" expresses a range where the minimum participation is zero (optional participation) to a maximum value of "N" (unlimited). The combined cardinality of all participating objects expresses the type of relationship as illustrated in Figure 12.

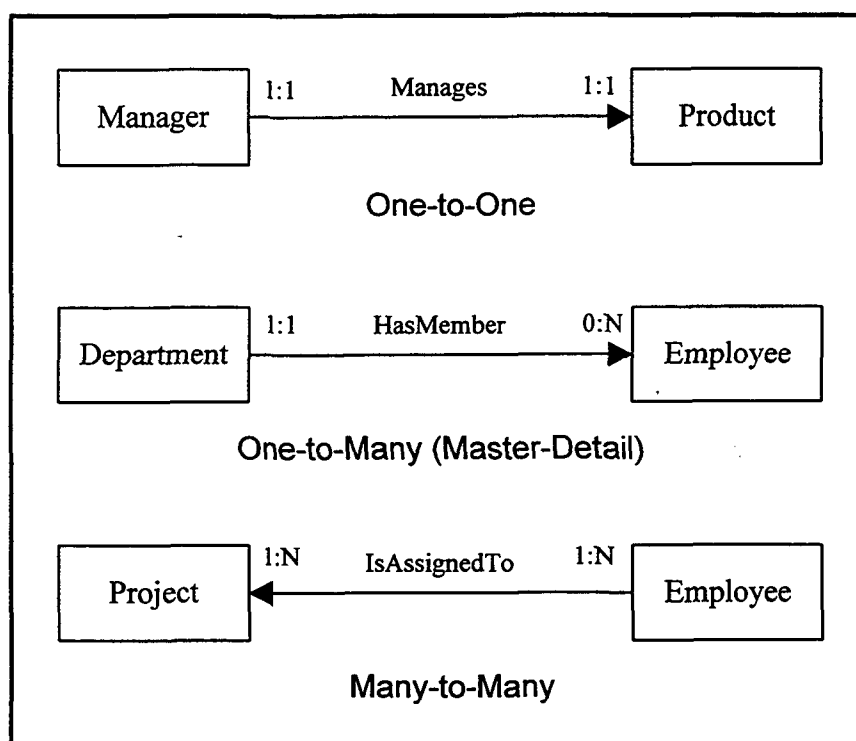


Figure 12
Examples of Types of Binary Relationships

The Data Model Subject Area differentiates between inner and outer cardinality values for *DataModelObjects* participating in relationships. These concepts are described in the next two sections.

4.1.5.1 Outer Cardinalities

The "Outer" cardinality defines the number of allowed instances of a participating *DataModelObject* from the viewpoint of the other participants in the *Relationship*. In the example in Figure 13, the outer cardinalities define the minimum and maximum number of instances of both the *Entity* called *LineItem* and the *Entity* called *Order* for the relationship called *Contains*. This example reads: "For a given *Order* there may be at minimum zero *LineItems* and at maximum an unlimited number of *LineItems* (0:N). For a given *LineItem* there may be one and only one *Order* (1:1)." It is assumed that for each *LineItem* there is a corresponding instance of the relationship *Contains*.

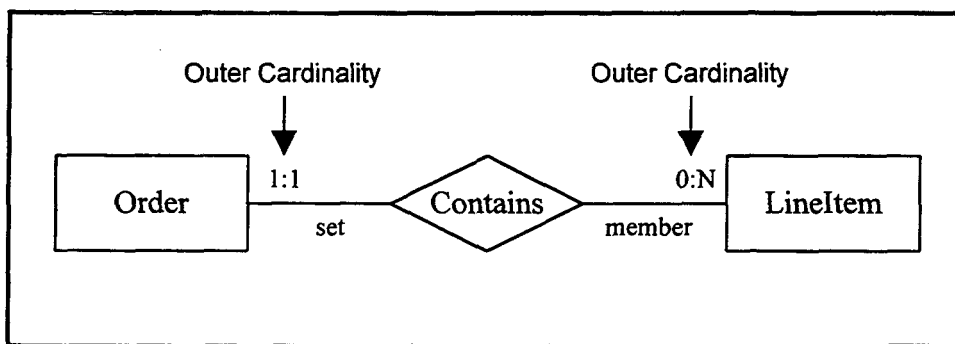


Figure 13
Outer Cardinalities

4.1.5.2 Inner Cardinalities

"Inner" cardinality defines the number of allowed instances of the *Relationship* from the viewpoint of a single instance of the *DataModelObject* playing a *Role*. In the context of unattributed binary relationships, with which most people are familiar, the issue, of how many relationship instances are allowed, does not arise. It is assumed that a single instance of the relationship exists for each pair of related *DataModelObjects*. If outer cardinalities are not provided they are assumed to be 1:1. However, in more complex modeling, when both inner and outer cardinality values are used, the concept is important. In the example in Figure 14, the model reads "One or more *Bonuses* may be given by one *Employer* to one *Employee*. We can see that each *Bonus* may be given only once and that *Employers* and *Employees* may participate in many instances of *Given*."

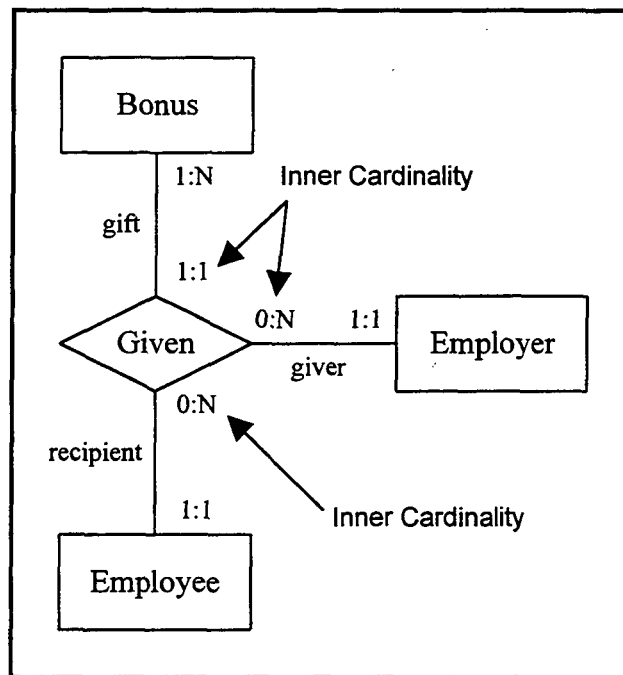


Figure 14
Inner Cardinalities

Even where attributed relationships (either binary or *n*-ary) are used, a physical implementation would provide multiple instances of the relationship. But in conceptual terms, one must be able to define whether one or many relationship instances are required for any situation, since there are situations in which each is applicable.

Figure 15 depicts the concept of relationship instances. A given relationship has either one instance or multiple instances and each has its uses. Inner cardinality specifies which case is being used. In this example we have two depictions of the relationship between an **Order** and its **LineItems**. The left hand side depicts a model that allows multiple relationship instances and the right, a single instance.

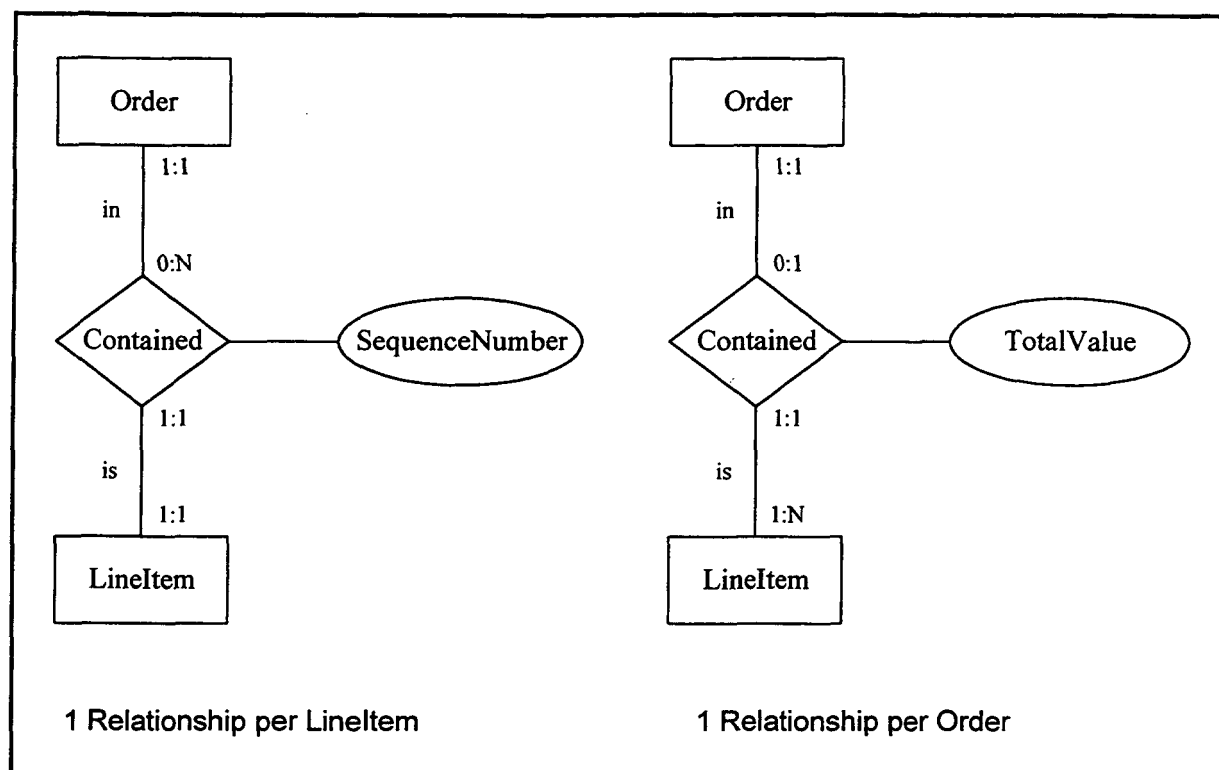


Figure 15
Inner Cardinalities and Attributed Binary Relationships

In the case of the left hand side of Figure 15, the relationship holds an attribute called **SequenceNumber** to order the **LineItems**. The inner cardinality for **Order** is **0:N** and the outer cardinality for **LineItems** of **1:1**, giving one instance of the relationship for each **LineItem**.

In the right hand side the **TotalValue** attribute on the relationship shall occur only once for the **Order**, rather than once per **LineItem** so there shall only be one instance of the relationship for each **Order**. The **TotalValue** attribute can be considered as a property of the relationship between the two entities, rather than a property of the **Order**, since its value is dependent on the **LineItem**, rather than solely on the **Order** itself. Thus the inner cardinality from the viewpoint of an **Order** instance is **0:1**, and the outer cardinality for **LineItem** is **1:N**, giving only one instance of the relationship for each **Order**, and thus only one instance of **TotalValue**.

In both cases the inner cardinality for **LineItem** is **1:1** as each **LineItem** can only relate to a single **Order**.

In the absence of inner cardinality values, the assumed semantic is multiple relationship instances. In the absence of outer cardinality values, the assumed semantic is mandatory participation of exactly one instance from each of the associated *DataModelObjects*.

4.1.6 Multi-player Roles

For each object participating in a relationship there shall be a *Role*. However, several different types of object may play the same *Role* in a *Relationship*. This is called a multi-player *Role* and it is modeled using the meta-entity *RolePlayer*. It is this mechanism that enables heterogeneous collections to be modeled. For example, in Figure 16 an *Office* is furnished with a *Desk*, *Chair*, *BookCase* and *FilingCabinet*. Note the specification of outer cardinalities on each *RolePlayer*, as each may have a different participation. There shall be only one *Desk* in an *Office* but one or more *Chairs* and zero or more *BookCases* and *FilingCabinets*.

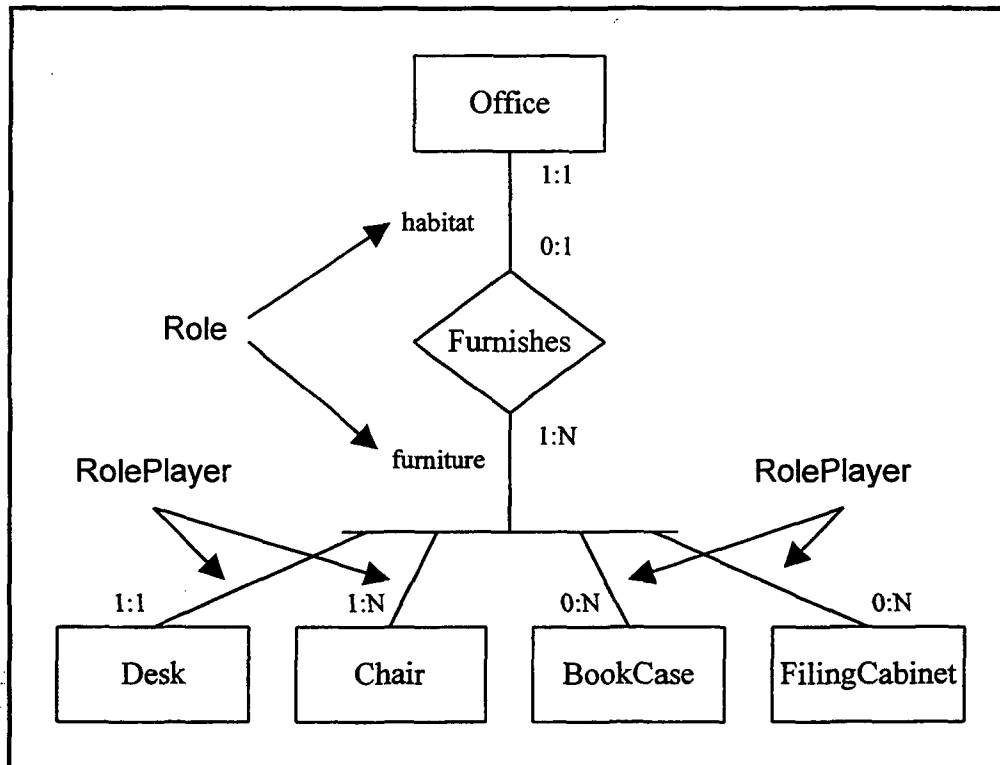


Figure 16
Multi-player Roles and Heterogeneous Collections

In contrast, in Figure 17 the *Role recruitingAgent* specifies the cardinalities of the *RolePlayers Recruiter* and *Employee*. Note that this is merely a graphical convention; cardinality values are properties of the *RolePlayer* meta-entity, and thus in this case the *RolePlayers* called *serviceProvider* and *reference* both contain the same cardinality values.

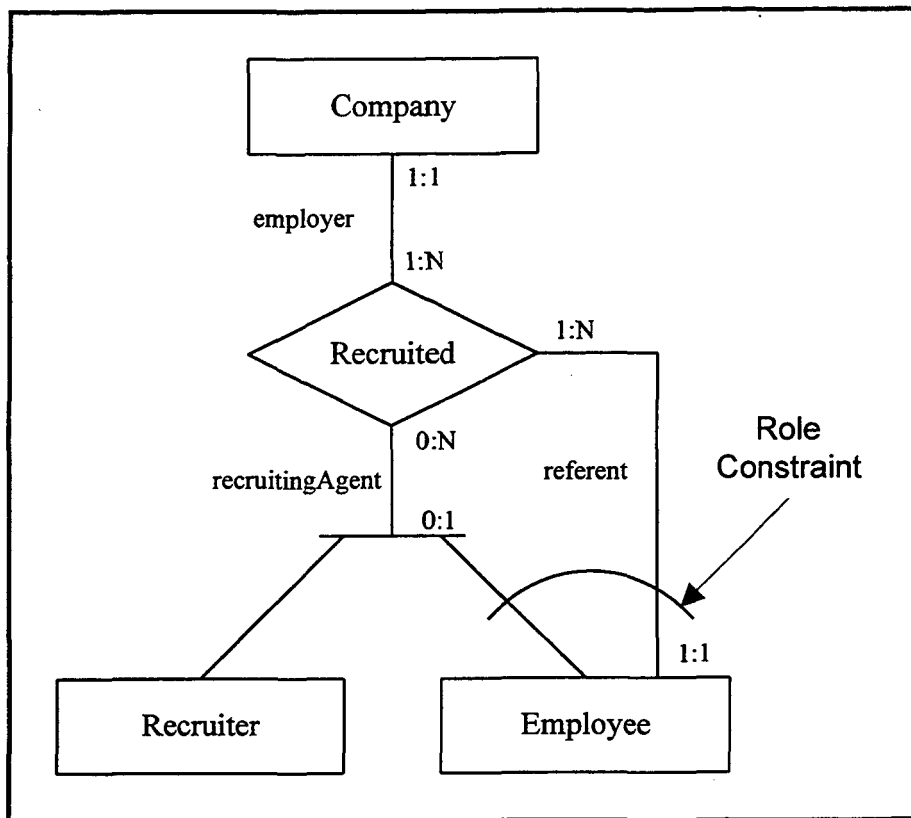


Figure 17
Multi-player Roles and Common Outer Cardinality

All relationships use *RolePlayer*, even for simple binary relationships. The *RolePlayer* meta-entity contains meta-attributes that describe cardinalities and volumetrics concerning the participating object. The volumetric meta-attributes specify the number of insertions, deletions, updates and reads over a given time period that are expected for the participating object.

4.1.7 Role Constraints

RoleConstraints provide a mechanism for expressing complex permissible groupings of relationships. This constrains the participation of an instance of a *DataModelObject* (*Entity*, *Relationship* or *Cluster*) in a set of relationships. The meta-attribute *Operator* in the meta-entity *RoleConstraint* expresses the constraint, whose values are the boolean operators inclusive **OR**, exclusive **XOR** and **AND**. The groupings are expressed on the *RolePlayers* which make up the relationships, rather than the relationships themselves, as multiple *DataModelObjects* may play a single role. The meta-relationship *RoleConstraint.Incorporates.RolePlayer* allows constraint groups to be constructed. Note that a role may participate in more than one constraint group. Figure 18 illustrates these concepts.

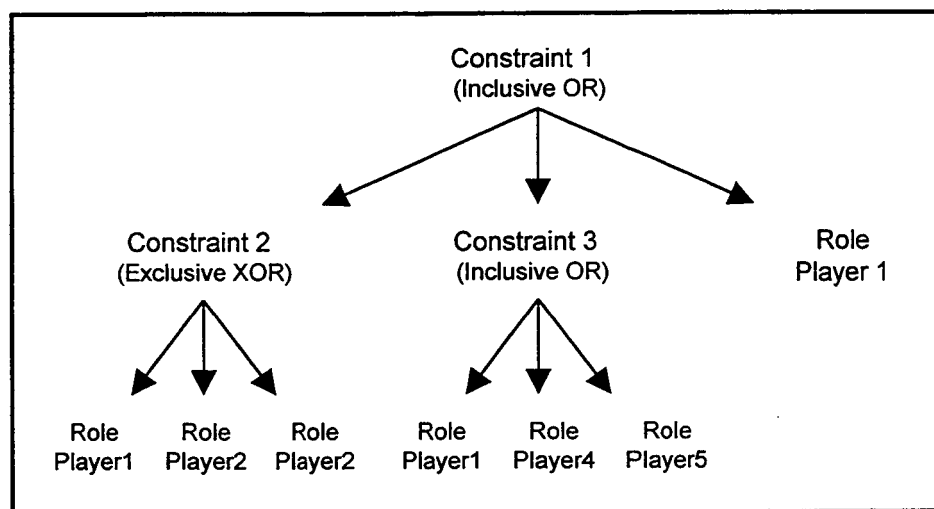


Figure 18
Role Constraint Network

A single object can act as more than one *RolePlayer* in a *Relationship* with a need to constrain the participation of its instances to only one *Role*. Figure 19 illustrates this point. In this example, **Employee** is constrained to being either a player in the role **recruitingAgent** or a player in the role **referent**. This example models employee recruitment and is read "An **Employee** participates in one or more acts of recruitment. For each recruitment, there is one **Company**, and optionally, either a single **Recruiter** or **Employee** is used. **Recruiters** and **Employees** can be used in zero or more acts of recruitment. An **Employee** is constrained from recruiting him/herself."

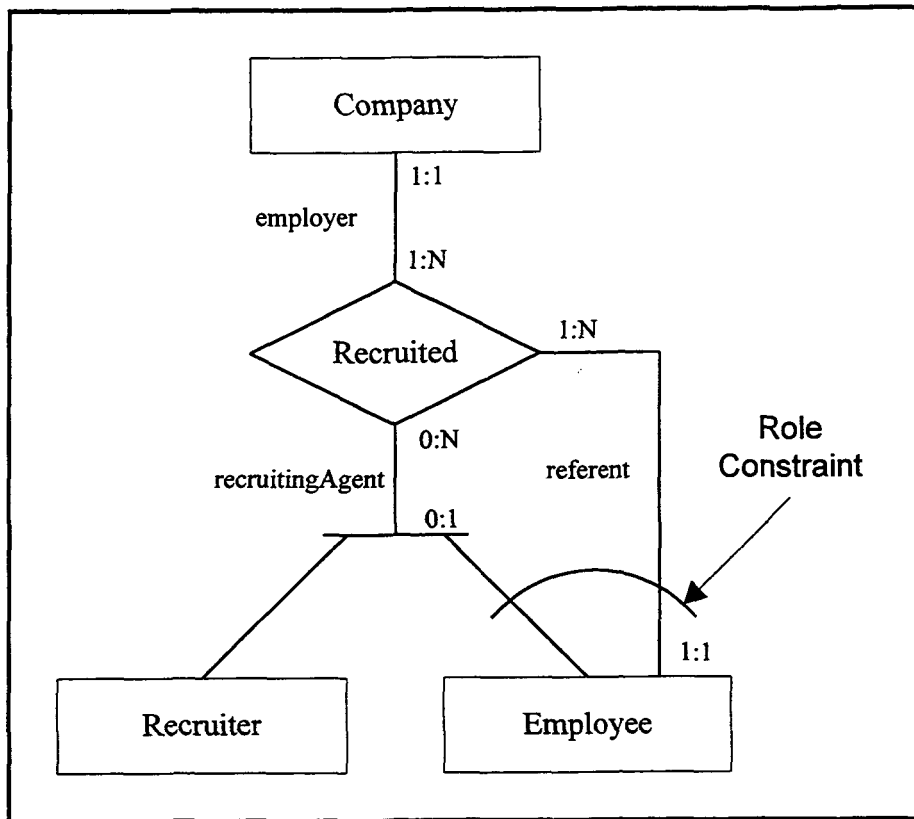


Figure 19
Constraints on Role Participation

Figure 20 is an instance diagram of Figure 19, illustrating the concept of *Role* constraints.

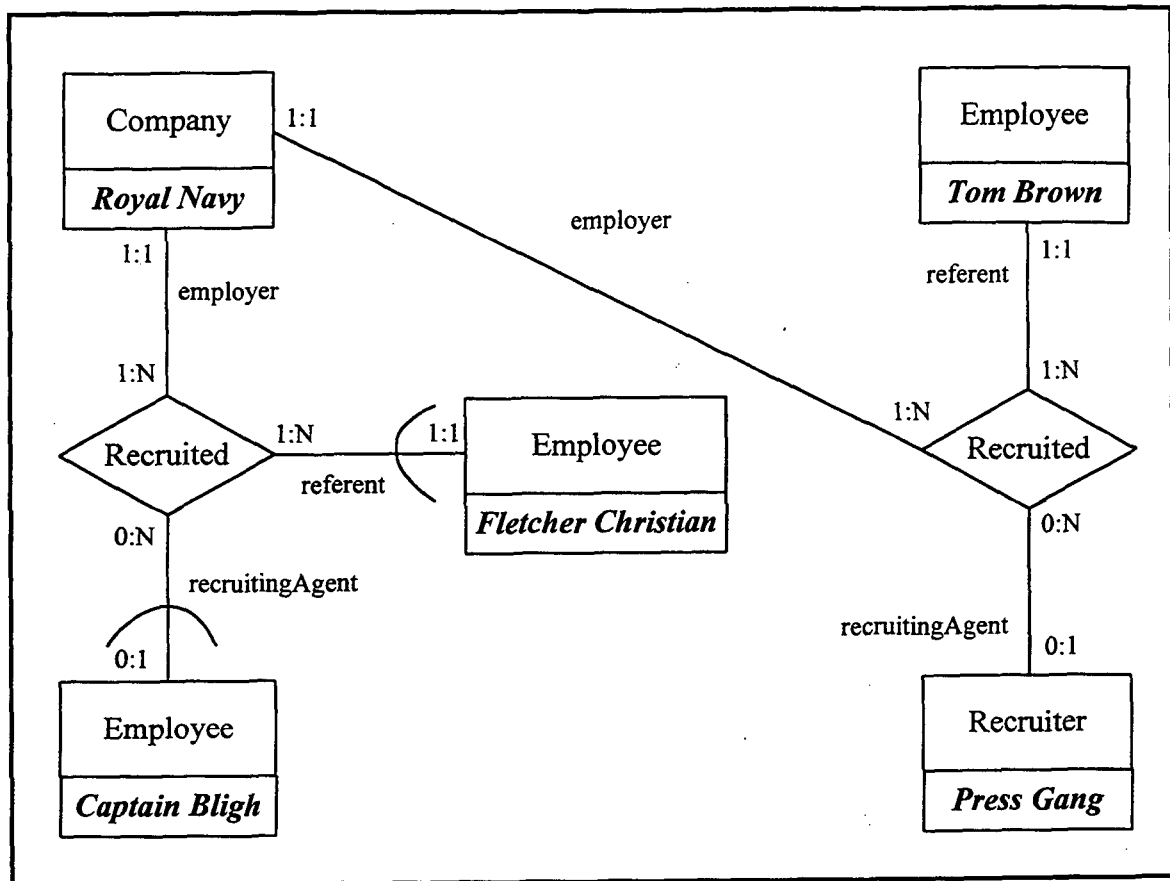


Figure 20
Instance Diagram showing Role Constraints

4.1.8 Attributes

In this subject area the data content of *DefinitionObjects* is described by the use of *Attribute* or *ProjectedAttribute* (which is a subtype of *Attribute*). These are assigned to the subtypes of *DefinitionObjects* through the meta-relationship *DefinitionObject.Contains.ComponentObject*. For example, an *Entity* called **Customer** contains attributes **Name**, **Address** and **PhoneNumber**.

ProjectedAttributes are dependent attributes that exist as a result of a projection operation such as computation. For example, the *ProjectedAttribute* called **Total** exists as a result of multiplying the *Attributes* called **Price** and **Quantity**. For a detailed discussion on projection see Section 4.1.15, Projections.

The detailed type information about attributes is not covered by this subject area; the Data Definition subject area is used to describe this. The connections between the Data Modeling subject area and the Data Definition (see *EIA/IS-113 CDIF - Integrated Meta-model - Data Definition Subject Area*) subject area is shown in Figure 46.

Roles, *RolePlayers*, and *Relationships* may also have attributes defined for them using the meta-relationship *DefinitionObject.Contains.ComponentObject*. Figure 21 illustrates the use of each of these. This models how an **Employee** is recruited by a **Company**. The *Relationship* called **Recruited** has an *Attribute* of **ApprovedBy**. Sometimes the services of a **Recruiter** are used and sometimes an existing **Employee** refers someone. In the case of the former the *RolePlayer* called **serviceProvider** is paid a **Fee** and the *RolePlayer* called **reference** is paid a **Bonus**.

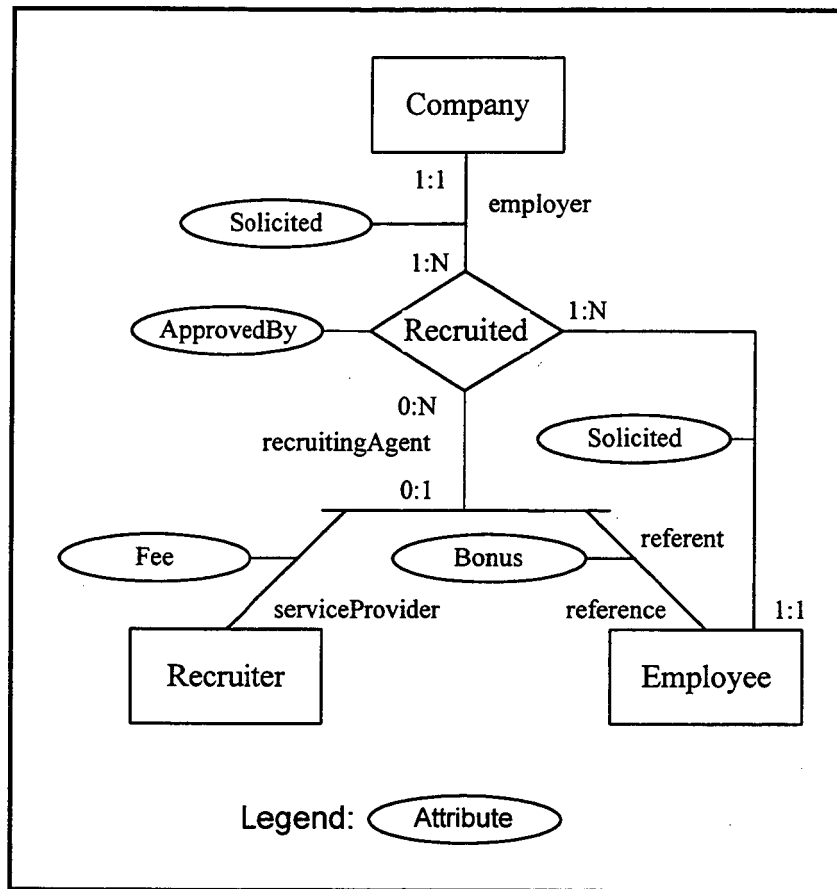


Figure 21
Attributed Relationships, Roles and RolePlayers

4.1.9 Keys and Access Paths

Entity instances are identified by *Key* values and are accessed using *AccessPath* values. A *Key* is a means of identification of *Entity* instances, providing the basis of normalization, referential integrity (see Section 4.1.10, Referential Integrity) and are used to support relationships.

AccessPaths, sometimes known as Alternate Keys, are used to specify efficient access to *Entity* instances and are often implementations of *Keys* (using the meta-relationships *Entity.IsAccessedUsing.AccessPath* and *AccessPath.Instantiates.Key*).

Keys are classified into two types:

1. Candidate Keys

CandidateKeys are a means of identification of the containing *Entity*. The concatenated values of *CandidateKey components* contain unique values and thus uniquely identify *Entity* instances. A primary key is simply one of the set of candidate keys for a given *Entity* and signifies that this is the key that is the major identifier. This concept of unique identification is the basis on which *Entity* normalization takes place. *CandidateKeys* used in conjunction with *ForeignKeys* provide the basis of referential integrity, which is the mechanism for maintaining consistency between related *Entities* after Delete, Update or Insert operations (See Section 4.1.10, Referential Integrity).

2. Foreign Keys

ForeignKeys are a replication of a *CandidateKey* from a related *Entity*. The *ForeignKey's* purpose is to provide a referential integrity constraint to ensure that any *ForeignKey* values have matching *CandidateKey* values affecting creation, modification and deletion of *Entity* instances. It is also used to support the concept of a relationship. The *ForeignKey* refers to the *CandidateKey* by using the meta-relationship called *ForeignKey.References.CandidateKey*. For example, from Figure 22, in the relationship **Customer.Places.Order** if there is a *ForeignKey* value in **Order** that does not have an equivalent *CandidateKey* value in **Customer** then there is an integrity failure.

Keys contain one or more *Attributes* drawn from the set that describes the related *Entity*. Multiple attributes comprising a key characterize the key as a concatenated key. Note that *CandidateKeys* may also contain *ForeignKeys* as in the example in Figure 22. **Order_Key** consists of both a foreign key, **Cust_FKey**, and an attribute **OrderNo**. The foreign key is **Cust_FKey** which contains a single attribute **CustomerNo**. This is modeled through the meta-relationship *CandidateKey.Incorporates.ForeignKey*. This implies that the attributes that describe the foreign keys are the same attributes (and attribute values) as those describing the associated candidate key, and thus the attributes are "global" in scope. However, *Attributes* are local in their ability to describe a *DefinitionObject* and thus *ForeignKeys* must contain *Attributes* that describe the entity that contains the *ForeignKey*. The association between a *CandidateKey's*

attributes and those of its *ForeignKeys* is accomplished by defining for each *CandidateKey* attribute a *Data Type* that is shared between the *CandidateKey* attribute and the corresponding *ForeignKey* attribute. Additionally, the sequence of attributes that comprise the *ForeignKey* must be the same sequence that comprise the *CandidateKey*.

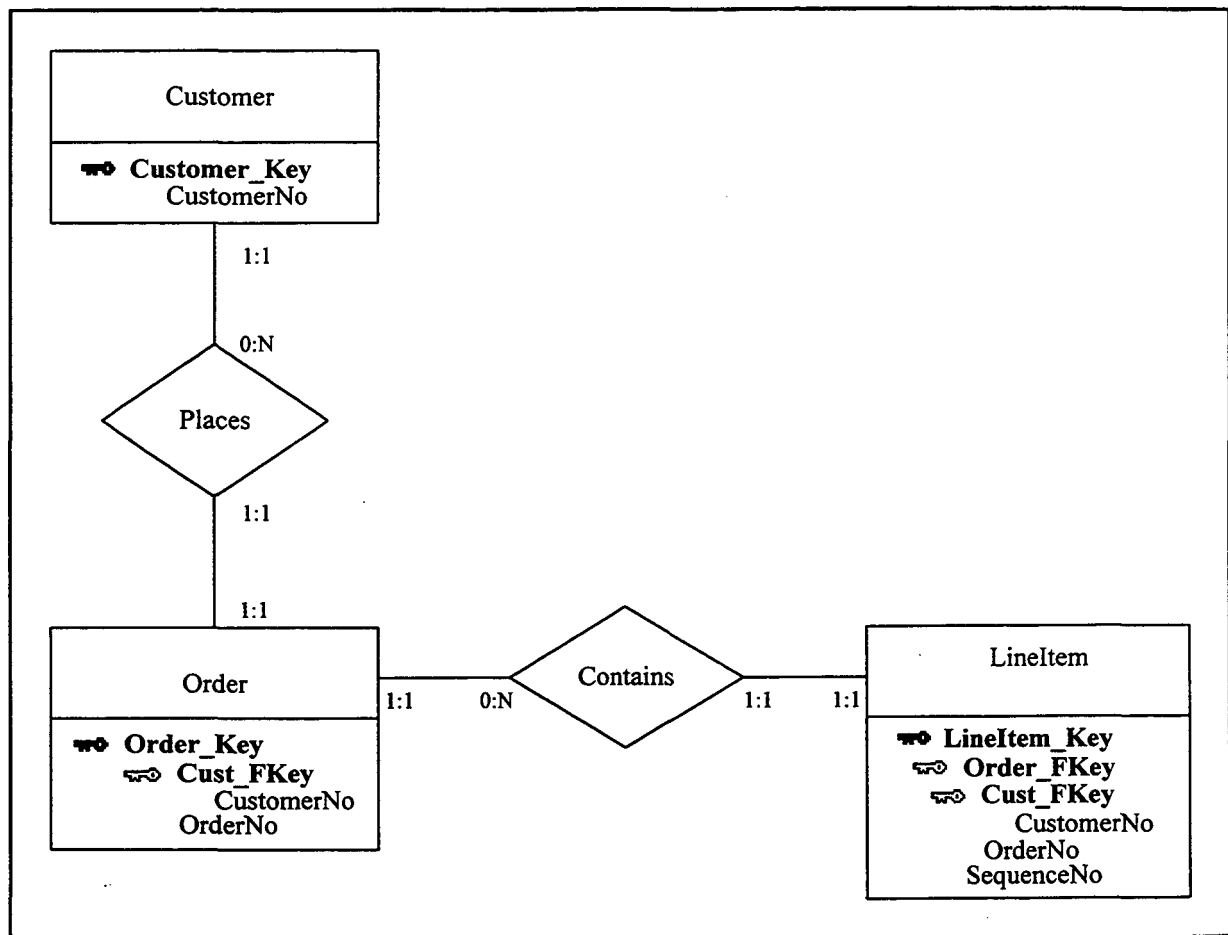


Figure 22
Foreign Keys

In some techniques a relationship can be a component of a *ForeignKey*. This is modeled by relating the relevant *RolePlayer* to the *ForeignKey* using *ForeignKey.Incorporates.RolePlayer*. In this technique attributes describing the key are not specified - they are implied from the candidate key; the foreign key is simply composed of the *RolePlayer* reference. Figure 23 illustrates this where *Employee_FKey* in *Software* is described by the *RolePlayer version* and *Employee_FKey* in *Hardware* is described by the *RolePlayer model*. This means that both *Hardware* and *Software* are not described by the attribute *Name* that is the description of the primary key in *Employee*.

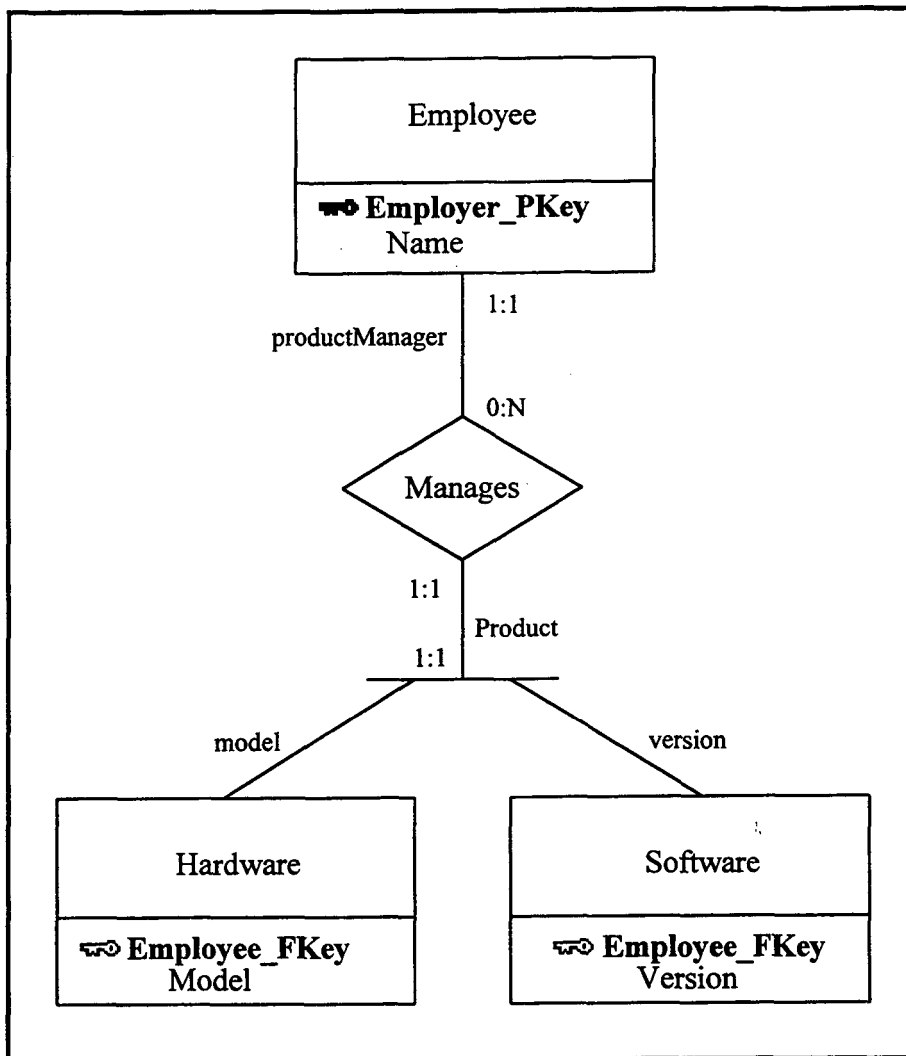


Figure 23
RolePlayer as a component of a *ForeignKey*

Pairings of *CandidateKeys* and *ForeignKeys* are used to more concretely express the *Relationships* defined between *Entities*. The meta-relationship *RolePlayer.IsSupportedBy.Key* is used as the mechanism to express this, by selecting a particular *CandidateKey* (usually the primary key) or *ForeignKey* from an entity to be used in support of the entity's participation as a *RolePlayer* in a particular *Relationship*. Where more than one *Entity* can play the same *Role* in a *Relationship*, there is one *Key* for each *RolePlayer* in the *Role*. Figure 24 illustrates this concept, where *Recruiter* and *Employee* act as *RolePlayers*, playing the role of *recruitingAgent* in the relationship. In this case the primary key *Recruiter_PKey* supports the *RolePlayer serviceProvider* and *Employee_PKey* supports the *RolePlayer reference*.

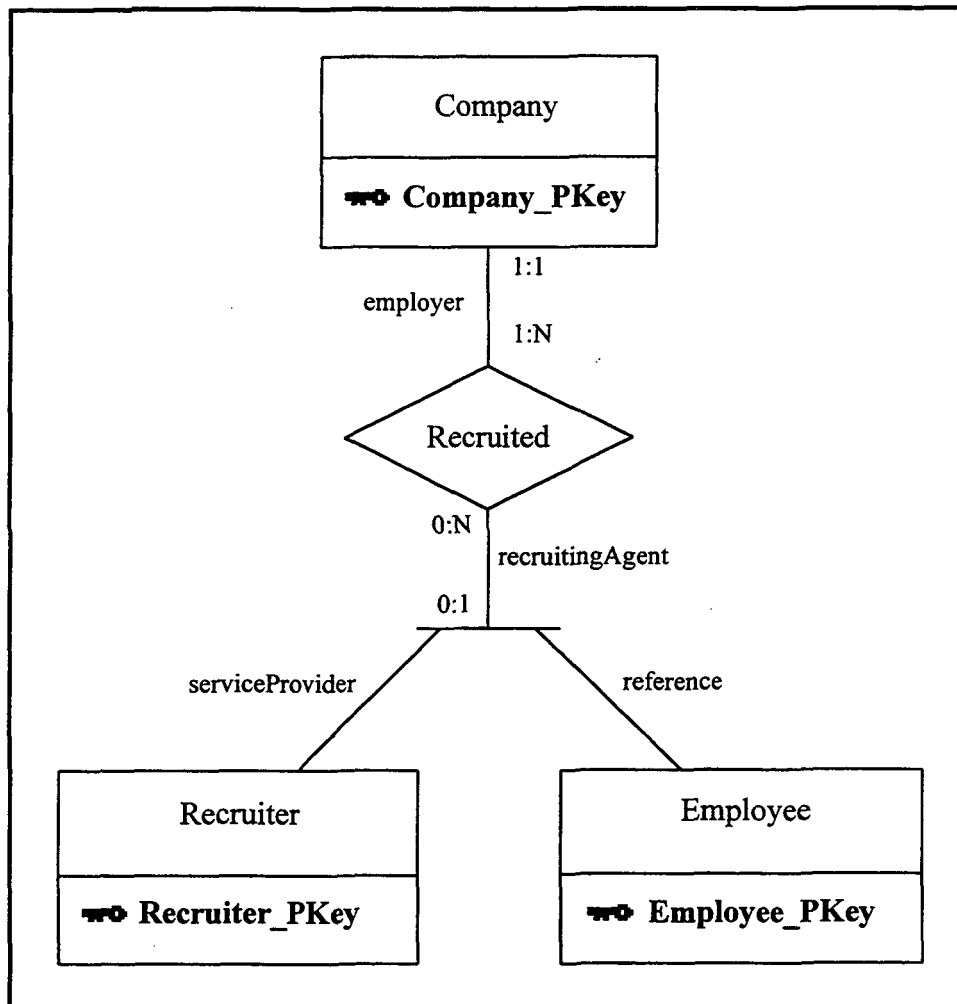


Figure 24
Key support of *RolePlayer*

The same *DataModelObject* can play both (binary) or many (n-ary) *Roles* in a reflexive or recursive *Relationship* as depicted in Figure 25. In this example, the relationship being modeled is that of a *Person* managing another *Person*. However, each *Person* instance plays only one role with respect to another *Person* instance - either taking the role of *Manager* or that of *Subordinate*. The meta-relationship *RolePlayer.IsSupportedBy.Key* identifies which role is played by which instance - the *CandidateKey Person_PKey* would play the *Manager* role and the *ForeignKey Manager_Key* would play the *Subordinate* role.

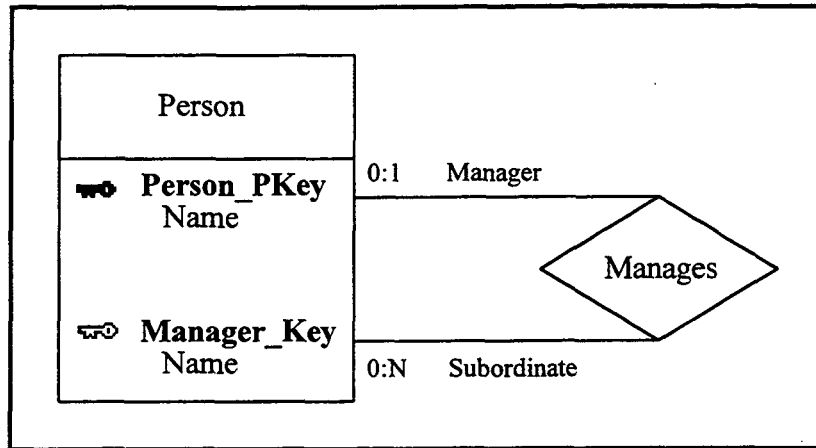


Figure 25
Key support for RolePlayer in a Reflexive Relationship

4.1.10 Referential Integrity

Keys are used to enforce referential integrity. Referential integrity rules are associated with *RolePlayer* as the mechanism for binding *CandidateKey* and *ForeignKey* pairs using the meta-relationship *RolePlayer.IsSupportedBy.Key*.

The meta-attributes *InsertEffect*, *DeleteEffect*, and *UpdateEffect*, belonging to *RolePlayer*, govern the behavior associated with Insert, Delete, and Update events on an *Entity* type. If the referenced *Entity* instance is inserted, deleted, or its *Key* is updated, then the rules outlined in Table 5 apply for each type of key when each event occurs:

Table 5
Referential Integrity Rules

Key	Insert	Delete	Update
Candidate	RESTRICTS CASCADES	RESTRICTS CASCADES SETNULL SETDEFAULT	RESTRICTS CASCADES SETNULL SETDEFAULT
Foreign	RESTRICTS CASCADES	RESTRICTS CASCADES	RESTRICTS CASCADES

CandidateKey InsertEffect=

- **RESTRICTS**
The insert operation cannot complete if there are no matching *ForeignKey* entity instances. This occurs where the *ForeignKey* entity's participation in the relationship is mandatory. For example, *Employee.Has.WorkSpace (0:1 to 1:1)*, where an instance of *Employee* is inserted.
- **CASCADES**
An instance of the *ForeignKey* entity shall be inserted. This is the equivalent of deferred checking of integrity. This applies to a mandatory relationship. For example, *ProductManager.IsResponsibleFor.Product (0:N to 1:N)*, where an instance of *ProductManager* is inserted.