

Australian/New Zealand Standard™

Information technology—CDIF semantic metamodel

Part 3: Data definitions



AS/NZS ISO/IEC 15476.3:2006

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT-015, Software and Systems Engineering. It was approved on behalf of the Council of Standards Australia on 27 October 2006 and on behalf of the Council of Standards New Zealand on 10 November 2006.
This Standard was published on 12 December 2006.

The following are represented on Committee IT-015:

Australian Computer Society
Australian Electrical and Electronic Manufacturers Association
Australian Society of Technical Communications
Australian Software Metrics Association
Engineers Australia/ACTS Joint Board in Software Engineering
Griffith University
National Association of Testing Authorities Australia
National ICT Australia
New Zealand Organisation for Quality
Software Quality Association, ACT
Software Quality Association, NSW
Systems Engineering Society of Australia
The University of Queensland
University of Auckland, NZ
University of South Australia
University of Technology, Sydney
Vendor Interests, NZ

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Web Shop at www.standards.com.au or Standards New Zealand web site at www.standards.co.nz and looking up the relevant Standard in the on-line catalogue.

Alternatively, both organizations publish an annual printed Catalogue with full details of all current Standards. For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia or Standards New Zealand at the address shown on the back cover.

This Standard was issued in draft form for comment as DR 06511.

Australian/New Zealand Standard™

Information technology—CDIF semantic metamodel

Part 3: Data definitions

First published as AS/NZS ISO/IEC 15476.3:2006.

COPYRIGHT

© Standards Australia/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Jointly published by Standards Australia, GPO Box 476, Sydney, NSW 2001 and Standards New Zealand, Private Bag 2439, Wellington 6020

ISBN 0 7337 7910 7

PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-015, Software and Systems Engineering.

The objective of this Standard is to provide people vendors and users of modelling tools and meta-data repositories with a definition of Data Definition Subject Area of the CDIF semantic metamodel. (This subject area contains meta-objects that are used as a basis of the data components of other subject area standards, and also meta-relationships and meta-attributes that are applicable to all data-related meta-objects).

This Standard is identical with, and has been reproduced from ISO/IEC 15476-3:2006, *Information technology—CDIF semantic metamodel—Part 3: Data definitions*.

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this International Standard’ should read ‘this Australian/New Zealand Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian/New Zealand Standard</i>	
ISO		AS	
31	Quantities and units	2900	Quantities and units
31-1	Part 1: Space and time	2900.1	Part 1: Space and time
ISO/IEC		AS/NZS ISO/IEC	
15474	Information technology—CDIF framework	15474	Information technology—CDIF framework
15474-1	Part 1: Overview	15474.1	Part 1: Overview
15474-2	Part 2: Modelling and extensibility	15474.2	Part 2: Modelling and extensibility
15476	Information technology—CDIF semantic metamodel	15476	Information technology—CDIF semantic metamodel
15476-1	Part 1: Foundation	15476.1	Part 1: Foundation
15476-2	Part 2: Common	15476.2	Part 2: Common
15476-4	Part 4: Data models	15476.4	Part 4: Data models

Only normative references in the source document that have been adopted as Australian or Australian/New Zealand Standard have been listed.

CONTENTS

Page

1	Scope	1
2	Conformance	2
2.1	General	2
2.2	Input conformance	2
2.3	Output conformance	2
2.4	Round-trip conformance	3
3	Normative references	3
4	Terms and definitions	4
4.1	From other International Standards	4
4.1.1	ISO/IEC 15474-1	4
4.1.2	ISO/IEC 13238-1	4
4.1.3	For this International Standard	4
5	Symbols (and abbreviated terms)	5
5.1	Naming, diagramming and definition conventions	5
5.2	Abbreviations	5
6	Data definition subject area overview	5
6.1	Introduction	5
6.2	Data Typing	5
6.3	The General Structuring Mechanism	5
6.3.1	Introduction	5
6.3.2	Meta-entities and Meta-relationships	5
6.3.3	DefinitionObject and ComponentObject	7
6.4	Alternate Decompositions	8
6.5	Pointers and Arrays	10
6.6	Data Types	10
6.7	Constraining Data Type and Attribute Values	11
6.8	Units for Numeric DataTypes	12
6.9	Void and Unknown Data Types	13
6.10	Computable Languages	13
6.11	Formats for Computable Values	13
6.12	Diagrams	16
7	Data definitions subject area summary	21
7.1	AttributableMetaObject classification hierarchy	21
7.2	MetaEntity summary	23
7.3	MetaRelationship summary	37
8	Data definitions subject area specification	39
8.1	Introduction	39
8.1.1	Subject area definition	39
8.2	Meta-entity definitions	39
8.2.1	AggregateDataType	39
8.2.2	ApproximateNumericType	40
8.2.3	ArrayQualifier	41
8.2.4	Attribute	42
8.2.5	BasicDataType	42
8.2.6	BinaryCodedDecimalType	43
8.2.7	BinaryType	43
8.2.8	BooleanType	44
8.2.9	BoundedArrayQualifier	44
8.2.10	CartesianComplexType	45

8.2.11	ComplexType	46
8.2.12	DataType	47
8.2.13	DateType	48
8.2.14	DayTimeIntervalType	49
8.2.15	DefinitionObject	49
8.2.16	EnumerationType	49
8.2.17	ExactNumericType	50
8.2.18	FixedDecimalType	51
8.2.19	FixedLengthBinaryType	52
8.2.20	FixedLengthStringType	54
8.2.21	IntegerType	55
8.2.22	MagnitudeType	55
8.2.23	MoneyType	56
8.2.24	NLFixedLengthStringType	57
8.2.25	NLVariableLengthStringType	58
8.2.26	NumericType	59
8.2.27	PackedDecimalType	59
8.2.28	PointerQualifier	60
8.2.29	PolarComplexType	60
8.2.30	QualifiedDataType	62
8.2.31	Qualifier	62
8.2.32	RefinedDataType	63
8.2.33	SerialType	63
8.2.34	StringType	65
8.2.35	TemporalType	66
8.2.36	TimeIntervalType	66
8.2.37	TimeStampType	67
8.2.38	TimeType	68
8.2.39	UnboundedArrayQualifier	69
8.2.40	Unit	70
8.2.41	ValueDomain	74
8.2.42	ValueDomainEnumeration	75
8.2.43	ValueDomainGroup	76
8.2.44	ValueDomainProcedure	77
8.2.45	ValueDomainRange	78
8.2.46	ValueDomainRule	80
8.2.47	VariableLengthBinaryType	81
8.2.48	VariableLengthStringType	82
8.2.49	VoidType	84
8.2.50	YearMonthIntervalType	84
8.3	Meta-relationship definitions	85
7.3.1	ArrayQualifier.HasType.DataType	85
7.3.2	DataType.TakesValueFrom.ValueDomain	85
7.3.3	NumericType.IsMeasuredIn.Unit	86
8.3.1	RootEntity.IsRelatedTo.RootEntity	86
8.3.2	QualifiedDataType.IsQualificationOf.DataType	86
8.3.3	QualifiedDataType.IsQualifiedBy.Qualifier	87
8.3.4	RefinedDataType.IsRefinementOf.DataType	87
8.3.5	ValueDomainGroup.Contains.ValueDomain	88

Table of Illustrations

Figure 1 – CDIF family of International Standards	1
Figure 2 – Part of general Structuring Mechanism for data definitions subject area	6
Figure 3 – Meta-model fragment for simple attribution with defined DataTypes	6
Figure 4 – Instance Diagram showing simple attribution with defined DataTypes	7

Figure 5 – Instance diagram of attributes sharing the same structured definition.....	8
Figure 6 – Instance diagram of alternate Data Structures	9
Figure 7 – Concepts of shared attribute values and definitions	10
Figure 8 – Usage of ValueDomainGroup and ValueDomain	12
Figure 9 – Data Definition Subject Area - Main Diagram.....	16
Figure 10	17
Figure 11 – MagnitudeType Subtypes	18
Figure 12 – NumericType Subtypes.....	19
Figure 13 – Subtypes and meta-relationships for Qualifier.....	20
Figure 14 – ValueDomainGroup and related meta-entities	20
Table of Tables	
Table 1 – Formats for Computable Values.....	13
Table 2 – Notation Used for Computable Values	15
Table 3 – Function Values for Computable Values	15

INTRODUCTION

This International Standard will assist the vendors and users of modelling tools and meta-data repositories in developing mechanisms for interchanging information. This International Standard specifies an element of a family of related standards. When used together, these International Standards specify a mechanism for transferring information between tools.

ISO/IEC 15474-1:2002, *Information technology - CDIF framework - Part 1: Overview* and ISO/IEC 15474-2, *Information technology - CDIF framework - Part 2: Modelling 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 modelling approach in CDIF. The CDIF meta-metamodel and extensibility mechanism are also defined in that document.

This International Standard explains the data definitions subject area of the CDIF semantic metamodel, which defines the primitive data types and the objects which are used for structured data. The CDIF semantic metamodel is used to ensure that the information transferred by tools communicating using CDIF is expressed with an agreed meaning.

This International 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.

This document is organized into the following Clauses:

— Clause 1 to 5 are prescribed ISO/IEC Clauses.

— Clause 6: Subject area overview:

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

— Clause 7: Subject area summary:

This Clause gives an overview of the content of this subject area.

— Clause 8: Subject area specification:

This Clause gives the formal specification of all the objects defined in the subject area, and the formal reference to those used, but not defined in the subject area.

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 semantic metamodel. It is suitable for:

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

This document, ISO/IEC 15474-1:2002, *Information technology - CDIF framework - Part 1: Overview*, and the framework document ISO/IEC 15474-2:2002, *Information technology - CDIF framework - Part 2: Modelling and extensibility*, should be read first when initially exploring CDIF and before attempting to read other documents in the CDIF family of International 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 modelling;
- Modelling (CASE) tools;
- Information repositories;
- Data dictionaries; and
- Multiple meta-layer modelling.

Information technology — CDIF semantic metamodel —

Part 3: Data definitions

1 Scope

The CDIF family of International Standards is primarily designed to be used as a description of a mechanism for transferring information between modelling 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 semantic metamodel defined for CASE also has applicability as the basis of standard definitions for use in repositories.

The International Standards which form the complete family of CDIF standards are documented in ISO/IEC 15474-1:2002, *Information technology — CDIF framework — Part 1: Overview*. These standards cover the overall framework, the transfer format and the CDIF semantic metamodel.

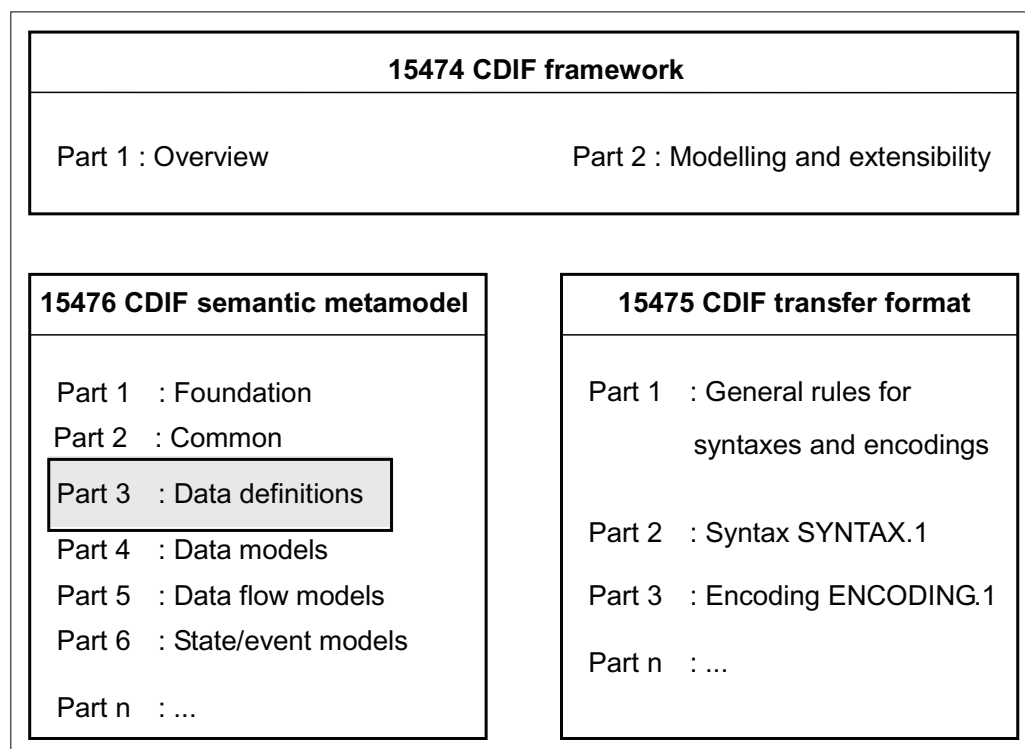


Figure 1 – CDIF family of International Standards

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

This International Standard defines the Data Definition Subject Area of the CDIF semantic metamodel. This subject area contains meta-objects that are used as the basis of the data components of other subject area standards, and also meta-relationships and meta-attributes that are applicable to all data-related meta-objects.

2 Conformance

2.1 General

A product is fully standards conformant to a CDIF subject area standard if and only if it is input-conformant, output-conformant and round-trip conformant to each and every *MetaEntity*, *MetaRelationship*, *MetaAttribute*, and *AttributableMetaObject* which is defined and/or used in that standard, and it is also CDIF architecture conformant. A product may be partially input-conformant, and/or partially output-conformant, and/or partially round-trip conformant to a CDIF subject area standard.

2.2 Input conformance

Input conformance for a specific *MetaEntity*, *MetaRelationship*, *MetaAttribute*, or *AttributableMetaObject* (short: *CollectableMetaObject*) is determined by applying the following test:

A set of meta-data containing all meanings and structures standardized by a CDIF subject area is imported by the product under test. Then the meta-data which has arrived in the product is examined. The following options exist for the relation between the input (CDIF) meta-data and the imported (product) meta-data:

For a specific *CollectableMetaObject*:

- 1 The product is input conformant if each instance of the specific *CollectableMetaObject* has arrived in the product without change of meaning or structure. If the *CollectableMetaObject* is a meta-entity or meta-relationship, its structural relationships to other *CollectableMetaObjects* have been preserved. If the *CollectableMetaObject* is a meta-attribute, the value of the meta-attribute has been preserved.
- 2 The product is input morphing conformant if each instance of the specific *CollectableMetaObject* has arrived in the product, but with some changes in meaning or structure. If the *CollectableMetaObject* is a meta-attribute, the value(s) for some instances of the meta-attribute have changed.
- 3 The product is not input conformant for that *CollectableMetaObject* if neither of the previous tests is satisfied.

2.3 Output conformance

Output conformance for a specific *CollectableMetaObject* is determined by applying the following test:

For the product being tested, a set of meta-data that includes all possible meanings and structures representable in that product is exported. Then the meta-data that has been exported is examined. The following options exist for the relation between the product's meta-data and the exported (CDIF) meta-data:

For a specific *CollectableMetaObject*:

- 1 The product is output conformant if all of the meaning and structure for the specific *CollectableMetaObject* has been represented as meta-data in the product and has been exported as one or more instances of that *CollectableMetaObject*. If the *CollectableMetaObject* is a meta-attribute, the correct value of the meta-attribute has been exported.
- 2 The product is output morphing conformant if each instance of meta-data in the product that has the same meaning and structure as the *CollectableMetaObject* has been exported, but some instances have been exported as a different *CollectableMetaObject* or some of the meaning and structure has been changed.

- 3 If the product does not represent the meaning and structure associated with the *CollectableMetaObject*, output conformance for that *CollectableMetaObject* is not applicable to the product.
- 4 In all other cases, the product is not output conformant for that *CollectableMetaObject*.

2.4 Round-trip conformance

Round-trip conformance for a specific *CollectableMetaObject* is determined by applying the following test:

A set of meta-data containing all meanings and structures standardized by a CDIF subject area is imported by the product under test. Then the meta-data is exported again. The following options exist for the relation between the input meta-data and the output meta-data:

For a specific *CollectableMetaObject*:

- 1 The product is round-trip conformant if the meaning and structure of each instance of the *CollectableMetaObject* is preserved without changes during the round-trip. For a vendor to claim round-trip conformance, it is also necessary for the tool to be able to perform create, read, update, and delete operations on the imported (product) meta-data corresponding to the instances of the *CollectableMetaObject*.
- 2 The product is round-trip morphing conformant if each instance of the input *CollectableMetaObject* is preserved, but with some changes in meaning and/or structure. If the *CollectableMetaObject* is a meta-entity or meta-relationship, some of its instances' structural relationships to other *CollectableMetaObjects* have changed, or some instances have been transformed into other *CollectableMetaObjects*, or instances of other *CollectableMetaObjects* have been transformed into instances of the *CollectableMetaObject*. If the *CollectableMetaObject* is a meta-attribute, the values of some instances of the meta-attribute have changed or the domain of the meta-attribute has changed.
- 3 In all other cases, the product is not round-trip conformant for that *CollectableMetaObject*.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 31-1:1992 *Quantities and units – Part 1: Space and time*

ISO 4217:2001, *Codes for the representation of currencies and funds*

ISO/IEC 9945-1:1996, *Information technology – Portable Operating System Interface (POSIX) –Part 1: System Application Program Interface (API) [C Language]*

ISO/IEC 13238-1, *Information technology - Data management export/import - Part 1: Standardization framework.*

ISO/IEC 15474-1, *Information technology — CDIF framework — Part 1: Overview*

ISO/IEC 15474-2, *Information technology — CDIF framework — Part 2: Modelling and extensibility*

ISO/IEC 15476-1, *Information technology — CDIF semantic metamodel — Part 1: Foundation*

ISO/IEC 15476-2, *Information technology — CDIF semantic metamodel — Part 2: Common*

ISO/IEC 15476-4 *Information technology — CDIF semantic metamodel — Part 4: Data models*

CHARACTER SETS, IANA, available at <<http://www.iana.org/assignments/character-sets>>

4 Terms and definitions

For the purposes of this document, the following definitions apply. Unless otherwise noted, the definitions are specific to this International Standard.

4.1 From other International Standards

4.1.1 ISO/IEC 15474-1

This part of ISO/IEC 15476 makes use of the following terms defined in ISO/IEC 15474-1:

CDIF
CDIF family of standards
CDIF semantic metamodel
CDIF meta-metamodel
CDIF transfer
Instance
Meta-attribute
Meta-entity
Metamodel
Meta-object
Meta-relationship
Model
Subject area
Transfer
Transfer format

4.1.2 ISO/IEC 13238-1

This part of ISO/IEC 15476 makes use of the following terms from ISO/IEC 13238-1:

Exporter
Importer

4.1.3 For this International Standard

For the purpose of this part of ISO/IEC 15476 new terms are defined when introduced. Double quotes are used to introduce new terms (e.g., "model layer")

5 Symbols (and abbreviated terms)

5.1 Naming, diagramming and definition conventions

Conventions for naming, diagramming, describing and defining meta-objects can be found in Clause 7 of the framework document (ISO/IEC 15474-2:2002, *Information technology - CDIF framework - Part 2: Modelling and extensibility*).

5.2 Abbreviations

The following abbreviation is used in this International Standard:

CDIF CASE Data Interchange Format (originally)

6 Data definition subject area overview

6.1 Introduction

The Data Definition Subject Area provides support for describing data objects and provides a data typing scheme. It allows for simple and complex structures, array and pointer qualification and domain specification. The Data Definition Subject Area diagrams are shown in Figure 9 through Figure 14.

6.2 Data Typing

A data type is described by one or more Attributes, and each of these Attributes may have a defined data type, represented by the meta-entity *DataType*. This Subject Area does not cover the internal representation of information within the basic data types; it only covers the concept of the type of information represented.

Figure 3 shows a fragment of the meta-model illustrating how data typing is represented. Figure 4 is an instance diagram of the model elements in Figure 3. Note that in the instance diagrams, values of meta-attributes (usually Name) are only given where they are required.

The data type of each Attribute is defined by relating it to the meta-entity *DataType* and its subtypes. These are described in subclauses 6.8 to 6.11. In the instance diagrams, the most appropriate subtype of *DataType* is used.

The meta-attribute Name of the meta-entity Attribute allows a local name to be defined for the attribute. If no value is given, then it is assumed that the name is the same as that of the underlying *DataType*. A default value for the Attribute can be defined using the meta-attribute *DefaultValue*, as can the optionality, using the meta-attribute *IsOptional* (The meta-attributes, *DefaultValue* and *IsOptional*, are defined in ISO/IEC 15476-4 *Data models*).

6.3 The General Structuring Mechanism

6.3.1 Introduction

Support for the decomposition and structuring of objects into other objects, and the reuse of definitions, is provided by a general structuring mechanism. This mechanism is defined in ISO/IEC 15476-2:2002, *Information Technology - CDIF Semantic Metamodel - Part 2: Common*, and is also used in other CDIF subject areas.

In this subject area *DataTypes* may be structured. Structuring of *DataTypes* is performed using the general structuring mechanism because its definition may be used for several objects in different contexts.

6.3.2 Meta-entities and Meta-relationships

The general structuring mechanism used in this subject area makes use of the following meta-entities:

- ComponentObject
- DefinitionObject

and the following meta-relationships:

- DefinitionObject.Contains.ComponentObject
- ComponentObject.References.DefinitionObject.

Figure 2 shows the part of meta-model for the general structuring mechanism related to meta-entity DataType.

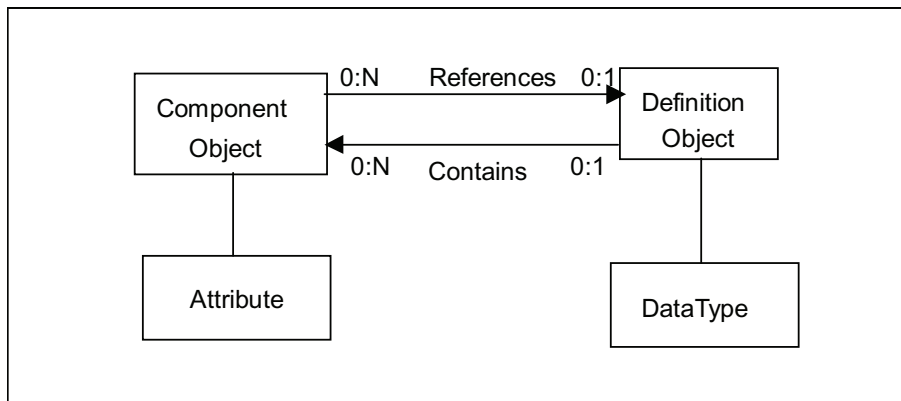


Figure 2 – Part of general Structuring Mechanism for data definitions subject area

Figure 3 and Figure 4 show how the general structuring mechanism represent a hierarchical data structure. The combination of the meta-relationship DefinitionObject.Contains.ComponentObject which is used for data structure and the meta-relationship ComponentObject.References.DefinitionObject which is used for data declaration is shown in Figure 3. The sample instance diagram is Figure 4. In this case, two data named **Customer** and **Order** are defined. Both data definition have fixed length sting type named **CustomerNumber**.

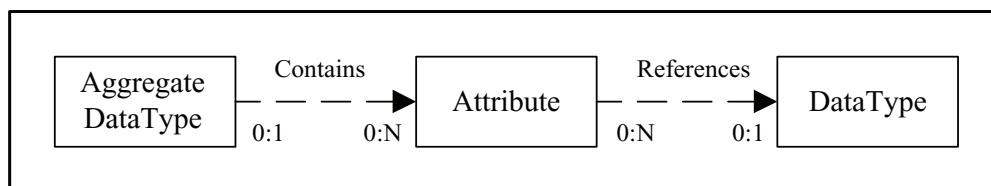


Figure 3 – Meta-model fragment for simple attribution with defined DataTypes

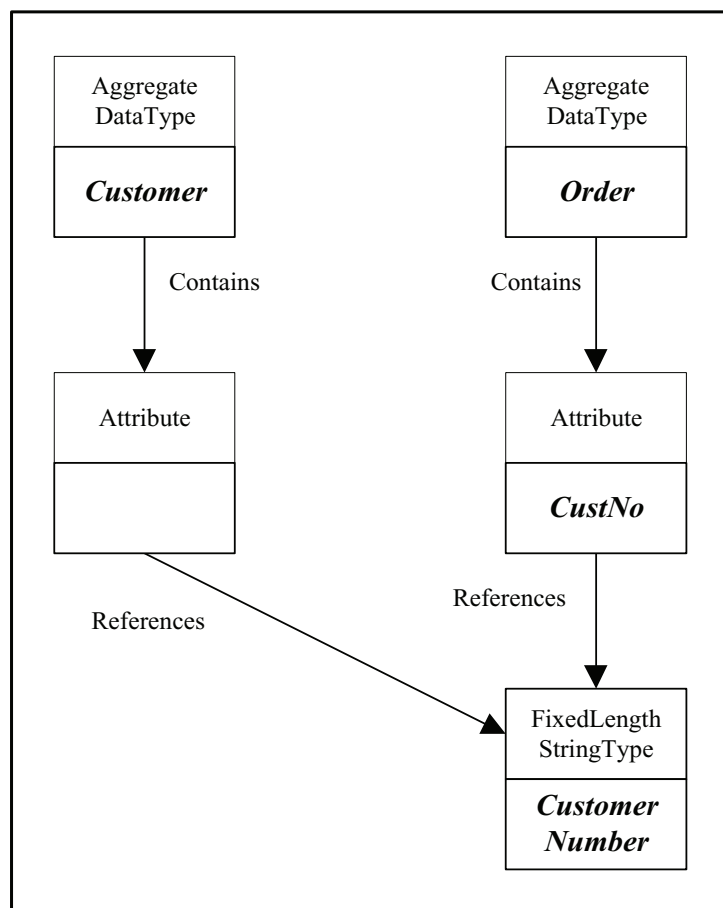


Figure 4 – Instance Diagram showing simple attribution with defined DataTypes

6.3.3 DefinitionObject and ComponentObject

DefinitionObject serves as an abstract supertype for all decomposition and structure definitions that may be reused. A *DefinitionObject* represents the definition of a *ComponentObject*, using meta-relationship *ComponentObject.References.DefinitionObject*. A *DefinitionObject* may contain *ComponentObjects*, using the *DefinitionObject.Contains.ComponentObject* meta-relationship. The *ComponentObjects* contained in a *DefinitionObject* represent the components of the definition.

Both *ComponentObject* and *DefinitionObject* represent abstract concepts and thus shall not be instantiated. Instead, appropriate subtypes, provided by this or other subject areas, are used for instantiating data definitions.

A subtype of *DefinitionObject* may be referenced by any number of *ComponentObjects* indicating that all those *ComponentObjects* share the same definition. For example, in Figure 5, *Attributes a* and *b* share the same definition, AggregateDataType **MyStruct**.

Instances of the subtypes of *ComponentObject* are used to describe the structure of a data described by a *DefinitionObject*. There may be any number instances of *ComponentObjects* per instance of *DefinitionObject*. In Figure 5, *Attributes x* and *y* comprise the structure of **MyStruct**.

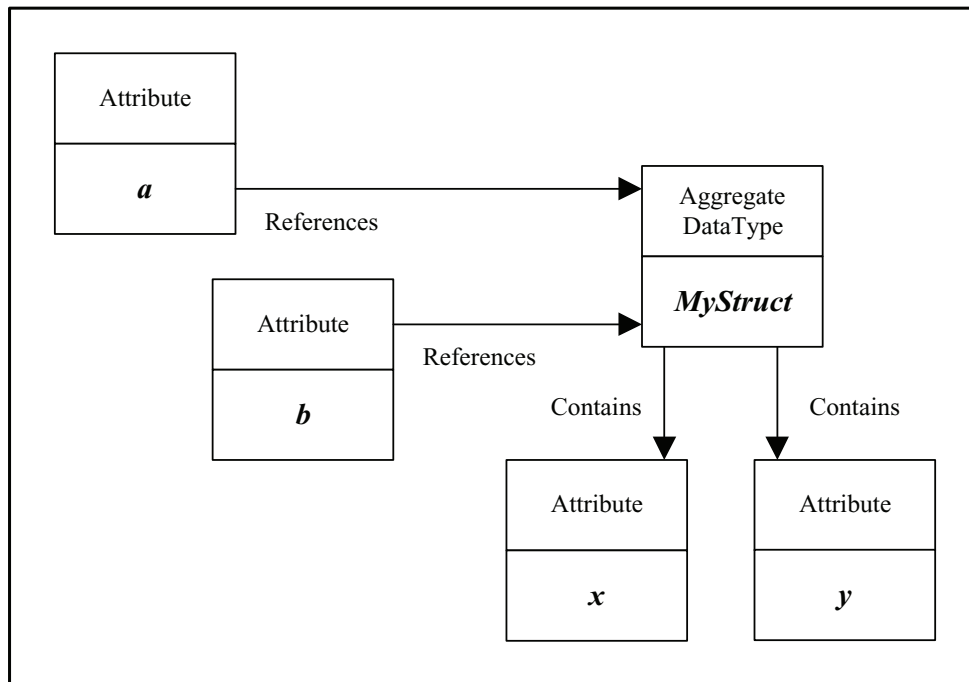


Figure 5 – Instance diagram of attributes sharing the same structured definition

For example, if a structure is built containing two components that share the same definition, the instance of structured *DefinitionObject* will contain two distinct instances of *ComponentObject* which refer to the same shared instance of the *DefinitionObject*.

The general structuring mechanism ensures that, for example, the instances of the components of one structured *Attribute* are different from those of another structured *Attribute* even if both *Attributes* share the same instance of a *DataType*.

6.4 Alternate Decompositions

This subject area supports alternate decompositions of *DataTypes* using the general structuring mechanism.

Decompositions may have one of two characteristics, represented by the meta-attribute *Operator* of the meta-entity *DefinitionObject*. *DataType* is a subtype of *DefinitionObject* and thus inherits the meta-attribute *Operator*.

If the instance value of this meta-attribute is **AND**, this indicates that all instances of the all *ComponentObjects* related to the instance of *DefinitionObject* using the instance of meta-relationship *DefinitionObject.Contains.ComponentObject* are contained at the same time (i.e. inclusive).

For example, in Figure 6, the *AggregateDataType* instance **Y1Def** representing a data structure definition would set the value of *Operator* to **AND** because it contains all the attributes concurrently.

The value **XOR** of the *Operator* meta-attribute specifies that the *ComponentObject* instances contained in the *DefinitionObject* instance are exclusive alternates. This means that the object is decomposed into exactly one of the *ComponentObject* instances but which one is not specified. An alternate decomposition leaves it open as to which of the decompositions is to be used.

For example, in Figure 6, if an *Attribute X* has two decompositions into *Attributes A, B* and *C*, or into *Attributes D* and *E*, a *AggregateDataType XDef* is created which is the definition of *Attribute X*, using the instance of meta-relationship *ComponentObject.References.DefinitionObject*. *AggregateDataType* instance **XDef** has its meta-

attribute *Operator* set to **XOR**. There are two *Attribute* instances **Y1** and **Y2** which are contained in **XDef** using meta-relationship *DefinitionObject.Contains.ComponentObject* instance, indicating that the *AggregateDataType* has two possible structures. The *Attribute* instances **Y1** and **Y2** have *AggregateDataType* instances **Y1Def** and **Y2Def**, using *ComponentObject.References.DefinitionObject* instance. Those *AggregateDataType* instances have set their meta-attribute *Operator* to **AND**, indicating that all their components are contained and they do not represent alternates. Note that if the *Operator* meta-attribute has a value of **XOR**, then the contained *Attributes* instances do not actually exist as a part of the containing *Attribute*'s structure, but are simply playing the role of a grouping mechanism. Those *AggregateDataType* instances **Y1Def** and **Y2Def** contain *Attribute* instances **A**, **B** and **C**, or **D** and **E**, respectively.

For example, the "C" programming language supports the union concept. In this case the *AggregateDataType*'s meta-attribute *Operator* would have the value **XOR**.

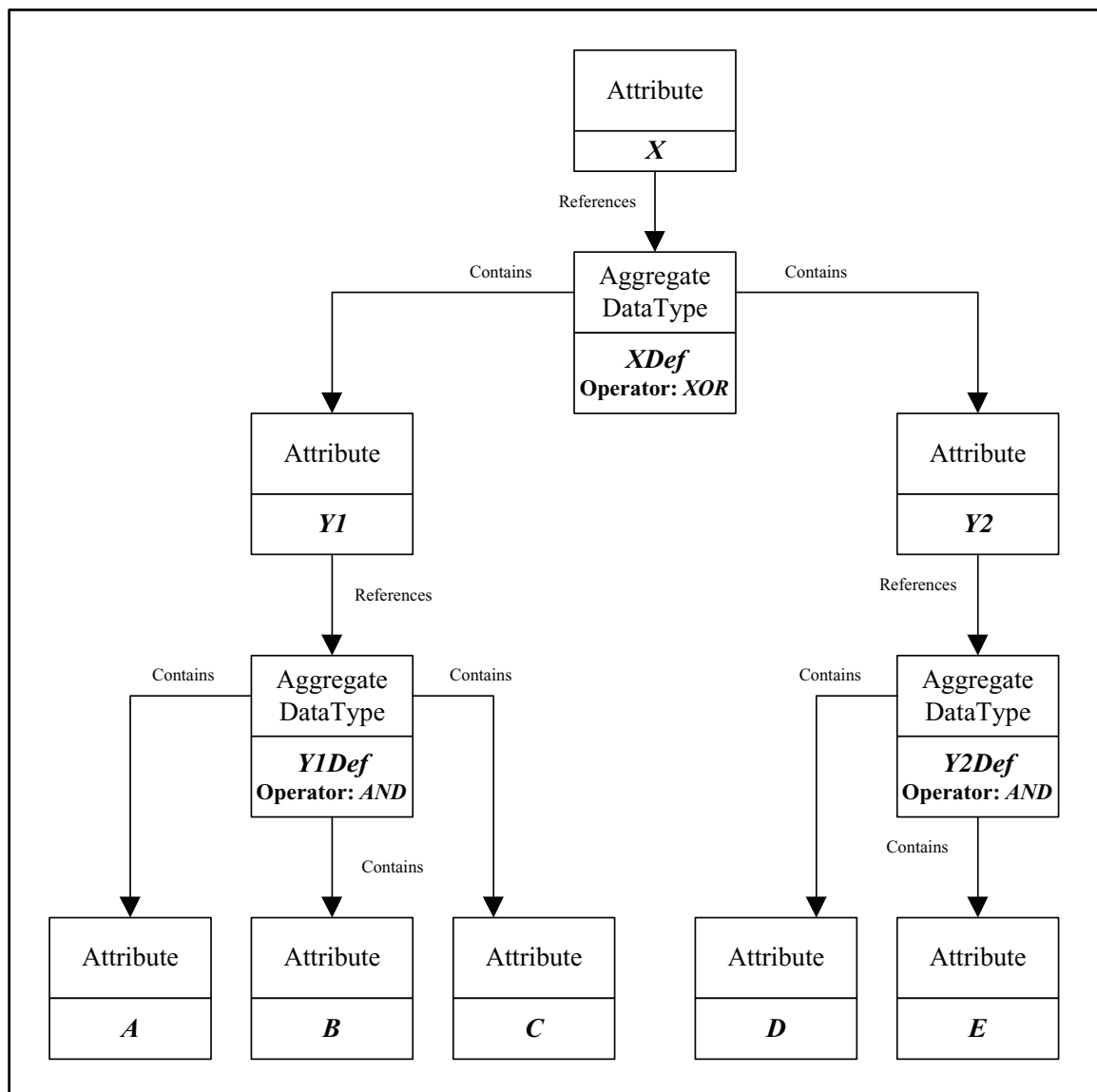


Figure 6 – Instance diagram of alternate Data Structures

The concepts of attributes, shared definitions and underlying types are illustrated in Figure 7.

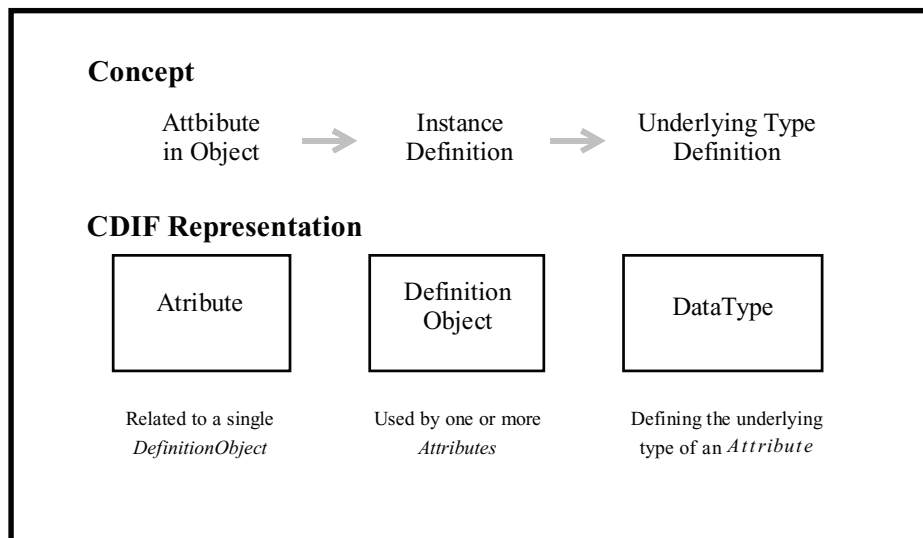


Figure 7 – Concepts of shared attribute values and definitions

6.5 Pointers and Arrays

A *DataType* can be qualified to indicate that, instead of a single occurrence of the data type itself, there is either an array of the data type or a pointer to the data type, or indeed any combination of these qualifiers. This is expressed using the meta-entities *QualifiedDataType* and *Qualifier* and its subtypes and the meta-relationship *QualifiedDataType.IsQualifiedBy.Qualifier* and *QualifiedDataType.IsQualificationOf.DataType*.

When a pointer is required, the *PointerQualifier* subtype is used. Likewise, when an array is required, the *ArrayQualifier* subtype is used. Arrays may be further specified as unbounded (via *UnboundedArrayQualifier*) or bounded (via *BoundedArrayQualifier*). The type of the *ArrayQualifier* may be expressed using the meta-relationship *ArrayQualifier.HasType.DataType*. Normally the data types will be *EnumerationType* or *IntegerType*.

More than one *Qualifier* can be defined for a single *DataType*. The meta-attribute *PrecedenceNumber* on the meta-entity *Qualifier* is used to determine the order of evaluation of the *Qualifiers*. For example a bounded two-dimensional array would have two instances of *BoundedArrayQualifier*, with the *BoundedArrayQualifier* instance expressing the first dimension having a *PrecedenceNumber* of **1**, and the second *BoundedArrayQualifier* instance having a value of **2**. In the case of unbounded one dimension array of pointers, it have **UnboundedArrayQualifier** with a *PrecedenceNumber* of **1** and a *PointerQualifier* with a *PrecedenceNumber* of **2**.

6.6 Data Types

Simple data types are represented by *BasicDataType*. Since many data types are in common use, a number of predefined data types are provided as subtypes of *BasicDataType*. These pre-defined data types are shown in Figure 9 through Figure14. They may be used directly or subtyped further, using extensibility.

6.7 Constraining Data Type and Attribute Values

ValueDomains and *ValueDomainGroups* are used to constrain the values a data type can take; the domain is the valid set of values for a data type. A *ValueDomain* can specify a single value or set of values (via *ValueDomainEnumeration*), a range of values (via *ValueDomainRange*), a procedure used to generate the valid values (via *ValueDomainProcedure*), or the format for the value (via *ValueDomainRule*).

Examples of:

A set of values for ***DaysOfTheWeek*** is (***Sunday, Monday, Tuesday*** ...).

A range is (***0..100***).

A procedure is: ***do I=1 to 30, if I modulo 3 equals 1 then return I, end.***

A format rule is ***AANNN***.

The domain of a *DataType* should be described precisely by one of the subtypes of *ValueDomain*. However, when only a textual description of the domain is required, the meta-entity *ValueDomain* itself should be used.

The domain of an *AggregateDataType* describes the interaction of the individual *ValueDomains* of the *DataTypes* in the structure. Thus, the domain of an *AggregateDataType* should be described by a *ValueDomainGroup* containing only *ValueDomainProcedures* or *ValueDomain* meta-entities.

When the domain of a data type cannot be expressed by a single domain, the *ValueDomainGroup* is used to compose complex domain expressions. *ValueDomainGroups* can connect using the *Operator* meta-attribute values ***AND, OR, NOT, or XOR***, *ValueDomains* and *ValueDomainGroups* to form complex domain expressions.

An example of the use of *ValueDomainGroup* to build up a complex domain specification is shown on Figure 8.

The values defined by a domain must be congruent with the only one underlying data type.

For refining the domain of an *Attribute*, the instance of meta-relationship *DataType.TakesValuesFrom.ValueDomain* should be declared to the *DataType* instance for the *Attribute*. In this case, the restricted domain takes precedence over the domain that related to its *DataType*.

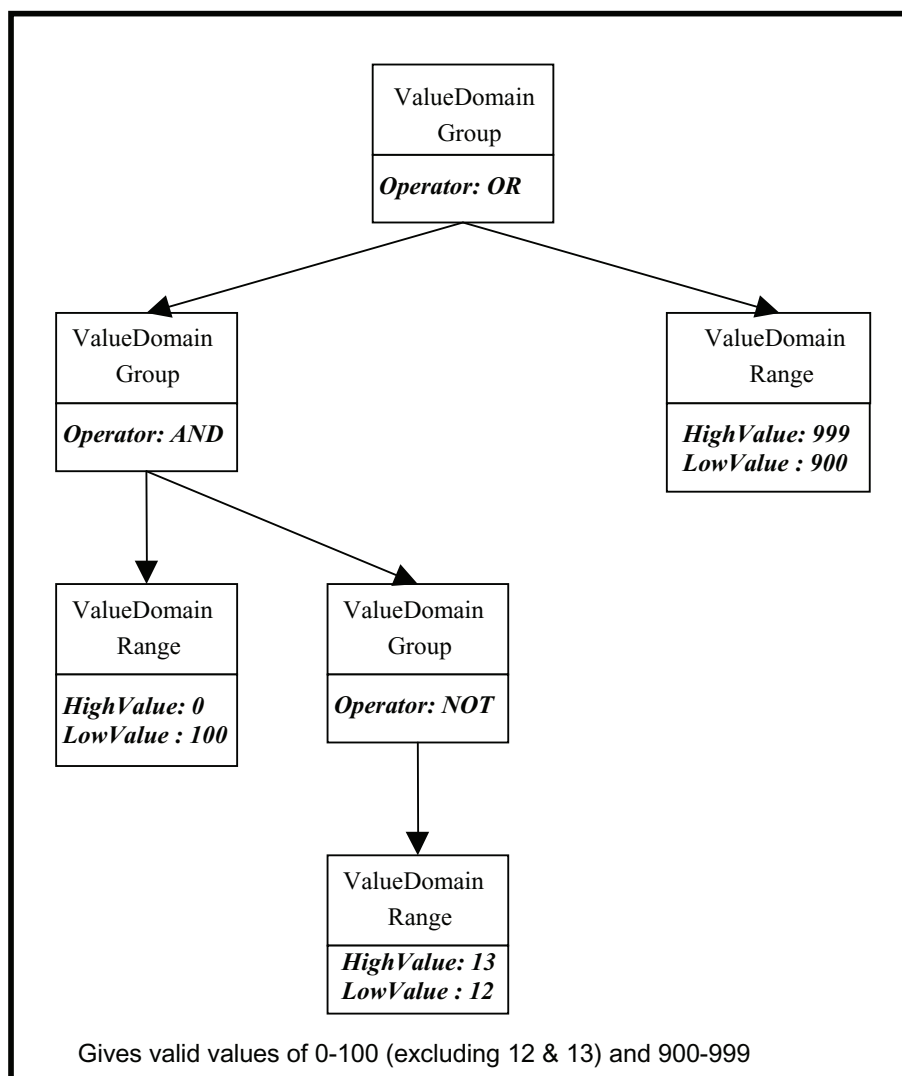


Figure 8 – Usage of ValueDomainGroup and ValueDomain

6.8 Units for Numeric DataTypes

The units of measurement for numeric data types can be expressed using the meta-entity *Unit*, related to the *NumericType* using the meta-relationship *NumericType.IsMeasuredIn.Unit*. The unit is expressed by defining the exponent for each of the SI concepts defined as a positive or negative integer.

NOTE: SI units are International System of Units defined in ISO 31-1:1992 *Quantities and units*. SI base units are meter, kilogram, second, ampere, kelvin, mole and candela.

If the unit is not actually expressed using the SI units, then the meta-attribute *IsSI* should be set to **'FALSE'** to indicate this; nevertheless the other meta-attributes should be set to indicate the concepts expressed by the unit. For example, acceleration with units of ms^{-2} ($\text{Meter}^{+1} * \text{Second}^{-2}$) should be expressed with the meta-attribute *ExponentForMeter* which express a meter unit and the meta-attribute *ExponentForSecond* which express a second unit set as follows for SI Units:

<i>ExponentForMeter</i>	1
<i>ExponentForSecond</i>	-2
<i>IsSI</i>	TRUE

If the acceleration is not expressed in SI units, then the meta-attribute *IsSI* sets to '**FALSE**' and the corresponding meta-attributes sets to the same values which expressed in SI units. For example ft per second per second, then the meta-attribute *IsSI* sets to '**FALSE**', the meta-attribute *ExponentForMeter* sets to 1, the meta-attribute *ExponentForSecond* set to -2 and the meta-attributes *name* sets to feet/(second*second).

6.9 Void and Unknown Data Types

The *VoidType* is an explicit data type to be used in certain situations where the data type cannot otherwise be defined; an example of this is where the data type is a pointer to memory - the type that it points at is not defined, or may be of different types at different times, and therefore *VoidType* is used.

Where the data type of an attribute is not known, *VoidType* must not be used. Instead, no data type should be related to the attribute concerned; the cardinality of the *ComponentObject.References.DefinitionObject* has been defined to permit the absence of a data type where it is not known.

6.10 Computable Languages

The set of computable languages that can be supported is defined in Subclause 6.12 of ISO/IEC 15476-2:2002, *Information Technology - CDIF Semantic Metamodel - Part 2: Common*.

6.11 Formats for Computable Values

There are situations where it is necessary to declare a value that will serve as a default value or domain value for an attribute or data type. This subclause defines how such values are to be expressed. The value is held in a meta-attribute of type String, delimited by single quotes. The form to be used for the value depends on its type, and is defined in Table 1. The notation used for this is explained in

Table 2.

It is also permissible for values to be derived by execution of run-time functions. Where the value is derived by a function, such as the user-id of the user making the change, or Null, the value is expressed without quotes to indicate that it is a special function rather than an explicit value. Some of the possible values for such functions are listed in 3.

Table 1 – Formats for Computable Values

DATA TYPE	Format to be used for Value	Example
Binary	as a hexadecimal string. If the resulting pattern results in too many binary digits for the underlying type or attribute, excess trailing digits are ignored.	'FF'
Boolean	TRUE or FALSE.	'TRUE'

Complex	as two Real Numbers, separated by a colon (:).	'1.125E3:0.5' '20:45'
Decimal	$[-]N(x)[.N(x)]$.	'12.25'
Enumerated	$X(x)$.	'Green'
Float	as for Real.	'123.56E11'
Integer	$[-]N(x)$.	'763'
Money	$[-]N(x)[.N(x)][AAA]$ where AAA is an optional 3-character Currency Symbol, as defined in ISO 4217 Currency Symbols.	'100.00USD'
Real	$[-]N(x)[.N(x)][E[-]N(x)]$.	'736.3145'
String	$X(x)$.	'Pair of Handcuffs'
Temporal	<p>YYYY/MM/DD ,</p> <p>YYYY/MM/DD-HH:MM[:SS[.N(x)]] or</p> <p>HH:MM[:SS[.N(x)]]</p> <p>where :</p> <p>YYYY is the year</p> <p>MM is the numeric representation of the month as 2 digits in the range 01..12</p> <p>DD is the numeric representation of the day as 2 digits in the range 01..31</p> <p>HH is the hour shown as 2 digits in the range 00..23</p> <p>MM is the minutes shown as 2 digits in the range 00..59</p> <p>SS is the seconds shown as 2 digits in the range 00..59</p> <p>N(x) are the fractional seconds to the degree of precision required.</p>	'1994/04/01-12:00'

Table 2 – Notation Used for Computable Values

Symbol	Meaning
[]	Square brackets enclose optional elements in an expression. The portion of the expression within the brackets may be explicitly specified or may be omitted.
(x)	The (x) indicates that the element that it follows in a formula may be repeated any number of times (but note that optionality is covered by the square brackets convention). If it appears immediately after a closing square bracket "]", then it applies to the portion of the expression enclosed between that closing bracket and the corresponding opening bracket "[". If it appears after any other element, then it applies only to that element.
N	Any numeric character in the range 0..9.
A	Any alphabetic character.
X	Any printable character. (See ISO/IEC 15475-3:2002, Information Technology - CDIF Transfer Format - Part 3: Encoding: ENCODING.1)

Table 3 – Function Values for Computable Values

Value	Meaning
NULL	Null value.
USER	The identity of the current user.
CURRENT_DATE	The current date.
CURRENT_TIME	The current time.
CURRENT_TIMESTAMP	The current timestamp.

6.12 Diagrams

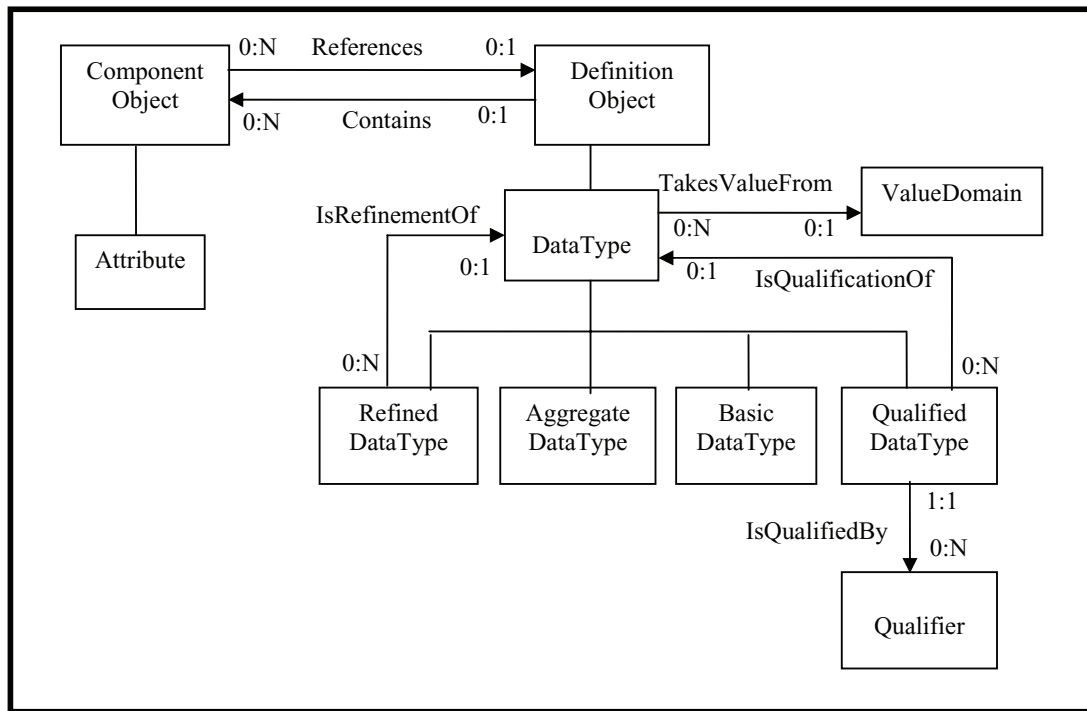


Figure 9 – Data Definition Subject Area - Main Diagram

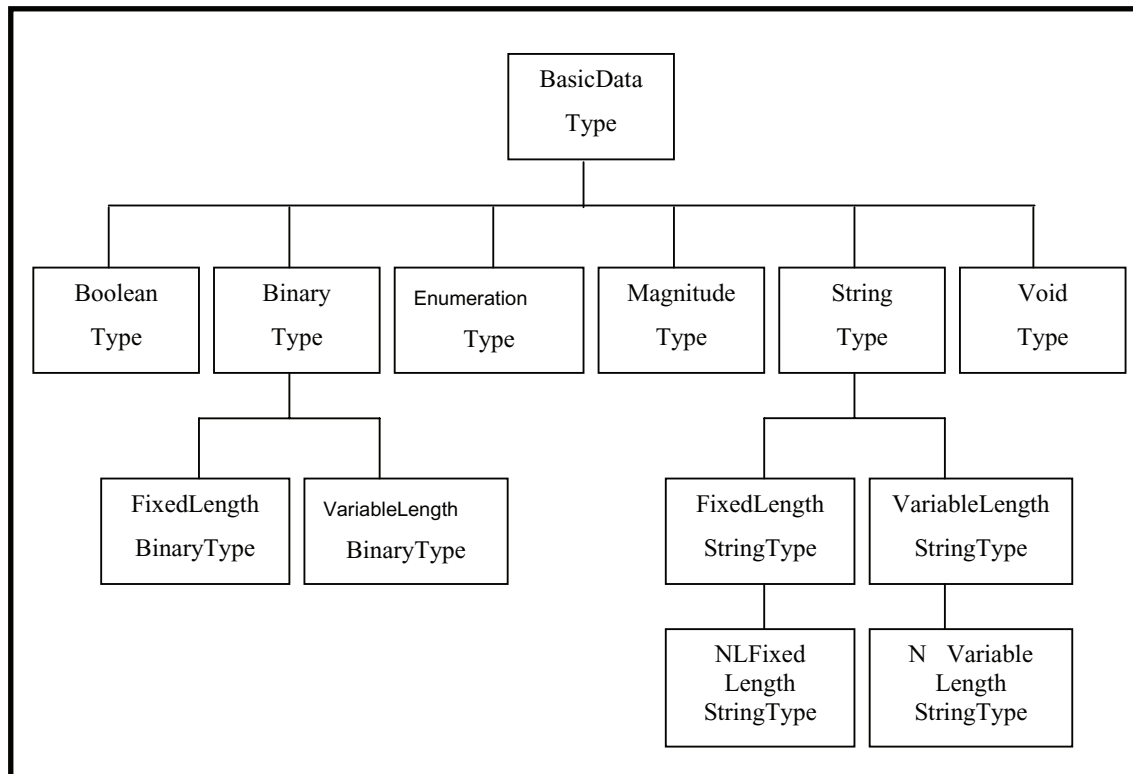


Figure 10 – BasicDataType Subtypes

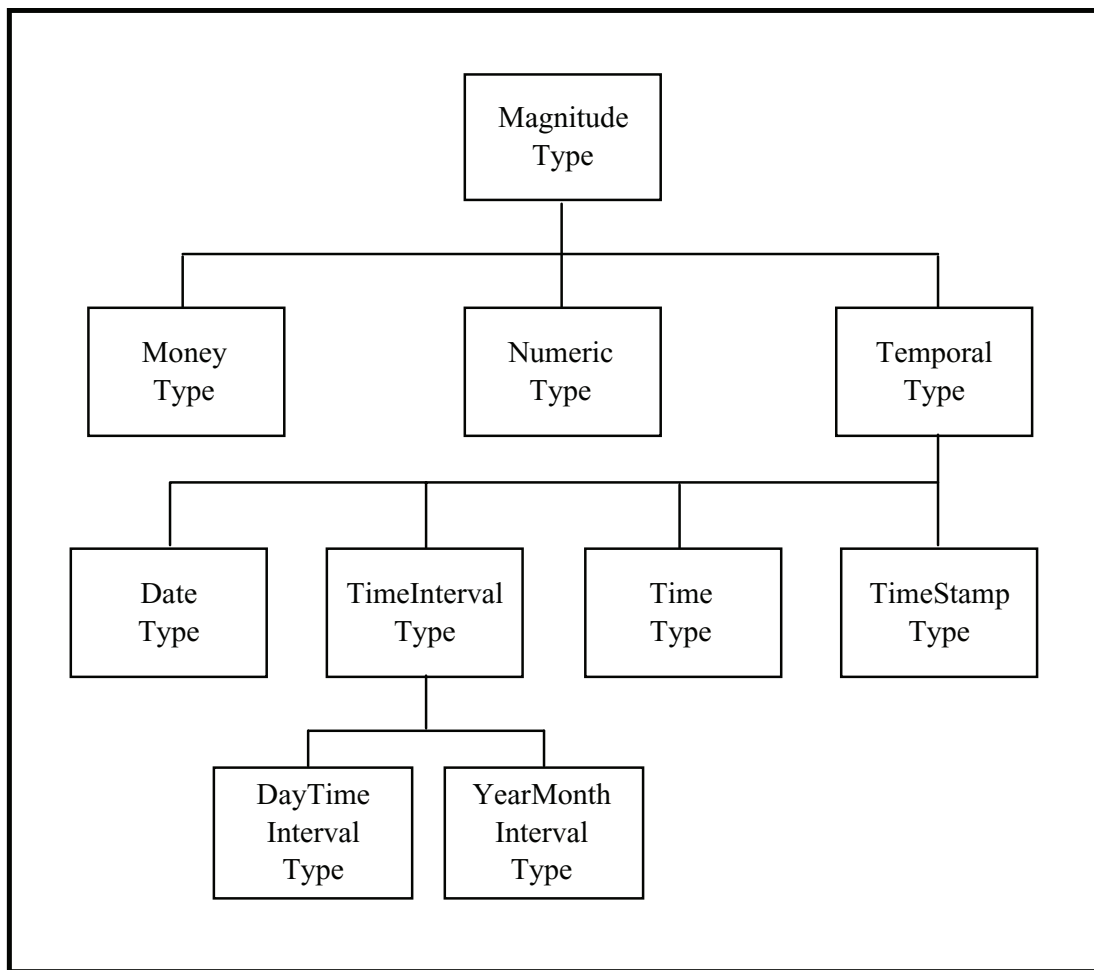


Figure 11 – MagnitudeType Subtypes

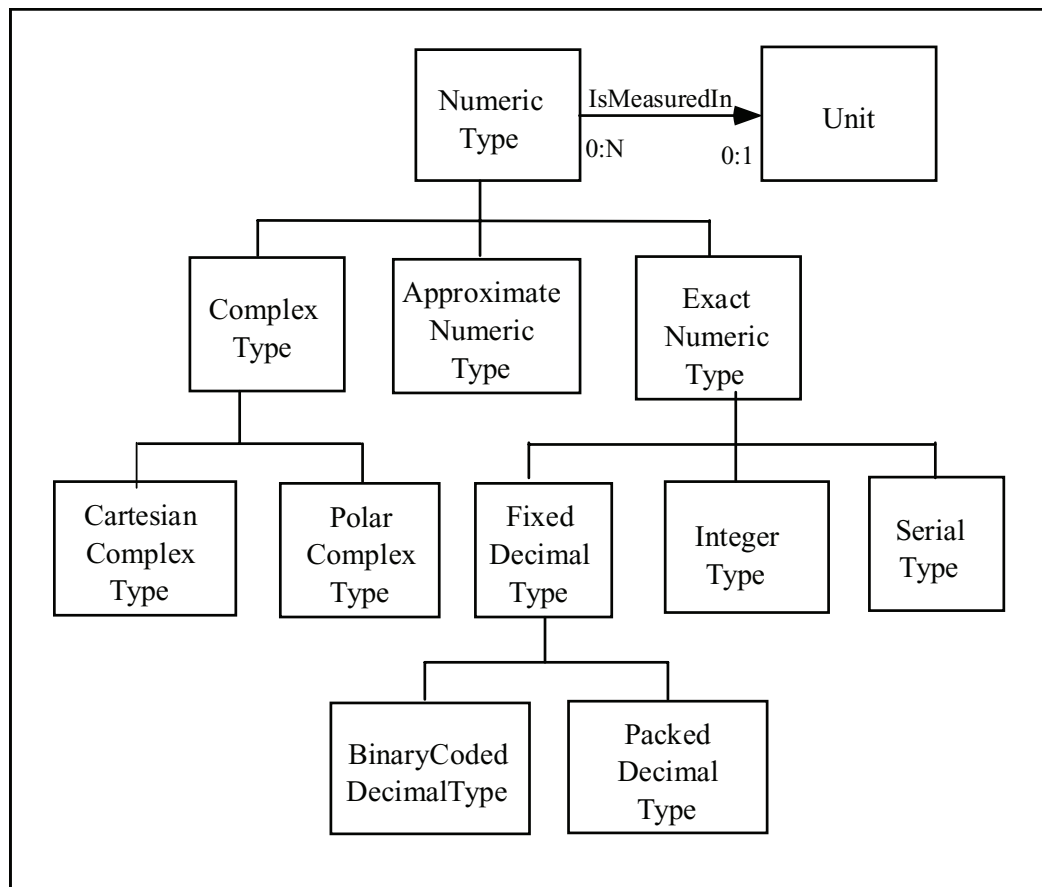


Figure 12 – NumericType Subtypes

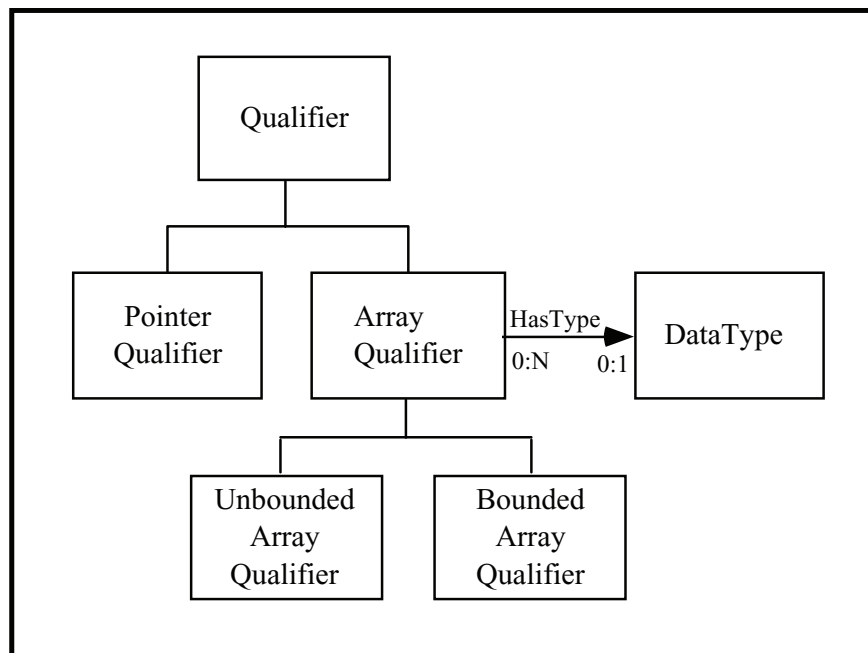


Figure 13 – Subtypes and meta-relationships for Qualifier

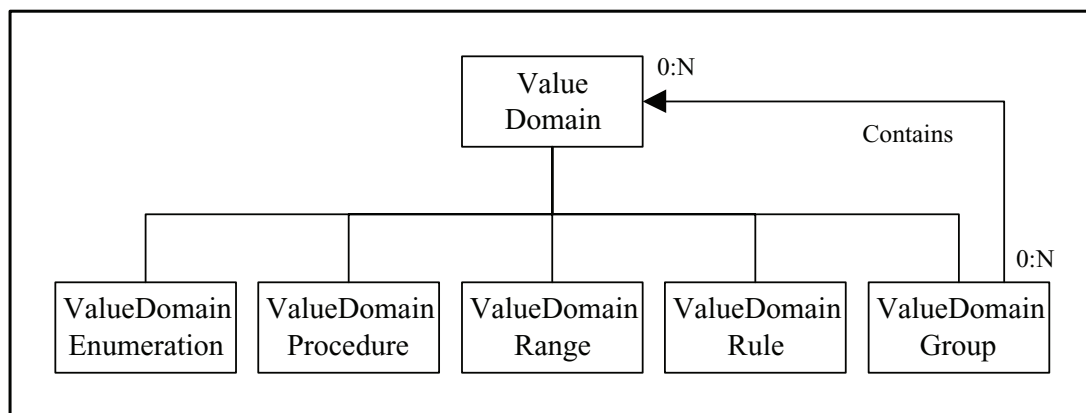


Figure 14 – ValueDomainGroup and related meta-entities

7 Data definitions subject area summary

7.1 AttributableMetaObject classification hierarchy

MetaObject Name	SubjectArea Name
RootEntity	Foundation
SemanticInformationObject	Common
ComponentObject	Common
Attribute	Data models
DefinitionObject	Common
DataType	Data definitions
AggregateDataType	Data definitions
BasicDataType	Data definitions
BinaryType	Data definitions
FixedLengthBinaryType	Data definitions
VariableLengthBinaryType	Data definitions
BooleanType	Data definitions
EnumerationType	Data definitions
MagnitudeType	Data definitions
MoneyType	Data definitions
NumericType	Data definitions
ApproximateNumericType	Data definitions
ComplexType	Data definitions
CartesianComplexType	Data definitions
PolarComplexType	Data definitions
ExactNumericType	Data definitions
IntegerType	Data definitions
FixedDecimalType	Data definitions
BinaryCodedDecimalType	Data definitions
PackedDecimalType	Data definitions
SerialType	Data definitions
TemporalType	Data definitions
DateType	Data definitions
TimeIntervalType	Data definitions
DayTimeIntervalType	Data definitions
YearMonthIntervalType	Data definitions
TimeStampType	Data definitions
TimeType	Data definitions
StringType	Data definitions
FixedLengthStringType	Data definitions
NLFixedLengthStringType	Data definitions
VariableLengthStringType	Data definitions
NLVariableLengthStringType	Data definitions
VoidType	Data definitions