INTERNATIONAL STANDARD

**IEC 61360-2**

First edition
1998-04

# Standard data element types with associated classification scheme for electric components –

## Part 2: EXPRESS Dictionary schema

*Types normalisés d'éléments de données avec plan de classification pour composants électriques –*

*Partie 2: Schéma d'un dictionnaire EXPRESS*

Reference number
IEC 61360-2:1998(E)

## Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

## Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

## Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
  Published yearly with regular updates
  (On-line catalogue)*
- **IEC Bulletin**
  Available both at the IEC web site* and as a printed periodical

## Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams.*

* See web site address on title page.

INTERNATIONAL STANDARD

# IEC 61360-2

Second edition
2002-02

**Standard data element types with associated classification scheme for electric components –**

**Part 2:**
**EXPRESS dictionary schema**

*Types normalisés d'éléments de données avec plan de classification pour composants électriques –*

*Partie 2:*
*Schéma d'un dictionnaire EXPRESS*

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http://www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XC

*For price, see current catalogue*

## CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

___________

# STANDARD DATA ELEMENT TYPES WITH ASSOCIATED CLASSIFICATION SCHEME FOR ELECTRIC COMPONENTS –

## Part 2: EXPRESS Dictionary schema

## FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.

3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical reports or guides and they are accepted by the National Committees in that sense.

4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.

5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.

6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61360-2 has been prepared by subcommittee 3D: Data sets for libraries, of IEC technical committee 3: Documentation and graphical symbols.

The text of this standard is based on the following documents:

| FDIS | Report on voting |
|---|---|
| 3D/53/FDIS | 3D/58/RVD |

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

IEC 61360 consists of the following parts under the general title Standard data element types with associated classification scheme for electric components:

Part 1: Definitions - Principles and methods
Part 2: EXPRESS Dictionary schema
Part 3: Maintenance and validation procedures
Part 4: IEC reference collection of standard data element types, component classes and terms.

Annexes A and B are for information only.

# STANDARD DATA ELEMENT TYPES WITH ASSOCIATED CLASSIFICATION SCHEME FOR ELECTRIC COMPONENTS –

# Part 2: EXPRESS Dictionary schema

## 1 General

### 1.1 Scope and object

The scope of this part of IEC 61360 is the common ISO/IEC dictionary schema based on the intersection of the scopes of the two base standards:

— IEC 61360-1, *Standard data element types with associated classification scheme for electric components - Part 1: Definitions - Principles and methods*, and

— ISO 13584-42, *Methodology for structuring part families*

The presented EXPRESS model represents a common formal model for the two standards and facilitates a harmonization of both.

Relevant parts of their scope clauses are cited below.

**From IEC 61360-1:**
" This part of IEC 61360 specifies the principles to be used for defining technical data element types with associated classification schemes needed to describe fully electric components, including electronic and electromechanical components and materials used in electro-technical equipment and systems."

**From ISO 13584-42:**
" This part of ISO 13584 specifies:

— the attributes that shall be provided by library suppliers to describe the families and properties of parts. These attributes are part of the content of their parts library and shall be stored in the dictionary of the user library;

— the specifications of these attributes in the EXPRESS information model that provides for the exchange of such dictionary data".

The object of this standard is to provide a formal model for data according to the scope as given above, and thus to provide a means for the computer-sensible representation and exchange of such data.

The intention is to provide a common information model for the work of both committees, thus allowing for the implementation of dictionary systems dealing with data delivered according to either of the standards elaborated by both committees.

Two schemas are provided in this part of IEC 61360 defining the two options that may be selected by an implementation. Each of these options is referred to as a conformance class.

— The **ISO13584_IEC61360_dictionary_schema**[1] provides for modelling and exchanging technical data element types with associated classification scheme but without modelling the definitions of the terms used in the data element type definitions. It constitutes conformance class 1 of this part of IEC 61360.

— The **IEC61360_extended_dictionary_schema** provides for modelling and exchanging technical data element types with associated classification scheme together with modelling definitions of and references to the terms used in the data element type definitions. It constitutes conformance class 2 of this part of IEC 61360.

[1] All the names that stand for items, formally defined within the EXPRESS model, are presented in **bold face.**

When used together with ISO 10303-21, each schema defines one single exchange format.

The exchange format defined by conformance class 1 is fully compatible with the ISO 13584 series.

The exchange format defined by conformance class 2 provides also for exchanging the definitions of the terms used in the definitions of data element types and their associated classification scheme when such an exchange is required, despite the lack of compatibility with implementations compliant with the ISO 13584 series.

Both committees agreed NOT to change and/or modify the presented EXPRESS model independent of each other in order to guarantee the harmonization and the reusability of the work from both committees. Requests for amendments should therefore be sent to both committees. These requests should be adopted by both committees before modifying the EXPRESS information model.

## 1.2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61360. At the time of publication, the editions indicated were valid. All normative documents are subject to revision, and parties to agreements based on this part of IEC 61360 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

| | |
|---|---|
| IEC 61360 (all parts), | *Standard data element types with associated classification scheme for electric components* |
| IEC 61360-1: 1995, | *Standard data element types with associated classification scheme for electric components - Part 1: Definitions - Principles and methods* |
| IEC 61360-4: 1997, | *Standard data element types with associated classification scheme for electric components - Part 4: IEC reference collection of standard data element types, component classes and terms* |
| ISO 31 (all parts), | *Quantities and units* |
| ISO 639: 1988, | *Code for the representation of names of languages* |
| ISO 843: 1997, | *Information and documentation - Conversion of Greek characters into Latin characters* |
| ISO 4217: 1995, | *Codes for the representation of currencies and funds* |
| ISO 6093: 1985, | *Information processing - Representation of numerical values in character strings for information interchange* |
| ISO 8601: 1988, | *Data elements and interchange formats - Information interchange - Representation of dates and times* |
| ISO 8859-1: 1987, | *Information processing - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1* |
| ISO 8879: 1986, | *Information processing - Text and office systems - Standard Generalized Markup Language (SGML)* |
| ISO 9735: 1988, | *Electronic data interchange for administration, commerce and transport (EDIFACT) - Application level syntax rules* |

| | |
|---|---|
| ISO 10303-11: 1994, | *Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual* |
| ISO 10303-21: 1994, | *Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure* |
| ISO 10303-41: 1994, | *Industrial automation systems and integration - Product data representation and exchange - Part 41: Integrated generic resources: Fundamentals of product description and support* |
| ISO 10303-42: 1994, | *Industrial automation systems and integration - Product data representation and exchange - Part 42: Integrated generic resources: Geometric and topological representation* |
| ISO 12083: 1994, | *Information and documentation - Electronic manuscript preparation and markup* |
| ISO 13584 (all parts), | *Industrial automation systems and integration - Parts library*[2] |
| ISO 13584-42:---, | *Industrial automation systems and integration - Parts library - Part 42: Description methodology: Methodology for structuring part families*[3] |

## 2 Definitions

For the purpose of this part of IEC 61360, the following definitions apply.

**2.1**
**Basic Semantic Unit (BSU)**
entity that provides an absolute and universal identification of certain objects of the application domain (e.g. classes, data element types)

**2.2**
**dictionary element**
set of attributes that constitutes the dictionary description of certain objects of the application domain (e.g. classes, data element types)

**2.3**
**common dictionary schema**
information model for a dictionary, using the information modelling language EXPRESS

**2.4**
**data type**
set of allowed values of a data element type

NOTE – Within IEC the **data_type** that is either a unit of measure or a value domain is defined separately for each data element type.

**2.5**
**IEC root class**
class that is the superclass of all the classes defined in IEC 61360-4; its class code is ‘AAA000' and its version is ‘001'

[2] To be published.
[3] To be published.

**2.6**
**applicable data element type**
data element type that is defined for a component class and that applies to any component that belongs to this component class

**2.7**
**visible data element type**
data element type that is defined for acomponent class and that may or may not apply to the different components of this component class

NOTE 1 The code of the class where a data element type is defined as visible is part of the identification of this data element type.

NOTE 2 Within IEC all data element types are defined as visible at the level of the root class, that is the superclass of both the component class and the material class.

## 3 Abbreviations

In this standard the following abbreviations are used:

— BSU: Basic Semantic Unit
— DET: Data Element Type
— ICS: International Classification of Standards
— SI: International System of Units

## 4 Overview of the common dictionary schema and compatibility with ISO 13584

In the following subclauses the architecture of the common dictionary schema will be presented and it will be explained how the same information model has to be used in the International Standards to ensure their compatibility.

The common dictionary schema combines the requirements of IEC 61360 and ISO 13584. Therefore, it contains resources to accommodate the specific requirements of both International Standards. These resources are provided either as optional capability or as subtypes of the types defined to fulfil the common requirements.

### 4.1 Use of the common dictionary schema to exchange IEC 61360-1 compliant data

a) The ISO 13584 specific extensions shall not be used for the exchange of dictionary elements defined according to IEC 61360-1.
These extensions are **present_translations**, **translated_label** and **translated_text** that provide for multilingual capabilities.

b) The fixed length of some strings is greater than the size defined in IEC 61360-1. When, however, these strings are exchanged using the common dictionary schema, they shall be padded as follows:
— **code** of a component class: the six character long IEC code shall be left justified in the fourteen character long exchange format that shall be padded with eight underscores ('_');
— **code** of a data element type: the six character long IEC code shall be left justified in the fourteen character long exchange format that shall be padded with eight underscores ('_');
— **value code** of the value of a data element type: the variable character length of the IEC code shall be left justified in the eighteen character long exchange format that shall be padded with underscores ('_');
— **coded name** of a component class: the variable character length of the IEC code shall be left justified in the eighteen character long exchange format that shall be padded with underscores ('_');
— **revision number:** the two character long IEC code shall be right justified and shall be padded with one zero ('0').

c) If a component class has a superclass, the **coded_name** shall be defined as a **value_code** in the **domain** of the classifying data element type of the superclass.

d) If a classifying data element type exists within a specific component class, for each **value** in its **domain** a subclass and a **term** shall be defined.

e) A classifying data element type, optional in conformance class 2 in the common dictionary schema, shall always be provided for the component classes defined according to IEC 61360-1.

f) Only SI units shall be used although the common dictionary schema enables the use of many kinds of system units. When using this schema however for the exchange of IEC 61360 compliant data, only SI units shall be used for quantitative data element types.

## 4.2 Compatibility with ISO 13584-42

An implementation compliant with this part of IEC 61360 shall support all the entities, types and associated constraints that belong to the conformance class it claims to support.
Therefore, conformity with conformance class 1 of this part of IEC 61360 requires that all the entities, types and associated constraints defined in the common dictionary schema be supported. ISO 13584 data conforming to the common dictionary schema may thus be processed by an IEC 61360 implementation, whether it conforms to conformance class 1 or to conformance class 2 that includes all the features of conformance class 1.

In ISO 13584, a specific conformance class[4] is intended to contain all the entities, types and associated constraints defined in the common dictionary schema. An ISO 13584 compliant implementation conforming to this conformance class shall therefore be able to support IEC data that belongs to conformance class 1 of this part of IEC 61360.

## 4.3 Naming correspondence between IEC 61360-1 and IEC 61360-2

Due to specific application restrictions - e.g. the EXPRESS language allows no spaces in names -, a number of similar 'EXPRESS names' are created by replacing the blank in a name by an underscore (e.g. preferred name is presented as **preferred_name**).

At other places names are used in the EXPRESS model that deviate from those used in IEC 61360-1. This is a consequence of the effort to reach one common EXPRESS information model together with parts libraries.

The table below offers a guide for matching the names used in the two parts of IEC 61360.

**X-REFERENCE table**

| Naming in 61360-2 | Naming in 61360-1 |
|---|---|
| **component_class** | Component class |
| **condition_DET** | Condition data element type |
| **dependent_P_DET** | Data element type |
| **det_classification** | Data element type class |
| **(DER)dic_identifier** | Identifier |
| **dic_value** | Value |
| **material_class** | Material class |
| **meaning** | Value meaning |
| **non_dependent_P_DET** | Data element type |
| **preferred_symbol** | Preferred letter symbol |
| **revision** | Revision number |
| **source_doc_of_definition** | Source document of data element type definition |
| **source_doc_of_definition** | Source document of component class definition |
| **source_doc_of_definition** | Source document of term definition |
| **synonymous_symbols** | Synonymous letter symbol |
| **unit** | Unit of measure |
| **value_code** | Value code |
| **version** | Version number |

[4] This conformance class is defined as conformance class 0 in ISO 13584-24.

## 4.4 Main structure of the common dictionary schema

This subclause explains the main resource constructs provided by the common dictionary schema.

— **dictionary_element** is any element defined in the dictionary;

— **supplier_element** captures the data of suppliers of dictionary elements (classes, properties, data types);

— **class** models the dictionary element of classes (families) which are described by properties;

— **property_DET** is the dictionary element of a property;

— **data_type** specifies the type of a property.

These parts of the dictionary schema are presented in more detail in clause 5: ISO13584_IEC61360_dictionary_schema.

In the presentation of the common dictionary schema, some overview diagrams are provided as planning models (figures 1 to 11). These planning models use the EXPRESS-G graphical notation for the EXPRESS language.

For clarification of the diagrams, some of the relationships that are defined in the EXPRESS model are omitted. Figure 1 below outlines as a planning model the main structure of the common dictionary schema.

Most of these figures contain overview models (or planning models) but show only that level of detail which is appropriate at a certain place.

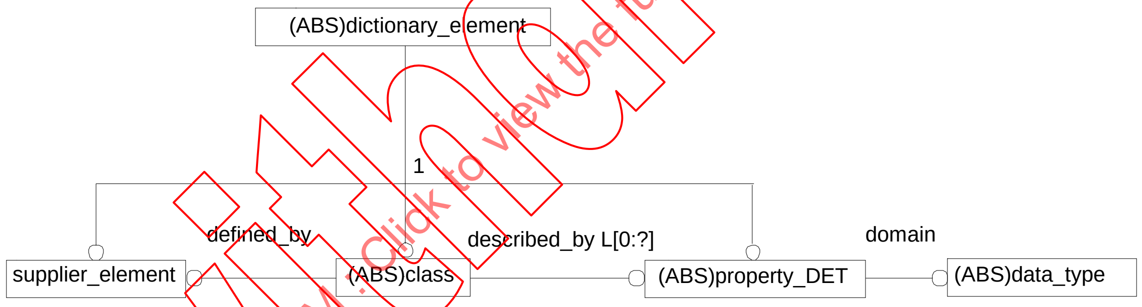

**Figure 1 - Overview of the Dictionary Schema**

## 5 ISO13584_IEC61360_dictionary_schema

This clause, which constitutes the main part of the common information model of ISO 13584-42 and IEC 61360, contains the full EXPRESS listing of the dictionary schema, annotated with comments and explanatory text. The order of text in this clause is determined firstly by the order imposed by the EXPRESS language, secondly by importance.

```
*)
SCHEMA ISO13584_IEC61360_dictionary_schema;
(*
```

### 5.1 References to other schemata

This subclause contains references to other EXPRESS schemata which are used in the Dictionary Schema. Their source is indicated in the respective comment.

```
*)
REFERENCE FROM support_resource_schema (identifier, label, text);
       (*    from ISO 10303-41: STEP Part 41: "Fundamentals of Product
             Description and Support" *)
REFERENCE FROM person_organization_schema (organization, address);
       (*    from ISO 10303-41: STEP Part 41: "Fundamentals of Product
             Description and Support" *)
REFERENCE FROM measure_schema;
       (*    from ISO 10303-41: STEP Part 41: "Fundamentals of Product
             Description and Support" *)
REFERENCE FROM ISO13584_IEC61360_language_resource_schema;
       (*    see clause 6 "ISO13584_IEC61360_language_resource_schema" *)
(*
```

## 5.2 Constant definitions

This subclause contains constant definitions used later in subclause 5.8 (basic type and entity definitions).

EXPRESS specification:

```
*)
CONSTANT
property_code_len:      INTEGER:= 14;
class_code_len:         INTEGER:= 14;
data_type_code_len:     INTEGER:= 14;
supplier_code_len:      INTEGER:= 18;
version_len:            INTEGER:= 3;
revision_len:           INTEGER:= 3;
value_code_len:         INTEGER:= 18;
pref_name_len:          INTEGER:= 30;
short_name_len:         INTEGER:= 15;
syn_name_len:           INTEGER:= pref_name_len;
DET_classification_len: INTEGER:= 3;
source_doc_len:         INTEGER:= 80;
value_format_len:  INTEGER:= 80;
sep_cv: STRING:= '-';
sep_id: STRING:= '.';
END_CONSTANT;
(*
```

## 5.3 Basic Semantic Units: defining and using the dictionary

### 5.3.1 Requirements for exchange

In the exchange of dictionary and parts library data it is customary to partition the data. For example, a dictionary could be updated with some classes that specify their superclass by a reference to a pre-existing class, or when the content of a library is exchanged, dictionary elements are only referenced and not included every time. It must be possible to refer unambiguously and consistently to the dictionary data.

Thus, it is a clear requirement first, to be able to exchange pieces of data, and secondly, to have relationships between these pieces. This is depicted in figure 2.

Every one of these pieces corresponds to a Physical File (according to ISO 10303-21). EXPRESS (see ISO 10303-11) attributes can only contain references to data within the same Physical File. Thus it is impossible to use EXPRESS attributes directly to implement inter-piece references.

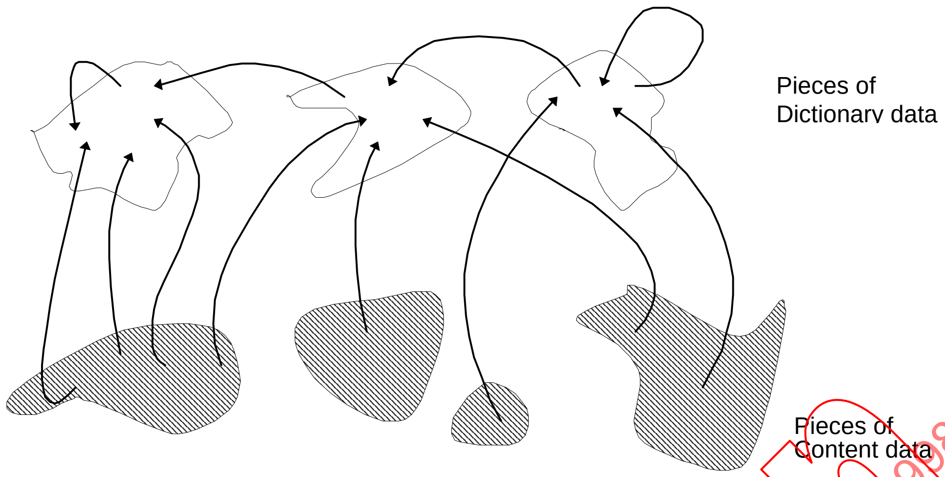

**Figure 2 - Pieces of data with relationships**

### 5.3.2 Three level architecture of the dictionary data

In this subclause the concept of **basic_semantic_unit** (BSU) is introduced as a means to implement these inter-piece references. A BSU provides a universally unique identification for dictionary descriptions. This is depicted in figure 3.

Assume that some piece of content (**content_item**) wants to refer to a certain dictionary description, e.g. to convey the value of a property of a component. It does this by referring to a Basic Semantic Unit through the attribute **dictionary_definition**.

A dictionary description (**dictionary_element**) refers to a Basic Semantic Unit through the attribute **identified_by**. From the correspondence of the absolute identifiers of the Basic Semantic Units this indirect relation is established.

Note that:

— both dictionary element and content item can be present in the same Physical File, but need not be;

— the dictionary element does not need to be present for the exchange of some Content Item referring to it. In this case it is assumed to be present in the dictionary of the target system already. Conversely, dictionary data can be exchanged without any content data;

— the basic semantic unit can be one single instance in the case where both dictionary element and content item instances are in the same Physical File;

— the same mechanism applies also to references between various dictionary elements (e.g. between a component class and the associated **property_DET**s).

A BSU provides a reference to a dictionary description in any place where this is needed, e. g. dictionary delivery, update delivery, library delivery, component data exchange. The data associated with a property could be exchanged as a couple (**property_BSU**, <value>).
Figure 3 outlines the implementation of this general mechanism.

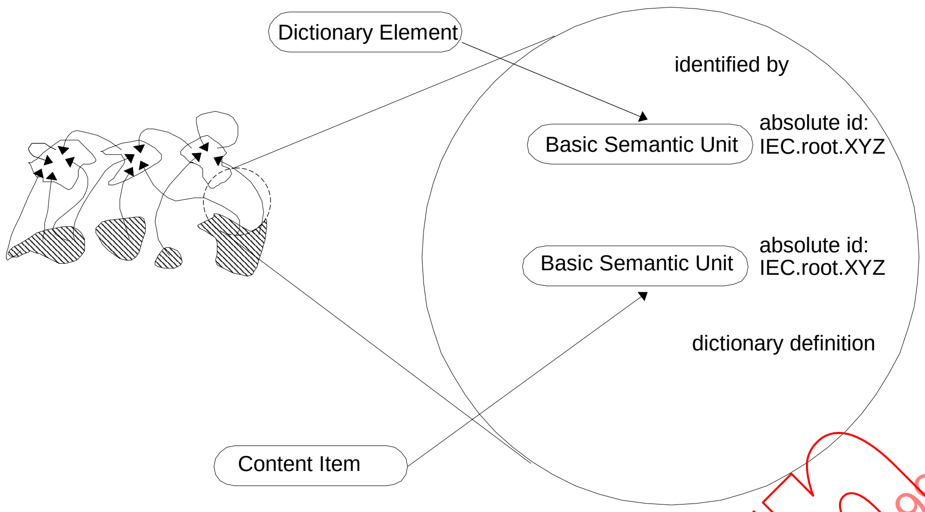

**Figure 3 - Implementation of "inter-piece" relationships using basic semantic units**

### 5.3.2.1 basic_semantic_unit

A **basic_semantic_unit** is a unique identification of a **dictionary_element**.

EXPRESS specification:

```
*)
ENTITY basic_semantic_unit
ABSTRACT SUPERTYPE OF ( ONEOF (
             supplier_BSU,
             class_BSU,
             property_BSU,
             data_type_BSU,
             supplier_related_BSU,
             class_related_BSU));
       code: code_type;
       version: version_Type;
DERIVE
       dic_identifier: identifier:= code + sep_cv + version;
INVERSE
       definition: SET [ 0: 1 ] OF dictionary_element FOR
       identified_by;
       referenced_by: SET [ 0: 1 ] OF content_item FOR
dictionary_definition;
END_ENTITY; -- basic_semantic_unit
(*
```

Attribute definitions:

**code**[5]: the code assigned to identify a certain dictionary element.

**version**[6]: the version number of a certain dictionary element.

**dic_identifier**[7]: the full identification, consisting of concatenation of code and version.

[5] See IEC 61360-1 clauses 3.2.1 and 5.3.1 .
[6] See IEC 61360-1 clauses 3.2.7 and 5.3.2 .
[7] See IEC 61360-1 clauses 3.2.9 and 5.3.4 .

**definition**: a reference to the dictionary element identified by this BSU. If not present in some exchange context, it is assumed to be present in the dictionary of the target system already.

**referenced_by:** items making use of the dictionary element associated with this BSU.

### 5.3.2.2 dictionary_element

A **dictionary_element** is a full definition of the data required to be captured in the semantic dictionary for some concepts. For every concept a separate subtype shall be used. The **dictionary_element** is associated with a **basic_semantic_unit**, which serves to uniquely identify this definition in the dictionary.
Figure 4 presents a planning model of the relationship between the basic semantic unit and the dictionary element.

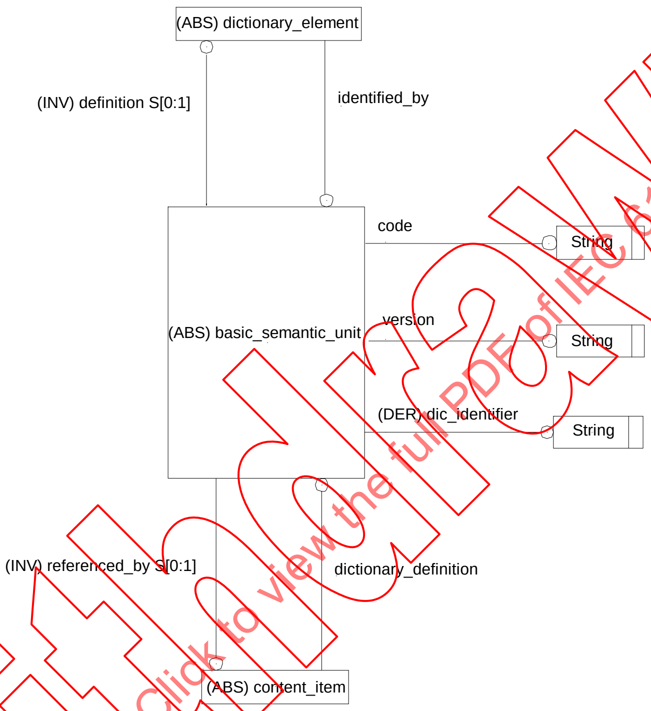

**Figure 4 - Relationship between basic semantic unit and dictionary element**

By including the version attribute in the **basic_semantic_unit** entity, it forms part of the identification of a dictionary element (in contrast to the **revision** and **time_stamps** attributes).

<u>EXPRESS specification:</u>

```
*)
ENTITY dictionary_element
ABSTRACT SUPERTYPE OF ( ONEOF (
             supplier_element,
             class_and_property_elements,
             data_type_element));
       identified_by: basic_semantic_unit;
       time_stamps: OPTIONAL dates;
       revision: revision_type;
END_ENTITY;
(*
```

Attribute definitions:

**identified_by**: the BSU identifying this dictionary element.

**time_stamps**: the optional dates of creation and update of this dictionary element.

**revision**[8]: the revision number of this dictionary element.

#### 5.3.2.3 content_item

A **content_item** is a piece of data referring to its description in the dictionary. It shall be subtyped.

EXPRESS specification:

```
*)
ENTITY content_item
ABSTRACT SUPERTYPE;
       dictionary_definition: basic_semantic_unit;
END_ENTITY;
(*
```

Attribute definitions:

**dictionary_definition:** the Basic Semantic Unit to be used for referring to the definition in the dictionary.

### 5.3.3 Overview of basic semantic units and dictionary elements

**For every kind of dictionary data a pair of basic_semantic_unit and dictionary_element** subtypes shall be defined. Figure 5 outlines, as a planning model, the Basic Semantic Units and Dictionary Elements defined later. Note that the relationship between BSUs and Dictionary Elements is redefined for each type of data, so that only corresponding pairs can be related. This is not graphically depicted here, however.

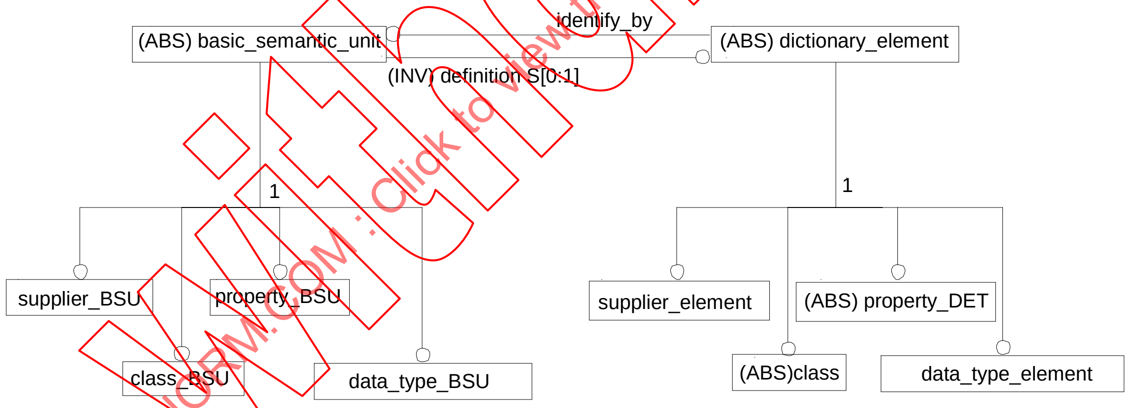

**Figure 5 - Current BSUs and dictionary elements**

Every kind of dictionary data is treated in one of the following subclauses:

— for suppliers, see 5.4 "Supplier data";
— for classes, see 5.5 "Class data";
— for properties / data element types, see 5.6 "Data element type / properties data";
— for data types, see 5.7 "Domain data: the type system".

[8] See IEC 61360-1 clauses 3.2.8 and 5.3.3 .

### 5.3.4 Identification of dictionary elements: three-level structure

The absolute identification of basic semantic units is based on the following three-level structure:

— supplier (of dictionary data);

— class;

— class-related dictionary elements (any dictionary element defined in the context of a class; in this standard class-related dictionary elements are **property_DET** and **data_type_element,** but there are provisions to extend this mechanism to other items).

An absolute identification can be achieved by concatenation of the applicable code for each level.

This identification scheme is appropriate within a multi-supplier context. If, in a certain application area, only data of one single (data-) supplier are relevant, the corresponding parts of the identification, which are then constant, can be eliminated. For the purpose of exchange, however, all the levels must be present, to avoid clashes of identifiers.

This identification scheme is described formally in the ..._BSU entities, in 5.3 through 5.7, attribute **absolute_id**.

### 5.3.5 Extension possibilities for other types of data

The BSU - Dictionary Element mechanism is very general and not limited to the four kinds of data used here (see figure 5). This subclause specifies some facilities which allow for extensions for other kinds. Depending on whether the scope of the identifier is given by a class or a supplier, the corresponding **......_related_BSU** entity has to be subtyped. It is necessary to redefine the **identified_by** attribute of the entity **dictionary_element** (as is done in 5.4 through 5.7 for the current kinds of data).

#### 5.3.5.1 supplier_related_BSU

The **supplier_related_BSU** provides for the dictionary elements to be associated with suppliers, e.g. for the ISO 13584 series: program libraries.

EXPRESS specification:

```
*)
ENTITY supplier_related_BSU
ABSTRACT SUPERTYPE
SUBTYPE OF (basic_semantic_unit);
END_ENTITY;
(*
```

#### 5.3.5.2 class_related_BSU

The **class_related_BSU** provides for the dictionary elements to be associated with classes, e.g. for ISO 13584 tables, documents, etc...

EXPRESS specification:

```
*)
ENTITY class_related_BSU
ABSTRACT SUPERTYPE
SUBTYPE OF (basic_semantic_unit);
END_ENTITY;
(*
```

#### 5.3.5.3 supplier_BSU_relationship

The **supplier_BSU_relationship** is a provision for association of BSUs with suppliers.

EXPRESS specification:

```
*)
ENTITY supplier_BSU_relationship
ABSTRACT SUPERTYPE;
       relating_supplier: supplier_element;
       related_tokens: SET [ 1: ? ] OF supplier_related_BSU;
END_ENTITY;
(*
```

Attribute definitions:

**relating_supplier:** the **supplier_element** which identifies the data supplier.

**related_tokens:** the set of dictionary elements associated with the supplier identified by the **relating_supplier** attribute.

### 5.3.5.4 class_BSU_relationship

The **class_BSU_relationship** entity is a provision for association of BSUs with classes.

EXPRESS specification:

```
*)
ENTITY class_BSU_relationship
ABSTRACT SUPERTYPE;
       relating_class: class;
       related_tokens: SET [ 1: ? ] OF class_related_BSU;
END_ENTITY;
(*
```

Attribute definitions:

**relating_class:** the class which identifies the dictionary element.

**related_tokens:** the set of dictionary elements associated to the class identified by the **relating_class** attribute.

## 5.4 Supplier data

This subclause contains definitions for the representation of data about a supplier itself. In a multi-supplier environment it is necessary to be able to identify the source of a certain dictionary element. Figure 6 presents a planning model of the data associated with suppliers, followed by the EXPRESS definition.

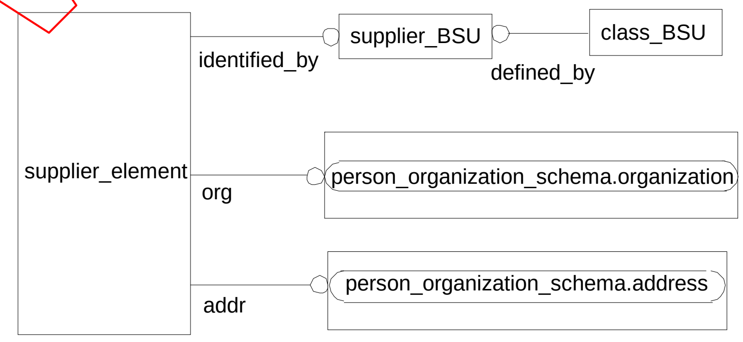
supplier_element
identified_by
supplier_BSU
defined_by
class_BSU
org
person_organization_schema.organization
addr
person_organization_schema.address

**Figure 6 - Overview of supplier data and relationships**

### 5.4.1 supplier_BSU

The **supplier_BSU** entity provides for unique identification of suppliers of dictionary data.

EXPRESS specification:

```
*)
ENTITY supplier_BSU
SUBTYPE OF (basic_semantic_unit);
       SELF\basic_semantic_unit.code: supplier_code_type;
DERIVE
       SELF\basic_semantic_unit.version: version_type:='001';
       absolute_id: identifier:= SELF\basic_semantic_unit.code;
UNIQUE
       UR1: absolute_id;
END_ENTITY;
(*
```

Attribute definitions:

**code:** the supplier's code assigned according to ISO 13584-26.

**version:** the version number of a supplier code shall be equal to 001.

**absolute_id:** the absolute identification of the supplier.

Formal propositions:

**UR1:** The supplier identifier defined by the **absolute_id** attribute is unique.

### 5.4.2 supplier_element

The **supplier_element** entity gives the dictionary description of suppliers.

EXPRESS specification:

```
*)
ENTITY supplier_element
SUBTYPE OF (dictionary_element);
       SELF\dictionary_element.identified_by: supplier_BSU;
       org: organization;
       addr: address;
INVERSE
       associated_items: SET [ 0: ? ] OF supplier_BSU_relationship
            FOR relating_supplier;
END_ENTITY;
(*
```

Attribute definitions:

**identified_by:** the **supplier_BSU** used to identify this **supplier_element.**

**org:** the organizational data of this supplier.

**addr:** the address of this supplier.

**associated_items:** allows access to other kinds of data via the BSU mechanism (e.g. program library in ISO 13584).

## 5.5 Class data

This subclause contains definitions for the representation of dictionary data of Classes.

### 5.5.1 General

Figure 7 outlines, as a planning model, the data associated with Classes and their relationship to other Dictionary Elements.

As indicated in figure 7 with the **its_superclass** attribute, classes form an inheritance tree. It is important to note that throughout this standard the terms "inheritance" and "to inherit" stand for this relationship between classes (defined in the dictionary), although EXPRESS has an inheritance concept, too. These shall be clearly distinguished to avoid misunderstandings.

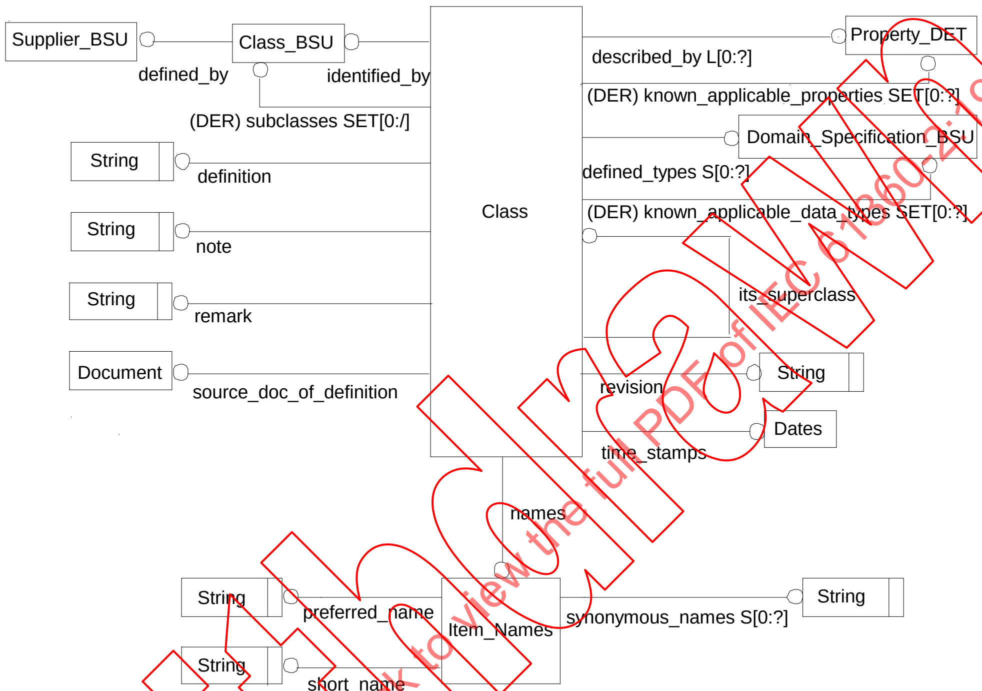

**Figure 7 - Overview of class data and relationships**

The dictionary data for Component Classes (as shown in figure 7) is spread over four inheritance levels:

— **class_and_property_elements** defines data common to both **classes** and **property_DETs**;

— **class** allows for other kinds of classes to be specified later (e.g. in ISO 13584-24);

— **item_class** is the entity to hold data of different classes of application domain objects (e.g. components, materials,...);

— **component_class** is the entity that models component classes and **material_class** is the entity that models classes of materials.

#### 5.5.1.1 class_BSU

The **class_BSU** entity provides for the identification of classes.

EXPRESS specification:

```
*)
ENTITY class_BSU
SUBTYPE OF (basic_semantic_unit);
       SELF\basic_semantic_unit.code: class_code_type;
       defined_by: supplier_BSU;
DERIVE
       absolute_id: identifier:= defined_by.absolute_id + sep_id +
             dic_identifier;
       known_visible_properties: SET [0: ?]OF property_BSU
             :=compute_known_visible_properties(SELF);
       known_visible_data_types: SET [0: ?]OF data_type_BSU
             :=compute_known_visible_data_types(SELF);
INVERSE
       subclasses: SET [0: ?] OF class FOR its_superclass;
       added_visible_properties:SET [0: ?] OF property_BSU
             FOR name_scope;
       added_visible_data_types:SET [0: ?] OF data_type_BSU
             FOR name_scope;
UNIQUE
       UR1: absolute_id;
END_ENTITY; -- class_BSU
(*
```

Attribute definitions:

**code:** the code assigned to this class by its supplier.

**defined_by:** the supplier defining this class and its dictionary element.

**absolute_id:** the unique identification of this class.

**known_visible_properties**[9]: the set of **property_BSUs** that refer to the class as their **name_scope** attribute or to any known superclass of this class and that are therefore visible for the class (and any of its subclass).

**known_visible_data_types**[10]: the set of **data_type_BSUs** that refer to the class as their **name_scope** attribute or to any known superclass of this class and that are therefore visible for the class (and any of its subclass).

**subclasses:** the set of classes specifying this class as their superclass.

**added_visible_properties**[11]: the set of **property_BSUs** that refer to the class as their **name_scope** and that are therefore visible for the class (and any of its subclass).

**added_visible_data_types**[12]: the set of **data_type_BSUs** that refer to the class as their **name_scope** and that are therefore visible for the class (and any of its subclass).

[9] Within IEC all the data element types refer to the IEC root at their **name_scope** attribute. Therefore, all the data element types defined within IEC are visible for every IEC class.

[10] The capability to define visible data types that may be reused for different data element types is not used in the current version of IEC 61360. Therefore, the **added_visible_data_types** attribute is empty for all classes.

[11] Within IEC all the data element types refer to the IEC root at their **name_scope** attribute. Therefore, the **added_visibility_properties** attribute is empty for all the other classes.

[12] The capability to define **data_types** that may be reused for different data element types is not used in the current version of IEC 61360.

Formal propositions:

**UR1:** The concatenation of supplier code and class code is unique.

### 5.5.1.2 class_and_property_elements

The **class_and_property_elements** entity captures the attributes which are common to both classes and **property_DETs.**

EXPRESS specification:

```
*)
ENTITY class_and_property_elements
ABSTRACT SUPERTYPE OF ( ONEOF (
             property_DET,
             class))
SUBTYPE OF (dictionary_element);
       names: item_names;
       definition: definition_type;
       source_doc_of_definition: OPTIONAL document;
       note: OPTIONAL note_type;
       remark: OPTIONAL remark_type;
END_ENTITY;
(*
```

Attribute definitions:

**names**[13]: the names describing this dictionary element.

**definition**[14]: the text describing this dictionary element.

**source_doc_of_definition**[15]: the source document of this textual description.

**note**[16]: further information on any part of the dictionary element, which is essential to the understanding.

**remark**[17]: explanatory text further clarifying the meaning of this dictionary element.

### 5.5.1.3 class

The **class**[18] entity is an abstract resource for all kinds of classes.

EXPRESS specification:

```
*)
ENTITY class
ABSTRACT SUPERTYPE OF (ONEOF (item_class))
SUBTYPE OF (class_and_property_elements);
       SELF\dictionary_element.identified_by: class_BSU;
       its_superclass: OPTIONAL class_BSU;
       described_by: LIST [0: ?]OF UNIQUE property_BSU;
       defined_types: SET [ 0: ? ] OF data_type_BSU;
```

[13] See IEC 61360-1 clauses 3.2.2, 3.2.3, 3.2.6 and 5.3.5 .
[14] See IEC 61360-1 clauses 3.3.1 and 5.4.1 .
[15] See IEC 61360-1 clauses 3.3.6 and 5.4.4 .
[16] See IEC 61360-1 clauses 3.3.2 and 5.4.2 .
[17] See IEC 61360-1 clauses 3.3.3 and 5.4.3 .
[18] See IEC 61360-1 figure 12**.**

```
DERIVE
       subclasses: SET [0: ?] OF class:= identified_by.subclasses;
       known_applicable_properties: SET [ 0: ? ] OF property_BSU
             := compute_known_applicable_properties(
                   SELF\dictionary_element.identified_by);
       known_applicable_data_types: SET [ 0: ? ] OF data_type_BSU
             := compute_known_applicable_data_types(
                   SELF\dictionary_element.identified_by);
INVERSE
       associated_items: SET [ 0: ? ] of class_BSU_relationship FOR
             relating_class;
WHERE
       WR1: acyclic_superclass_relationship ( SELF.identified_by, [ ] );
       WR2: NOT all_class_descriptions_reachable (
                   SELF\dictionary_element.identified_by)
             OR (list_to_set (SELF. described_by) <=
       SELF\dictionary_element.identified_by.known_visible_properties);
       WR3: NOT all_class_descriptions_reachable (
                   SELF\dictionary_element.identified_by)
             OR (SELF. defined_types <=
       SELF\dictionary_element.identified_by.known_visible_data_types);
       WR4: list_to_set (SELF. described_by) *
             NVL (SELF.its_superclass.definition[1]
                   .known_applicable_properties, [])= [] ;
       WR5: SELF.defined_types *
             NVL (SELF.its_superclass.definition[1]
                   ;known_applicable_data_types, [])= [] ;
END_ENTITY;
(*
```

Attribute definitions:

**identified_by:** the **class_BSU** identifying this class.

**its_superclass:** reference to the class of which the current one is a subclass.

**described_by:** the list of references to the additional properties available for use in the description of the part within the class, and any of its subclasses.

**defined_types**[19]: the optional set of references to the types that can be used for various **property_DET**s throughout the inheritance tree descending from this class.

**subclasses:** the set of classes specifying this class as their superclass[20].

**known_applicable_properties**[21]: the **property_BSUs** that are referenced by the class or any of its known superclass by their **described_by** attribute, and that are therefore applicable to this class (and to any of its subclass).

**known_applicable_data_types**[22]: the **data_type_BSUs** that are referenced by the class or any of its known superclass by their **defined_types** attribute, and that are therefore applicable to this class (and to any of its subclass).

[19] The capability to define **data_types** that may be reused for different data element types is not used in the current version of IEC 61360.

[20] According to IEC 61360-1 a class shall have 0 or more than 1 subclass .

[21] According to IEC 61360-1 subclause 5.1, the data element types that apply to some component classes apply also to all component classes at lower levels. This attribute gathers the data element types that apply to a class either by virtue of belonging to the **described_by** attribute or by inheritance.

[22] This attribute gathers the **data_types** that may be referenced from a component class either by virtue of belonging to the **described_by** attribute or by inheritance. Note that the capability to define **data_types** that may be reused for different data element types is not used in the current version of IEC 61360.

**associated_items:** allows to access other kinds of data using the BSU mechanism.

Formal propositions:

**WR1:** the inheritance structure defined by the class hierarchy does not contain cycles.

**WR2:** only those properties that are visible for a class may become applicable to this class by virtue of being referenced by the **described_by** attribute.

**WR3:** only those data types that are visible for a class may become applicable to this class by virtue of being referenced by the **defined_types** attribute.

**WR4:** only those properties that are not applicable for a class by inheritance may become applicable to this class by virtue of being referenced by the **described_by** attribute.

**WR5:** only those data types that are not applicable for a class by inheritance may become applicable to this class by virtue of being referenced by the **defined_types** attribute.

Informal propositions:

The **described_by** attribute contains references (using the BSU concept) to the **property_DET**s which describe this class. This list shall contain only those which are defined at this node of the inheritance tree. Inherited **property_DET**s shall not be repeated.

## 5.5.2 item_class

The entity **item_class** enables the modelling of any type of entity of the application domain that corresponds to an autonomous and stand-alone abstraction as a class. It is a supertype intended to be sub-typed to define the nature of the objects. Nevertheless, it is not defined as ABSTRACT to enable its instanciation to model the classes that are superclasses of two classes corresponding to two different kinds of objects (e.g. components and materials).

EXAMPLE 1 — A material is an autonomous and stand-alone abstraction of an object of the parts library application domain. It is represented as a specific subclass of **item_class**.

EXAMPLE 2 — A feature is an autonomous and stand-alone abstraction of an object of the parts library application domain. It might be represented as a specific subclass of **item_class**.

EXAMPLE 3 — A product representation is not an autonomous and stand-alone abstraction of an object of the parts library application domain: it may only exist with a relation to a product. In ISO 13584-24, part representations are represented as a specific subclass of **class**.

EXPRESS specification:

```
*)
ENTITY item_class
SUPERTYPE OF (ONEOF(component_class, material_class))
SUBTYPE OF (class);
       simplified_drawing: OPTIONAL graphics;
       sub_class_properties: SET [ 0: ? ] OF property_BSU;
       class_constant_values: SET [ 0: ? ] OF class_value_assignment;
       coded_name: OPTIONAL value_code_type;
WHERE
       WR1:  QUERY (p <* sub_class_properties
             | NOT (p IN SELF.described_by)) = [ ];
       WR2:  NOT all_class_descriptions_reachable(SELF.identified_by) OR
             (QUERY (va <* class_constant_values | SIZEOF (QUERY (c <*
             va.super_class_defined_property.describes_classes |
             is_subclass (SELF, c)
             AND (va.super_class_defined_property
             IN c.sub_class_properties))) <> 1)   = []);
END_ENTITY;
(*
```

Attribute definitions:

**simplified_drawing:** optional drawing **(graphic)** that can be associated to the described class.

**sub_class_properties**[23]: declares properties as class-valued, i.e. in subclasses one single value will be assigned per class. See 5.6.4 "Class-valued properties ".

**class_constant_values**[24]: assignments in the current class for class-valued properties declared in superclasses. See 5.6.4 "Class-valued properties ".

**coded_name**[25]: to be used in the value domain of the Classifying DET of the superclass.

Formal propositions:

**WR1:** The **class_valued_properties** belong to the **described_by** list .

**WR2:** The properties referenced in **class_constant_values** were declared as class-valued in some superclasses of the current class.

### 5.5.3 component_class

The entity **component_class** captures the dictionary description of a class of items that represent, at some level of abstraction, parts or components. A property of which the data type is defined by a **component_class** stands for the aggregation relationship.

EXPRESS specification:

```
*)
ENTITY component_class
SUBTYPE OF (item_class);
END_ENTITY;
(*
```

### 5.5.4 material_class

The entity **material_class** captures the dictionary description of a class of materials. Materials are used to define properties of parts or components. Materials are associated with an idea of amount, they may not be counted. A property of which the data type is defined by a **material_class** captures that some (part of a) product is made of, or contains, some material.

```
*)
ENTITY material_class
SUBTYPE OF (item_class);
END_ENTITY;
(*
```

## 5.6 Data Element Type / properties data

This subclause contains definitions for the dictionary data for properties.

### 5.6.1 property_BSU

The entity **property_BSU** provides for identification of a property.

[23] See IEC 61360-1 clause 5.1 .

[24] See IEC 61360-1 clause 5.1 .

[25] See IEC 61360-1 clause 5.3.6 .

EXPRESS specification:

```
*)
ENTITY property_BSU
SUBTYPE OF (basic_semantic_unit);
       SELF\basic_semantic_unit.code: property_code_type;
       name_scope: class_BSU;
DERIVE
       absolute_id: identifier:=
             name_scope.defined_by.absolute_id
             + sep_id + name_scope.dic_identifier
             + sep_id + dic_identifier;
INVERSE
       describes_classes: SET OF class FOR described_by;
UNIQUE
       UR1: absolute_id;
WHERE
       WR1: QUERY ( c <* describes_classes |
             NOT ( is_subclass (c, name_scope.definition[1]\class )))= [ ];
END_ENTITY;
(*
```

Attribute definitions:

**code:** to allow for unique identification within the scope indicated by the **name_scope** attribute.

**name_scope**[26]: is the reference to the class at which or below which the property element is available for reference by the **described_by** attribute.

**absolute_id:** the unique identification of this property.

**describes_classes**[27]: the classes declaring this property as available for use in the description of a part.

Formal propositions:

**WR1:** The class used in the **name_scope** attribute is a superclass of the one(s) where this **property_DET** is defined.

**UR1:** The property identifier **absolute_id** is unique.

### 5.6.2 property_DET

The **property_DET** entity captures the dictionary description of properties.

EXPRESS specification:

```
*)
ENTITY property_DET
ABSTRACT SUPERTYPE OF ( ONEOF (
       condition_DET,
       dependent_P_DET,
       non_dependent_P_DET))
SUBTYPE OF (class_and_property_elements);
       SELF\dictionary_element.identified_by: property_BSU;
       preferred_symbol: OPTIONAL mathematical_string;
       synonymous_symbols: SET [ 0: 2 ] OF mathematical_string;
       figure: OPTIONAL graphics;
       det_classification: OPTIONAL DET_classification_type;
       domain: data_type;
       formula: OPTIONAL mathematical_string;
```

[26] Within IEC 61360 all the data element types refer to the IEC root class by their **name_scope** attribute. The **dic_identifier** of the IEC root class is AAA000-001.

[27] See IEC 61360-1 clause 3.5.1 .

```
DERIVE
       describes_classes: SET [ 0: ? ] OF class
             := identified_by.describes_classes;
END_ENTITY;
(*
```

Attribute definitions:

**identified_by:** the **property_BSU** identifying this property.

**preferred_symbol**[28]: a shorter description of this property.

**synonymous_symbols**[29]: synonymous for the shorter description of the property.

**figure**[30]: an optional **graphic** which describes the property.

**det_classification**[31]: the ISO 31 class for this property.

**domain**: the reference to the **data_type** associated with the property.

**formula**[32]: a mathematical expression for explaining the property.

**describes_classes**[33]: the classes declaring this property as available for use in the description of a part.

Figure 8 presents a planning model of the data associated with **property_DETs.**

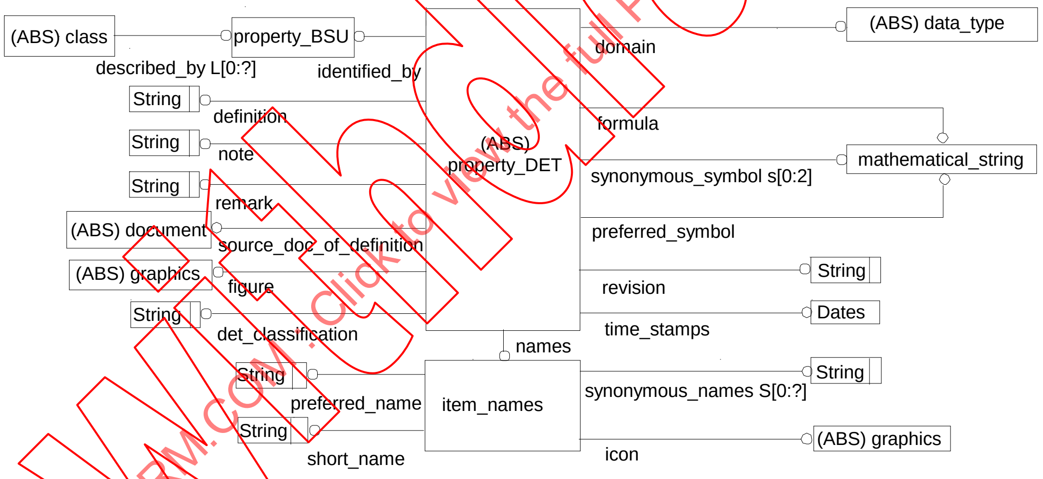

**Figure 8 - Overview of property data element type data and relationships**

[28] See IEC 61360-1 clauses 3.2.4 and 3.2.6 .

[29] See IEC 61360-1 clause 3.2.5 .

[30] See IEC 61360-1 clause 3.3.4 .

[31] See IEC 61360-1 clause 3.5.2 . The **det_classification** within IEC 61360 is mandatory.

[32] See IEC 61360-1 clause 3.3.5 .

[33] See IEC 61360-1 clause 3.5.1 .

### 5.6.3 Condition, dependent and non-dependent Data Element Types

Figure 9 depicts the various kinds of Data Element Types in the format of a planning model.

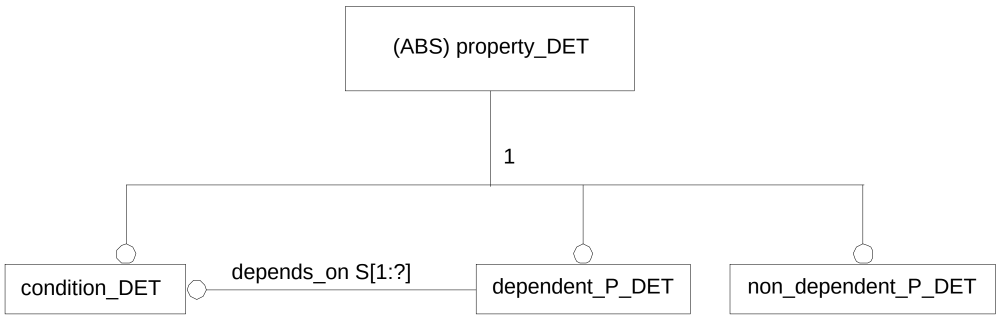

**Figure 9 - Kinds of data element types**

Note that this figure (like others) is simplified: the "depends_on" relation is essentially implemented with a BSU reference, but a constraint is specified that the referred-to **property_DET** shall be a **condition_DET** (see entity **dependent_P_DET**).

#### 5.6.3.1 condition_DET[34]

A **condition_DET** is a property upon which other properties may depend upon.

EXPRESS specification:

```
*)
ENTITY condition_DET
SUBTYPE OF (property_DET);
END_ENTITY;
(*
```

#### 5.6.3.2 dependent_DET[35]

A **dependent_P_DET** is a property whose value depends explicitly on the value(s) of some condition(s), like e.g. ambient temperature.

EXPRESS specification:

```
*)
ENTITY dependent_P_DET
SUBTYPE OF (property_DET);
       depends_on: SET [ 1: ? ] OF property_BSU;
WHERE
       WR1: QUERY ( p <* depends_on | NOT (definition_available_implies (p,
             ('ISO13584_IEC61360_DICTIONARY_SCHEMA.CONDITION_DET'
             IN TYPEOF (p.definition)))) ) = [ ];
END_ENTITY;
(*
```

Attribute definitions:

**depends_on:** the set of basic semantic units identifying the properties on which this property depends on.

[34] See IEC 61360-1 clause 3.5.3 .

[35] See IEC 61360-1 clause 3.5.3 .

Formal propositions:

**WR1:** Only **condition_DET**s shall be used in the **depends_on** list.

#### 5.6.3.3 non-dependent_DET[36]

A **non_dependent_P_DET** is a property that does not depend explicitly on certain conditions.

EXPRESS specification:

```
*)
ENTITY non_dependent_P_DET
SUBTYPE OF (property_DET);
END_ENTITY;
(*
```

### 5.6.4 Class_valued properties

Class-valued properties are those properties to which in each subclass one single value, valid for the whole subclass, is assigned, not for every instance within the class individually. By being included in the list **sub_class_properties** in entity **item_class** a property is distinguished as being of that type. In subclasses which inherit this property, values may be assigned using the attribute **class_constant_values** which contains a set of **class_value_assignments**.

EXPRESS specification:

```
*)
ENTITY class_value_assignment;
       super_class_defined_property: property_BSU;
       assigned_value: value_code_type;
WHERE
       WR1:  definition_available_implies (super_class_defined_property,
       ('ISO13584_IEC61360_DICTIONARY_SCHEMA'+'.NON_QUANTITATIVE_CODE_TYPE'
             IN TYPEOF (

       super_class_defined_property.definition[1]\property_DET.domain)));
       WR2:  definition_available_implies (super_class_defined_property,
             ( SIZEOF ( QUERY ( v <*
             super_class_defined_property.definition[1]\property_DET.domain
             \non_quantitative_code_type.domain.its_values |
             assigned_value = v.value_code)) = 1));
END_ENTITY;
(*
```

Attribute definitions:

**super_class_defined_property:** the reference to the property (defined in a superclass as being a **class_valued_property**) to which a certain value is assigned.

**assigned_value:** the value assigned to the property, valid for the whole class, referring this **class_value_assignment**[37] instance in the **class_constant_values** list.

Formal propositions:

**WR1:** the **super_class_defined_property** shall be of the non-quantitative type, with codes (strings) as values.

[36] See IEC 61360-1 clause 3.5.3 .

[37] See IEC 61360-1 clause 5.1 .

**WR2:** the value assigned to the **super_class_defined_property** shall be of the compatible type, i.e. it occurs in the value domain of the **super_class_defined_property**.

## 5.7 Domain data: the type system

The following subclauses contain definitions for the representation of the data types of a **property_DET**. Figure 10 outlines, as a planning model, the entity hierarchy for data types.

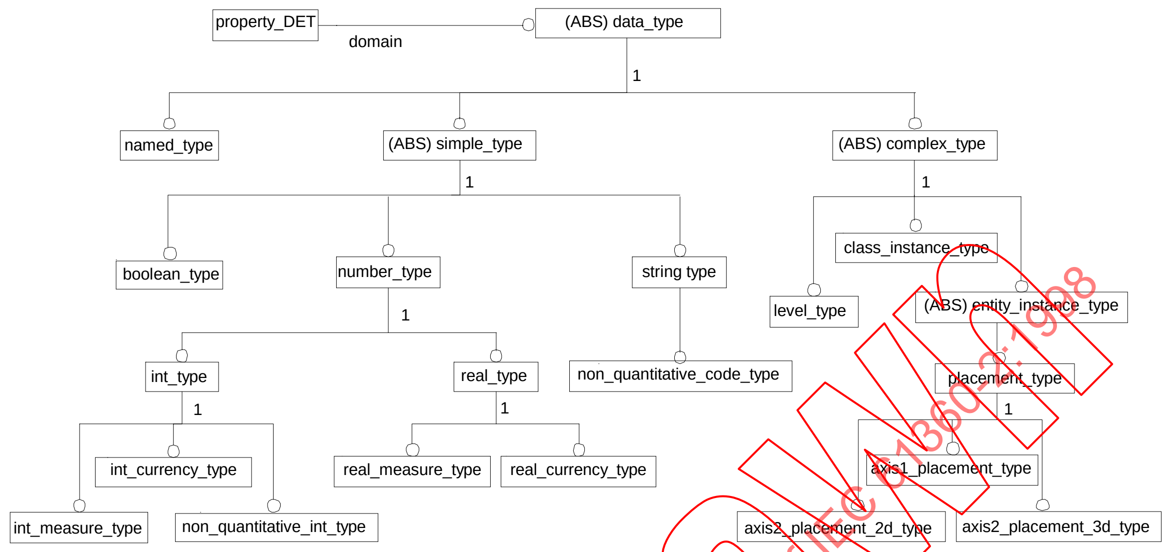

**Figure 10 - Entity hierarchy for the type system**

### 5.7.1 General

In contrast to the other dictionary elements (Suppliers, Classes, Properties) an identification with the Basic Semantic Unit concept is not mandatory for **data_type**, since it will be attached directly to the **property_DET** in many cases, and thus doesn't need an identification. However, the entities **data_type_BSU** and **data_type_element** allow for a unique identification where this is suitable. It provides for re-using the same type definition in another **property_DET** definition, even outside the current Physical File.

#### 5.7.1.1 data_type_BSU

The **data_type_BSU** entity provides for identification of **data_type_elements**.

EXPRESS specification:

```
*)
ENTITY data_type_BSU
SUBTYPE OF (basic_semantic_unit);
       SELF\basic_semantic_unit.code: data_type_code_type;
       name_scope: class_BSU;
DERIVE
       absolute_id: identifier:=
             name_scope.defined_by.absolute_id
             + sep_id + name_scope.dic_identifier
             + sep_id + dic_identifier;
INVERSE
       defining_class: SET [ 0: 1 ] OF class FOR defined_types;
UNIQUE
       absolute_id;
WHERE
       WR1: is_subclass ( defining_class[1], name_scope.definition[1] );
END_ENTITY;
(*
```

Attribute definitions:

**code:** to allow for unique identification within the scope indicated by the **name_scope** attribute.

**name_scope**: the reference to the class at which or below which the data type element is available for reference by the **defined_types** attribute.

**absolute_id:** the unique identification of this property.

**defining_class:** the classes declaring this **data_type** as available for use in the description of a part.

Formal propositions:

**WR1**: The class used in the **name_scope** attribute is a superclass of the one where this **data_type** is defined.

#### 5.7.1.2 data_type_element

The **data_type_element** entity describes the dictionary element for types. Note that it is not necessary in every case to have BSU and **dictionary_element** for a certain **data_type**, because a **property_DET** can refer to the **data_type** directly. Usage of the BSU relation is only necessary when a supplier wants to refer to the same type in a different Physical File.

EXPRESS specification:

```
*)
ENTITY data_type_element
SUBTYPE OF (dictionary_element);
       SELF\dictionary_element.identified_by: data_type_BSU;
       names: item_names;
       type_definition: data_type;
END_ENTITY;
(*
```

Attribute definitions:

**identified_by**: the BSU that identifies the described **data_type_element.**

**names:** the names that allow the description of the defined **data_type_element**.

**type_definition:** the description of the type carried by the **data_type_element**.

### 5.7.2 The type system

#### 5.7.2.1 Data_type

The **data_type** entity serves as a common supertype for the entities used to indicate the type of the associated DET.

EXPRESS specification:

```
*)
ENTITY data_type
ABSTRACT SUPERTYPE OF ( ONEOF (
       simple_type,
       complex_type,
       named_type));
END_ENTITY;
(*
```

### 5.7.2.2 simple_type

The **simple_type** entity serves as a common supertype for the entities used to indicate a simple type of the associated DET.

EXPRESS specification:

```
*)
ENTITY simple_type
ABSTRACT SUPERTYPE OF ( ONEOF (
       number_type,
       boolean_type,
       string_type))
SUBTYPE OF(data_type);
       value_format: value_format_type;
END_ENTITY;
(*
```

Attribute definitions:

**value_format**[38]: the encoding of the format of values for properties.

### 5.7.2.3 number_type

The **number_type** entity provides for values of DETs that are of type NUMBER.

EXPRESS specification:

```
*)
ENTITY number_type
SUPERTYPE OF ( ONEOF (
       int_type,
       real_type))
SUBTYPE OF (simple_type);
END_ENTITY;
(*
```

### 5.7.2.4 int_type

The **int_type** entity provides for values of DETs that are of type INTEGER.

EXPRESS specification:

```
*)
ENTITY int_type
SUPERTYPE OF ( ONEOF (
       int_measure_type,
       int_currency_type,
       non_quantitative_int_type))
SUBTYPE OF (number_type);
END_ENTITY;
(*
```

[38] See IEC 61360-1 clause 3.4.1 .

### 5.7.2.5 int_measure_type

The **int_measure_type** entity provides for values of DETs that are measures of type INTEGER.

EXPRESS specification:

```
*)
ENTITY int_measure_type
SUBTYPE OF (int_type);
       unit: dic_unit;
END_ENTITY;
(*
```

Attribute definitions:

**unit**[39]: the unit associated to the described measure.

### 5.7.2.6 int_currency_type

The **int_currency_type** entity provides for values of DETs that are integer currencies.

EXPRESS specification:

```
*)
ENTITY int_currency_type
SUBTYPE OF (int_type);
       currency: OPTIONAL currency_code;
END_ENTITY;
(*
```

Attribute definitions:

**currency:** the associated code of the described currency according to ISO 4217. If not present, the currency code has to be exchanged together with the data (values).

### 5.7.2.7 non_quantitative_int_type

The **non_quantitative_int_type** entity is an enumeration type where elements of the enumeration are represented with an INTEGER value (see also ENTITY **non_quantitative_code_type** and figure 11.).

EXPRESS specification:

```
*)
ENTITY non_quantitative_int_type
SUBTYPE OF (int_type);
       domain: value_domain;
WHERE
       WR1: QUERY (v <* domain.its_values |
             'ISO13584_IEC61360_DICTIONARY_SCHEMA.VALUE_CODE_TYPE' IN
       TYPEOF ( v.value_code )) = [ ];
END_ENTITY;
(*
```

Attribute definitions:

**domain:** the set of enumerated values described in the **value_domain** entity.

[39] See IEC 61360-1 clause 3.4.3 .

Formal propositions:

**WR1:** The values associated with the **domain.its_values** list shall not contain a **value_code_type**.

### 5.7.2.8 real_type

The **real_type** entity provides for values of DETs that are of type REAL.

EXPRESS specification:

```
*)
ENTITY real_type
SUPERTYPE OF ( ONEOF (
       real_measure_type,
       real_currency_type))
SUBTYPE OF (number_type);
END_ENTITY;
(*
```

### 5.7.2.9 real_measure_type

The **real_measure_type** entity provides for values of DETs that are measures of type REAL.

EXPRESS specification:

```
*)
ENTITY real_measure_type
SUBTYPE OF (real_type);
       unit: dic_unit;
END_ENTITY;
(*
```

Attribute definitions:

**unit**[40]: the unit associated to the described measure.

### 5.7.2.10 real_currency_type

The **real_currency_type** entity defines real currencies.

EXPRESS specification:

```
*)
ENTITY real_currency_type
SUBTYPE OF (real_type);
       currency: OPTIONAL currency_code;
END_ENTITY;
(*
```

Attribute definitions:

**currency:** the associated code of the described currency according to ISO 4217. If not present, the currency code has to be exchanged together with the data (values).

[40] See IEC 61360-1 clause 3.4.3 .

### 5.7.2.11 boolean_type

The **boolean_type** entity provides for values of DETs that are of type BOOLEAN.

EXPRESS specification:

```
*)
ENTITY boolean_type
SUBTYPE OF (simple_type);
END_ENTITY;
(*
```

### 5.7.2.12 string_type

The **string_type** provides for values of DETs that are of type STRING.

EXPRESS specification:

```
*)
ENTITY string_type
SUBTYPE OF (simple_type);
END_ENTITY;
(*
```

### 5.7.2.13 non_quantitative_code_type

The **non_quantitative_code_type** entity is an enumeration type where elements of the enumeration are represented with a STRING value (see also ENTITY **non_quantitative_int_type** and figure 11).

EXPRESS specification:

```
*)
ENTITY non_quantitative_code_type
SUBTYPE OF (string_type);
       domain: value_domain;
WHERE
       WR1: QUERY (v <* domain.its_values |
             NOT ('ISO13584_IEC61360_DICTIONARY_SCHEMA.VALUE_CODE_TYPE' IN
             TYPEOF ( v.value_code ))) = [ ];
END_ENTITY;
(*
```

Attribute definitions:

**domain:** the set of enumerated values described in the **value_domain** entity.

Formal propositions:

**WR1:** The values associated with the **domain.its_values** list shall only contain elements of type **value_code_type**.

### 5.7.2.14 complex_type

The **complex_type** entity provides for the definition of types of which the values are represented as EXPRESS instances.

EXPRESS specification:

```
*)
ENTITY complex_type
ABSTRACT SUPERTYPE OF ( ONEOF (
             level_type,
             class_instance_type,
             entity_instance_type))
SUBTYPE OF(data_type );
END_ENTITY;
(*
```

### 5.7.2.15 level_type[41]

The **level_type** entity provides an indicator to qualify the values of a quantitative data element type.

EXPRESS specification:

```
*)
ENTITY level_type
SUBTYPE OF (complex_type);
       levels: LIST [ 1: 4 ] OF UNIQUE level;
       value_type: simple_type;
WHERE
       WR1: 'ISO13584_IEC61360_DICTIONARY_SCHEMA.NUMBER_TYPE'
             IN TYPEOF ( value_type );
END_ENTITY;
(*
```

Attribute definitions:

**levels:** the list of unique elements that specifies which qualified values shall be associated with the property.

**value_type:** the type of value of the different qualified values.

Formal propositions:

**WR1:** The **SELF.value_type** shall be of type **number_type**

### 5.7.2.16 level

The **level** type provides an abbreviated name of the different qualified values that may be associated with a physical quantity, distinguishing it from other possible or allowed values of the same quantity.

EXPRESS specification:

```
*)
TYPE level = ENUMERATION OF (
       min,  (*corresponds to the minimum value of the physical quantity*)
       nom,  (*corresponds to the nominal value of the physical quantity*)
       typ,  (*corresponds to the typical value of the physical quantity*)
       max); (*corresponds to the maximal value of the physical quantity*)
END_TYPE;
(*
```

### 5.7.2.17 class_instance_type

The **class_instance_type** entity provides for values of DETs that are represented as instances of a **class**. It is used, in particular, for the description of assemblies or to describe the material a (part of a) component consists of.

EXPRESS specification:

```
*)
ENTITY class_instance_type
SUBTYPE OF (complex_type);
       domain: class_BSU;
END_ENTITY;
(*
```

[41] See IEC 61360-1 clause 3.4.5 .

Attribute definitions:

**domain:** the **class_BSU** referring to the **class** representing the described type.

#### 5.7.2.18 entity_instance_type

The **entity_instance_type** entity provides for values of DETs that are represented as instances of some EXPRESS entity data types. A **type_name** attribute enables the specification to know what are the allowed data types. This attribute, together with the EXPRESS TYPEOF function applied to the value, permits strong type checking and polymorphism. This entity will be subtyped below for some data types that are allowed for use in the dictionary schema.

EXPRESS specification:

```
*)
ENTITY entity_instance_type
ABSTRACT SUPERTYPE
SUBTYPE OF (complex_type);
       type_name: SET OF STRING;
END_ENTITY;
(*
```

Attribute definitions:

**type_name:** the set of strings that describe, in the format of the EXPRESS TYPEOF function, the EXPRESS entity data type names that shall belong to the result of the EXPRESS TYPEOF function when
it is applied to a value that references the present entity as its data type.

#### 5.7.2.19 placement_type

The **placement_type** entity provides for values of DETs that are instances of **placement** entity data type. (See ISO 10303-42 for details).

EXPRESS specification:

```
*)
ENTITY placement_type
SUPERTYPE OF ( ONEOF (
       axis1_placement_type
       axis2_placement_2d_type,
       axis2_placement_3d_type))
SUBTYPE OF (entity_instance_type);
WHERE
       WR1: 'GEOMETRY_SCHEMA.PLACEMENT'
             IN SELF\entity_instance_type.type_name;
END_ENTITY;
(*
```

Formal propositions:

**WR1:** The string 'GEOMETRY_SCHEMA.PLACEMENT' shall be contained in the set defined by the **SELF\entity_instance_type.type_name** attribute.

#### 5.7.2.20 axis1_placement_type

The **axis1_placement_type** entity provides for values of DETs that are instances of **axis1_placement** entity data type (see ISO 10303-42 for details).

EXPRESS specification:

```
*)
ENTITY axis1_placement_type
SUBTYPE OF (placement_type);
```

```
WHERE
       WR1: 'GEOMETRY_SCHEMA.AXIS1_PLACEMENT'
            IN SELF\entity_instance_type.type_name;

END_ENTITY;
(*
```

Formal propositions:

**WR1:** The string 'GEOMETRY_SCHEMA.AXIS1_PLACEMENT' shall be contained in the set defined for the **SELF\entity_instance_type.type_name** attribute.

### 5.7.2.21 axis2_placement_2d_type

The **axis2_placement_2d_type** entity provides for values of DETs that are instances of **axis2_placement_2d** entity data type (see ISO 10303-42 for details).

EXPRESS specification:

```
*)
ENTITY axis2_placement_2d_type
SUBTYPE OF (placement_type);
WHERE
       WR1: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_2D'
            IN SELF\entity_instance_type.type_name;
END_ENTITY;
(*
```

Formal propositions:

**WR1:** The string 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_2D' shall be contained in the set defined for the **SELF\entity_instance_type.type_name** attribute.

### 5.7.2.22 axis2_placement_3d_type

The **axis2_placement_3d_type** entity provides for values of DETs that are instances of **axis2_placement_3d** entity data type (see ISO 10303-42 for details).

EXPRESS specification:

```
*)
ENTITY axis2_placement_3d_type
SUBTYPE OF (placement_type);
WHERE
       WR1: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_3D'
            IN SELF\entity_instance_type.type_name;
END_ENTITY;
(*
```

Formal propositions:

**WR1:** The string 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_3D' shall be contained in the set defined for the **SELF\entity_instance_type.type_name** attribute.

### 5.7.2.23 named_type

The **named_type** entity provides for referring to other types via the BSU mechanism.

EXPRESS specification:

```
*)
ENTITY named_type
SUBTYPE OF (data_type );
       referred_type: data_type_BSU;
END_ENTITY;
(*
```

Attribute definitions:

**referred_type:** the BSU identifying the **data_type** the present entity refers to.

### 5.7.3 Values

This clause contains definitions for non-quantitative data element types (see entity **non_quantitative_int_type** and entity **non_quantitative_code_type**).
Figure 11 outlines, as a planning model, the data associated with non-quantitative data element types.

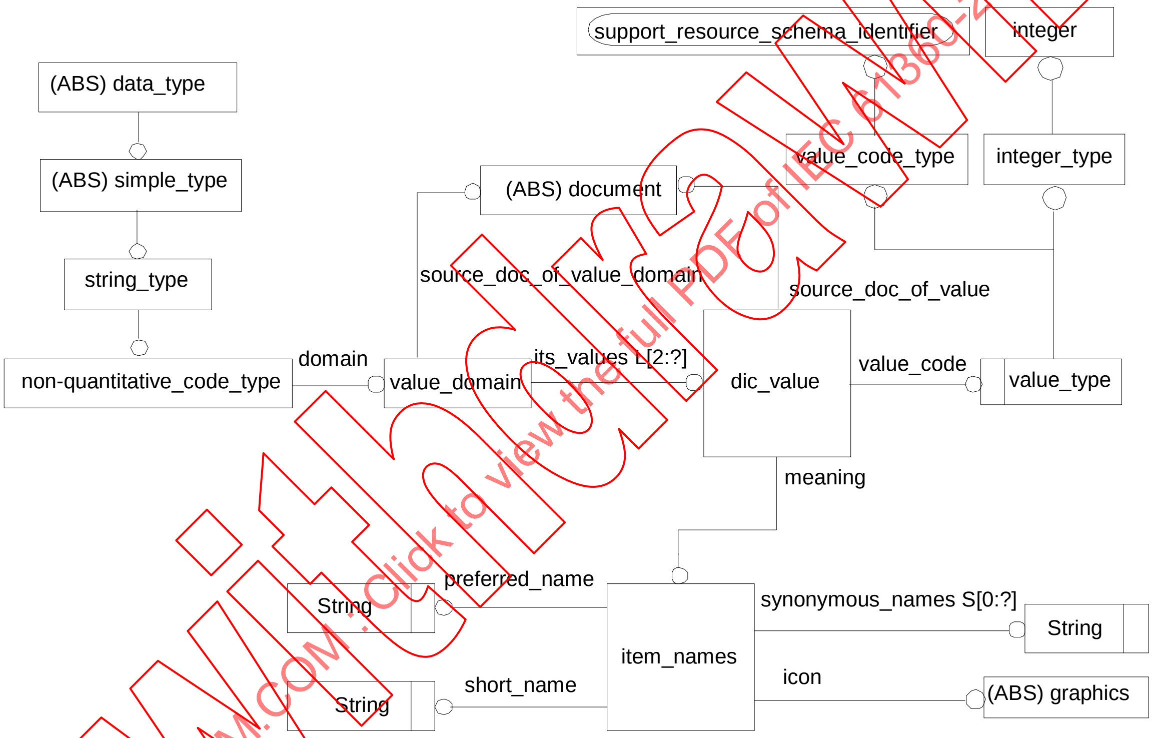

**Figure 11 - Overview of non-Quantitative data element types**

#### 5.7.3.1 value_domain[42]

The **value_domain** entity describes the set of allowed values for a non-quantitative data element type.

EXPRESS specification:

```
*)
ENTITY value_domain;
       its_values: LIST [ 2: ? ] OF dic_value;
       source_doc_of_value_domain: OPTIONAL document;
       languages: OPTIONAL present_translations;
       terms: LIST [ 0: ? ] OF item_names;
```

[42] See IEC 61360-1 clause 3.4.2 .

```
WHERE
       WR1: NOT EXISTS ( languages ) XOR ( QUERY ( v <* its_values |
             languages:<>: v.meaning.languages ) = [ ]);
       WR2: codes_are_unique (its_values);
END_ENTITY;
(*
```

Attribute definitions:

**its_values:** the enumeration list of values contained in the described domain.

**source_doc_of_value_domain:** the document describing the domain associated to the described **value_domain** entity.

**languages:** the optional list of languages in which the translations are provided.

**terms:** the optional list of **item_names** to allow for IEC 61360 the link to the terms dictionary.

Formal propositions:

**WR1:** If the value meanings are provided in more than one language, then the set of languages used must be the same for the whole set of values.

**WR2:** Value codes must be unique within this data type.

### 5.7.3.2 value_code

Each value of a non-quantitative data element is associated with a code, that characterizes the value. A **value_code** may be either an INTEGER or a **value_code_type**.

EXPRESS specification:

```
*)
TYPE integer_type = INTEGER; END_TYPE;
TYPE value_type = SELECT (value_code_type, integer_type); END_TYPE;
(*
```

### 5.7.3.3 dic_value

The **dic_value**[43] entity is one of the values of a **value_domain** entity.

EXPRESS specification:

```
*)
ENTITY dic_value;
       value_code: value_type;
       meaning: item_names;
       source_doc_of_value: OPTIONAL document;
END_ENTITY;
(*
```

Attribute definitions:

**value_code**[44]: the code associated to the described value. It can be either a **value_code_type** or an INTEGER.

**meaning**[45]: the meaning associated to this value. It is provided by names.

[43] See IEC 61360-1 clause 3.4.6 .

[44] See IEC 61360-1 clause 3.4.7 .

[45] See IEC 61360-1 clause 3.4.8 .

**source_doc_of_value**[46]: the optional source document in which the value is defined.

### 5.7.4 Extension to ISO 10303-41 unit definitions

This clause defines the resources for description of units in a dictionary. It extends the resources defined in ISO 10303-41.

#### 5.7.4.1 non_si_unit

The **non_si_unit** entity extends the unit model of ISO 10303-41 to allow for the representation of non-SI-units which are neither context dependent, nor conversion-based (see ISO 10303-41 for details).

EXPRESS specification:

```
*)
ENTITY non_si_unit
SUBTYPE OF (named_unit);
       name: label;
END_ENTITY;
(*
```

Attribute definitions:

**name:** the **label** used to name the described unit.

#### 5.7.4.2 assert_ONEOF rule

The **assert_ONEOF** rule asserts that ONEOF holds between the following subtypes of **named_unit**: **si_unit, context_dependent_unit, conversion_based_unit, non_si_unit**.

EXPRESS specification:

```
*)
RULE assert_ONEOF FOR (named_unit);
WHERE
       QUERY (u <* named_unit |
             ('ISO13584_IEC61360_DICTIONARY_SCHEMA.NON_SI_UNIT'
             IN TYPEOF(u)) AND
             ('MEASURE_SCHEMA.SI_UNIT' IN TYPEOF(u))
             OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.NON_SI_UNIT'
             IN TYPEOF(u)) AND
             ('MEASURE_SCHEMA.CONTEXT_DEPENDENT_UNIT' IN TYPEOF(u))
             OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.NON_SI_UNIT'
             IN TYPEOF(u)) AND
             ('MEASURE_SCHEMA.CONVERSION_BASED_UNIT' IN TYPEOF(u))) = [ ];
END_RULE;
(*
```

#### 5.7.4.3 dic_unit

The basic representation of units is in structured form according to ISO 10303-41. But since one of the purposes of storing units in the dictionary is for the presentation to the user, a structured representation alone is not sufficient, it must be supplemented by a string representation. The present definitions allow various possibilities:

— the function **string_for_unit** (see clause 5.9 "Function definitions") can be used. For a given structured representation of a unit it returns a string representation corresponding to the one used in Annex B of IEC 61360-1;

[46] See IEC 61360-1 clause 3.4.4 .

— a string representation can be supplied in plain text form (entity **mathematical_string,** attribute **text_representation**);

— an SGML representation can be supplied to allow for an enhanced presentation of the unit including sub- and superscripts etc. (entity **mathematical_string,** attribute **SGML_representation**).

The **dic_unit** entity describes a unit to be stored in a dictionary.

EXPRESS specification:

```
*)
ENTITY dic_unit;
       structured_representation: unit;
       string_representation: OPTIONAL mathematical_string;
END_ENTITY;
(*
```

Attribute definitions:

**structured_representation:** structured representation, from ISO 10303-41, including extension defined in clause 5.7.4 "Extension to ISO 10303-41 definitions".

**string_representation:** the function **string_for_unit** can be used to compute a string representation from the **structured_representation,** for the case where no **string_representation** is present.

## 5.8 Basic type and entity definitions

This subclause contains the basic type and entity definitions which were used in the main part of the model.

### 5.8.1 Basic type definitions

This subclause contains the basic type and entity definitions, sorted alphabetically.

#### 5.8.1.1 class_code_type

The **class_code_type** identifies the allowed values for a class code.

EXPRESS specification:

```
*)
TYPE class_code_type = code_type;
WHERE
       WR1: LENGTH(SELF) <= class_code_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of values corresponding to **class_code_type** shall be less than or equal to the length of **class_code_len** (i.e. 14).

#### 5.8.1.2 code_type

The **code_type** identifies the allowed values for a code type.

EXPRESS specification:

```
*)
TYPE code_type = identifier;
WHERE
       WR1: NOT (SELF LIKE '*.*';
       WR2: NOT (SELF LIKE '*-*');
       WR3: NOT( SELF LIKE '* *');
END_TYPE;
(*
```

Formal propositions:

**WR1:** The '.' shall not be contained in a **code_type** value. '.' is used to concatenate identifiers (see: CONSTANT **sep_id**).

**WR2:** The '-' shall not be contained in a **code_type** value. '-' is used to compute the identifier as the concatenation of code and version (see: CONSTANT **sep_cv**).

**WR3:** spaces are not allowed, to avoid problems with leading and trailing blanks when concatenating codes.

### 5.8.1.3 currency_code

The **currency_code** identifies the values allowed for a currency code.

These values are defined according to ISO 4217. Values are e.g. "CHF" for Swiss Francs, "CNY" for Yuan Renminbi (Chinese), "DEM" for German Mark, "FRF" for French Francs, "JPY" for Yen (Japanese), "SUR" for SU Rouble, "USD" for US Dollars, "XEU" for ECU (European Community Unit), etc...

EXPRESS specification:

```
*)
TYPE currency_code = identifier;
WHERE
       WR1: LENGTH(SELF) = 3;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **currency_code** value shall be equal to 3.

### 5.8.1.4 date_type

The **date_type** identifies the values allowed for a date. These values are defined according to ISO 8601 (e.g. "1994-03-21").

EXPRESS specification:

```
*)
TYPE date_type = STRING(10) FIXED;
END_TYPE;
(*
```

### 5.8.1.5 definition_type

The **definition_type** identifies the values allowed for a definition.

EXPRESS specification:

```
*)
TYPE definition_type = translatable_text;
END_TYPE;
(*
```

### 5.8.1.6 DET_classification_type

The **DET_classification_type** identifies the values allowed for the DET classification. These values are used for DET classification according to ISO 31.

EXPRESS specification:

```
*)
TYPE DET_classification_type = identifier;
WHERE
       WR1: LENGTH(SELF) = DET_classification_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **DET_classification_type** value shall be equal to the value of a **DET_classification_len** (i.e. 3).

### 5.8.1.7 data_type_code_type

The **data_type_code_type** identifies the values allowed for a data type code.

EXPRESS specification:

```
*)
TYPE data_type_code_type = code_type;
WHERE
       WR1: LENGTH(SELF) <= data_type_code_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **data_type_code_type** value shall be less than or equal to the value of a **data_type_code_len** (i.e. 14).

### 5.8.1.8 note_type

The **note_type** identifies the values allowed for a note.

EXPRESS specification:

```
*)
TYPE note_type = translatable_text;
END_TYPE;
(*
```

### 5.8.1.9 pref_name_type

The **pref_name_type** identifies the values allowed for a preferred name.

EXPRESS specification:

```
*)
TYPE pref_name_type = translatable_label;
WHERE
       WR1: check_label_length (SELF, pref_name_len);
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **pref_name_type** value shall not exceed the length of **pref_name_len** (i.e. 30).

### 5.8.1.10 property_code_type

The **property_code_type** identifies the values allowed for a property code.

EXPRESS specification:

```
*)
TYPE property_code_type = code_type;
WHERE
       WR1: LENGTH(SELF) <= property_code_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **property_code_type** value shall be less than or equal to the value of a **property_code_len** (i.e. 14).

### 5.8.1.11 remark_type

The **remark_type** identifies the values allowed for a remark.

EXPRESS specification:

```
*)
TYPE remark_type = translatable_text;
END_TYPE;
(*
```

### 5.8.1.12 revision_type

The **revision_type** identifies the values allowed for a revision.

EXPRESS specification:

```
*)
TYPE revision_type = code_type;
WHERE
       WR1: LENGTH(SELF) <= revision_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **revision_type** value shall not exceed the length of **revision_len** (i.e. 3).

### 5.8.1.13 short_name_type

The **short_name_type** identifies the values allowed for a short name.

EXPRESS specification:

```
*)
TYPE short_name_type = translatable_label;
WHERE
       WR1: check_label_length (SELF, short_name_len);
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **short_name_type** value shall not exceed the length of **short_name_len** (i.e. 15).

### 5.8.1.14 supplier_code_type

The **supplier_code_type** identifies the values allowed for a supplier code.

EXPRESS specification:

```
*)
TYPE supplier_code_type = code_type;
WHERE
       WR1: LENGTH(SELF) <= supplier_code_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **supplier_code_type** value shall be less than or equal to the value of the **supplier_code_len** (i.e. 18).

### 5.8.1.15 syn_name_type

The **syn_name_type** identifies the values allowed for a synonymous name.

EXPRESS specification:

```
*)
TYPE syn_name_type = SELECT (label_with_language, label);
WHERE
       WR1: check_syn_length (SELF, syn_name_len);
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **syn_name_type** value shall be equal to the value of a = **syn_name_len** (i.e. 30).

### 5.8.1.16 value_code_type

The **value_code_type** identifies the values allowed for a value code.

EXPRESS specification:

```
*)
TYPE value_code_type = identifier;
WHERE
       WR1: LENGTH(SELF) <= value_code_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **value_code_type** value shall not exceed the length of **value_code_len** (i.e. 18).

#### 5.8.1.17 value_format_type

The **value_format_type** identifies the values allowed for a value format. These values are defined according to ISO 6093 and ISO 9735.

EXPRESS specification:

```
*)
TYPE value_format_type = identifier;
WHERE
       WR1: LENGTH(SELF) <= value_format_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **value_format_type** value shall not exceed the length of **value_format_len** (i.e. 80).

#### 5.8.1.18 version_type

The **version_type** identifies the values allowed for a version.

EXPRESS specification:

```
*)
TYPE version_type = code_type;
WHERE
       WR1: LENGTH(SELF) = version_len;
       WR2:SELF LIKE '###';
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **version_type** value shall be equal to **version_len** (i.e. 3).

**WR2:** The **version_type** shall contain digits only.

#### 5.8.1.19 source_doc_type

The **source_doc_type** identifies the values allowed for a source document.

EXPRESS specification:

```
*)
TYPE source_doc_type = identifier;
WHERE
       WR1: LENGTH(SELF) <= source_doc_len;
END_TYPE;
(*
```

Formal propositions:

**WR1:** The length of a **source_doc_type** value shall not exceed the length of **source_doc_len** (i.e. 80).

### 5.8.2 Basic entity definitions

This subclause contains the basic entity definitions, sorted alphabetically.

#### 5.8.2.1 dates

The **dates** entity describes the three dates associated respectively to the first version, the current version and the current revision for a given description.

EXPRESS specification:

```
*)
ENTITY dates;
       date_of_original_definition: date_type;
       date_of_current_version: date_type;
       date_of_current_revision: OPTIONAL date_type;
END_ENTITY;
(*
```

Attribute definitions:

**date_of_original_definition:** the date associated to version 001.

**date_of_current_version:** the date associated to the present version.

**date_of_current_revision:** the date associated to the last revision.

#### 5.8.2.2 document

The **document** entity is an abstract resource that stands for a document. The dictionary schema only provides for exchanging the identification of documents (see below). The **document** entity may also be subtyped with entities implementing a means for exchanging document data, e.g. by reference to an external file and exact specification of the format of the file.

EXPRESS specification:

```
*)
ENTITY document
ABSTRACT SUPERTYPE;
END_ENTITY;
(*
```

#### 5.8.2.3 graphics

The **graphics** entity is to be subtyped with entities implementing a means for exchanging graphical data, e.g. by reference to an external file and exact specification of the format of the file.

EXPRESS specification:

```
*)
ENTITY graphics
ABSTRACT SUPERTYPE;
END_ENTITY;
(*
```

#### 5.8.2.4 identified_document

The **identified_document** entity describes a document identified by its code.

EXPRESS specification:

```
*)
ENTITY identified_document
SUBTYPE OF (document);
       document_identifier: source_doc_type;
END_ENTITY;
(*
```

Attribute definitions:

**document_identifier:** the code of the described document.

### 5.8.2.5 item_names

The **item_names** entity identifies the names that can be associated to a given description. It states the preferred name, the set of synonymous names, the short name and the languages in which the different names are provided. It may be associated with an icon.

EXPRESS specification:

```
*)
ENTITY item_names;
       preferred_name: pref_name_type;
       synonymous_names: SET OF syn_name_type;
       short_name: short_name_type;
       languages: OPTIONAL present_translations;
       icon: OPTIONAL graphics;
WHERE
       WR1: NOT ( EXISTS ( languages ) XOR (
             ('ISO13584_IEC61360_LANGUAGE_RESOURCE_SCHEMA'
             +'.TRANSLATED_LABEL' IN TYPEOF (preferred_name))
             AND (languages:=: preferred_name\translated_label.languages)
             AND ('ISO13584_IEC61360_LANGUAGE_RESOURCE_SCHEMA'
             +'.TRANSLATED_LABEL' IN TYPEOF (short_name))
             AND (languages:=: short_name\translated_label.languages)
             AND (QUERY ( s <* synonymous_names |
             NOT ('ISO13584_IEC61360_DICTIONARY_SCHEMA.LABEL_WITH_LANGUAGE'
             IN TYPEOF (s))) = [ ])));
       WR2:  NOT EXISTS(languages) XOR (QUERY ( s <* synonymous_names |
             EXISTS(s.language) AND NOT (s.language IN
             QUERY ( l <* languages.language_codes | TRUE
             (* i.e. take all *) ))) = [ ]);
       WR3:  at_most_two_synonyms_per_language
             (languages, synonymous_names);
END_ENTITY;
(*
```

Attribute definitions:

**preferred_name**[47]: the name which is preferred for use.

**synonymous_names**[48]: the set of synonymous names.

**short_name**[49]: the abbreviation of the preferred name.

**languages:** the optional list of **languages** in which the different names are provided.

**icon**: an optional **icon** which graphically represents the description associated with the **item_names.**

[47] See IEC 61360-1 clauses 3.2.2 and 5.3.5 .

[48] See IEC 61360-1 clause 3.2.3 .

[49] See IEC 61360-1 clause 3.2.6 .

Formal propositions:

**WR1:** If preferred and short names are provided in more than one language, then all the "languages" attributes of the **translated_labels** must contain the **present_translations** instance as in the languages attribute of this **item_names** instance.

**WR2:** If synonymous names are provided in more than one language, then only languages indicated in the **present_translations** instance in the "languages" attribute of this **item_names** instance can be used.

**WR3:** At most two synonymous shall be defined in the **languages** attribute, if it exists.

#### 5.8.2.6 label_with_language

The **label_with_language** entity provides resources for associating a label to a language.

EXPRESS specification:

```
*)
ENTITY label_with_language;
       l: label;
       language: language_code;
END_ENTITY;
(*
```

Attribute definitions:

**l:** the label associated to a language.

**language:** the code of the labelled language.

#### 5.8.2.7 mathematical_string

The **mathematical_string** entity provides resources defining a representation for mathematical strings. It also allows a representation in the SGML format.

EXPRESS specification:

```
*)
ENTITY mathematical_string;
       text_representation: text;
       SGML_representation: OPTIONAL text;
END_ENTITY;
(*
```

Attribute definitions:

**text_representation:** "linear" form of a mathematical string, using ISO 843: “Information and documentation - Conversion of Greek characters into Latin characters", if necessary.

**SGML_representation:** SGML-Text (ISO 8879), marked up according to the Math DTD (Document Type Definition) in ISO 12083: "Electronic Manuscript Preparation and Markup". The SGML text must be processed so that it will be treated as one single string during the exchange (see ISO 10303-21).

## 5.9 Function definitions

This subclause contains functions which are referenced in WHERE clauses to assert data consistency, or which provide resources for application development.

### 5.9.1 acyclic_superclass_relationship function

The **acyclic_superclass_relationship** function checks that there is no cycle in the superclass relationship. By the cardinality of the **its_superclass** attribute in ENTITY class, it is ensured that there is an inheritance tree, no acyclic graph. Thus, this function merely has to check that no class instance refers in the **its_superclass** attribute to another one which is essentially a subclass.

EXPRESS specification:

```
*)
FUNCTION acyclic_superclass_relationship (
             current: class_BSU;
             visited: SET OF class): LOGICAL;
IF SIZEOF (current.definition) = 1 THEN
             IF current.definition[1]\class IN visited THEN
             RETURN (FALSE);
       (* wrong: current declares a subclass as its superclass *)
       ELSE
             IF EXISTS
             (current.definition[1]\class.its_superclass)
             THEN
                   RETURN (acyclic_superclass_relationship (
                   current.definition[1]\class.its_superclass,
                   visited + current.definition[1]\class));
             ELSE
                   RETURN (TRUE);
             END_IF;
       END_IF;
ELSE
       RETURN (UNKNOWN);
END_IF;
END_FUNCTION; -- acyclic_superclass_relationship
(*
```

### 5.9.2 at_most_two_synonyms_per_language function

The **at_most_two_synonyms_per_language** function checks that there are at most two synonymous names in the **synonymous_names** parameter that correspond to each language parameter.

EXPRESS specification:

```
*)
FUNCTION at_most_two_synonyms_per_language (
       languages: present_translations;
       synonymous_names: SET OF syn_name_type): BOOLEAN;
IF EXISTS (languages) THEN
       REPEAT i := 1 TO SIZEOF (languages.language_codes);
             IF SIZEOF( QUERY ( s <* synonymous_names |
             s.language = languages.language_codes[i] )) > 2
             THEN
                   RETURN (FALSE);
             END_IF;
       END_REPEAT;
       RETURN (TRUE);
ELSE
       RETURN (SIZEOF (synonymous_names) <= 2);
END_IF;
END_FUNCTION; -- at_most_two_synonyms_per_language
(*
```

### 5.9.3 check_syn_length function

The **check_syn_length** function checks that the length of **s** doesn't exceed the length indicated by **s_length**.

EXPRESS specification:

```
*)
FUNCTION check_syn_length (
             s: syn_name_type;
             s_length: INTEGER): BOOLEAN;
IF 'ISO13584_IEC61360_DICTIONARY_SCHEMA.LABEL_WITH_LANGUAGE'
       IN TYPEOF(s) THEN
       RETURN (LENGTH(s.l) <= s_length);
ELSE
       RETURN (LENGTH(s) <= s_length);
END_IF;
END_FUNCTION; -- check_syn_length
(*
```

### 5.9.4 codes_are_unique function

The **codes_are_unique** function returns TRUE if the **value_codes** are unique within this list of **values**.

EXPRESS specification:

```
*)
FUNCTION codes_are_unique (values: LIST OF dic_value): BOOLEAN;
LOCAL
       l: SET OF STRING:= [ ];
END_LOCAL;
REPEAT i:= 1 TO SIZEOF (values);
       l:= l + values[i].value_code;
       (* duplicates are eliminated. *)
END_REPEAT;
RETURN (SIZEOF(values) = SIZEOF(l));
END_FUNCTION; -- codes_are_unique
(*
```

### 5.9.5 definition_available_implies function

The **definition_available_implies** function checks whether the definition corresponding to the **BSU** parameter exists or not. Then, if this definition exists, the **expression** parameter is returned.

EXPRESS specification:

```
*)
FUNCTION definition_available_implies (
             BSU: basic_semantic_unit;
             expression: LOGICAL): LOGICAL;
       RETURN (NOT (SIZEOF(BSU.definition) = 1) OR expression);
END_FUNCTION; -- definition_available_implies
(*
```

### 5.9.6 is_subclass function

The function **is_subclass** returns TRUE if **sub** is defined as a subclass of **super**.

EXPRESS specification:

```
*)
FUNCTION is_subclass ( sub, super: class): LOGICAL;
IF (NOT EXISTS (sub)) OR (NOT EXISTS (super)) THEN RETURN (UNKNOWN);
END_IF;
IF NOT EXISTS (sub.its_superclass) THEN
       RETURN (FALSE);
       (* end of chain reached, didn't meet super so far *)
END_IF;
```

```
IF SIZEOF(sub.its_superclass.definition) = 1 THEN
       (* definition available *)
       IF (sub.its_superclass.definition[1]\class = super) THEN
             RETURN ( TRUE );
       ELSE
             RETURN (is_subclass (
             sub.its_superclass.definition[1]\class, super));
       END_IF;
ELSE RETURN (UNKNOWN);
END_IF;
END_FUNCTION; -- is_subclass
(*
```

### 5.9.7 string_for_derived_unit function

The function **string_for_derived_unit** returns a STRING representation of the **derived_unit** (according to ISO 10303-41) passed as parameter. First, the elements of the derived unit are separated according to the sign of the exponent. If there are elements of both kinds, the '/' notation is used to separate those with positive from those with negative sign. If there are only negative exponents, the u-e notation is used.
A dot '.' (decimal code 46 according to ISO 8859-1, see ISO 10303-21) is used to separate individual elements.

EXPRESS specification:

```
*)
FUNCTION string_for_derived_unit (u: derived_unit): STRING;

       FUNCTION string_for_derived_unit_element
             (u: derived_unit_element; neg_exp: BOOLEAN
             (* print negative exponents with power -1 *)): STRING;
             (* returns a STRING representation of the derived_unit_element
             (according to ISO 10303-41) passed as parameter *)
             LOCAL
                   result: STRING;
             END_LOCAL;
                   result:= string_for_named_unit(u.unit);
             IF (u.exponent <> 0) THEN
                   IF (u.exponent > 0) OR NOT neg_exp THEN
                         result:= result +'**'
                               + FORMAT (ABS(u.exponent), '1I');
                   ELSE
                         result:= result + '**'
                               + FORMAT (u.exponent, '1I');
                   END_IF;
             END_IF;
             RETURN (result);
       END_FUNCTION; -- string_for_derived_unit_element
       LOCAL
             pos, neg: SET OF derived_unit_element;
             us: STRING;
       END_LOCAL;
             (* separate unit elements according to the sign of the
             exponents: *)
       pos:= QUERY ( ue <* u.elements | ue.exponent > 0 );
       neg:= QUERY ( ue <* u.elements | ue.exponent < 0 );
       us:= '';
       IF SIZEOF (pos) > 0 THEN
             (* there are unit elements with positive sign *)
             REPEAT i:= LOINDEX (pos) TO HIINDEX (pos);
                   us:= us + string_for_derived_unit_element(pos[i], FALSE);
                   IF i <> HIINDEX (pos) THEN us:= us + '.';  END_IF;
             END_REPEAT;
             IF SIZEOF (neg) > 0 THEN
                   (* there are unit elements with negative sign, use '/'
                   notation: *)
                   us:= us + '/';
                   IF SIZEOF (neg) > 1 THEN us:= us + '('; END_IF;
                   REPEAT i:= LOINDEX (neg) TO HIINDEX (neg);
                         us:= us + string_for_derived_unit_element(neg[i],
                         FALSE);
                         IF i <> HIINDEX (neg) THEN us:= us + '.'; END_IF;
                   END_REPEAT;
                   IF SIZEOF (neg) > 1 THEN us:= us + ')'; END_IF;
             END_IF;
ELSE
             (* only negative signs, use u-e notation *)
             IF SIZEOF(neg) > 0 THEN
                   REPEAT i:= LOINDEX (neg) TO HIINDEX (neg);
                         us:= us + string_for_derived_unit_element(neg[i],
                         TRUE);
                         IF i <> HIINDEX (neg) THEN us:= us + '.'; END_IF;
                   END_REPEAT;
             END_IF;
END_IF;
RETURN (us);
END_FUNCTION; -- string_for_derived_unit
(*
```

### 5.9.8 string_for_named_unit function

The **string_for_named_unit** function returns a STRING representation of the **named_unit** (according to ISO 10303-41 and the extension in clause 5.7.4) passed as parameter.

EXPRESS specification:

```
*)
FUNCTION string_for_named_unit (u: named_unit): STRING;
IF 'MEASURE_SCHEMA.SI_UNIT' IN TYPEOF(u) THEN
       RETURN (string_for_SI_unit (u\si_unit));
ELSE
       IF 'MEASURE_SCHEMA.CONTEXT_DEPENDENT_UNIT' IN TYPEOF(u)
       THEN
             RETURN (u\context_dependent_unit.name);
       ELSE
             IF 'MEASURE_SCHEMA.CONVERSION_BASED_UNIT'
             IN TYPEOF(u)
             THEN
                   RETURN (u\conversion_based_unit.name);
             ELSE
                   IF 'ISO13584_IEC61360_DICTIONARY_SCHEMA'+'.NON_SI_UNIT'
                   IN TYPEOF(u)
                   THEN
                         RETURN (u\non_si_unit.name);
                   ELSE
                   (* pure named_unit instance, not subtyped further. *)
                         RETURN ('name_unknown');
                   END_IF;
             END_IF;
       END_IF;
END_IF;
END_FUNCTION; -- string_for_named_unit
(*
```

### 5.9.9 string_for_SI_unit function

The **string_for_SI_unit** function returns a STRING representation of the **si_unit** (according to ISO 10303-41) passed as parameter.

EXPRESS specification:

```
*)
FUNCTION string_for_SI_unit (unit: si_unit): STRING;
LOCAL
             prefix_string, unit_string: STRING;
END_LOCAL;
IF EXISTS (unit.prefix) THEN
       CASE unit.prefix OF
             exa        : prefix_string:= 'E';
             peta       : prefix_string:= 'P';
             tera       : prefix_string:= 'T';
             giga       : prefix_string:= 'G';
             mega       : prefix_string:= 'M';
             kilo       : prefix_string:= 'k';
             hecto      : prefix_string:= 'h';
             deca       : prefix_string:= 'da';
             deci       : prefix_string:= 'd';
             centi      : prefix_string:= 'c';
             milli      : prefix_string:= 'm';
             micro      : prefix_string:= 'u';
             nano       : prefix_string:= 'n';
             pico       : prefix_string:= 'p';
             femto      : prefix_string:= 'f';
             atto       : prefix_string:= 'a';
       END_CASE;
       ELSE
             prefix_string:= '';
       END_IF;
       CASE unit.name OF
             metre            : unit_string:= 'm';
             gram             : unit_string:= 'g';
             second           : unit_string:= 's';
             ampere           : unit_string:= 'A';
             kelvin           : unit_string:= 'K';
             mole             : unit_string:= 'mol';
             candela          : unit_string:= 'cd';
             radian           : unit_string:= 'rad';
             steradian        : unit_string:= 'sr';
             hertz            : unit_string:= 'Hz';
             newton           : unit_string:= 'N';
             pascal           : unit_string:= 'Pa';
             joule            : unit_string:= 'J';
             watt             : unit_string:= 'W';
             coulomb          : unit_string:= 'C';
             volt             : unit_string:= 'V';
             farad            : unit_string:= 'F';
             ohm              : unit_string:= 'Ohm';
             siemens          : unit_string:= 'S';
             weber            : unit_string:= 'Wb';
             tesla            : unit_string:= 'T';
             henry      : unit_string:= 'H';
             degree_Celsius   : unit_string:= 'Cel';
             lumen            : unit_string:= 'lm';
             lux              : unit_string:= 'lx';
             becquerel        : unit_string:= 'Bq';
             gray             : unit_string:= 'Gy';
             sievert          : unit_string:= 'Sv';
       END_CASE;
       RETURN (prefix_string + unit_string);
END_FUNCTION; -- string_for_SI_unit
(*
```

### 5.9.10 string_for_unit function

The **string_for_unit** function returns a STRING representation of the **unit** (according to ISO 10303-41) passed as parameter.

EXPRESS specification:

```
*)
FUNCTION string_for_unit (u: unit): STRING;
       IF 'MEASURE_SCHEMA.DERIVED_UNIT' IN TYPEOF(u) THEN
       RETURN (string_for_derived_unit(u));
       ELSE (* 'MEASURE_SCHEMA.NAMED_UNIT' IN TYPEOF(u) holds true *)
       RETURN (string_for_named_unit(u));
       END_IF;
END_FUNCTION; -- string_for_unit
(*
```

### 5.9.11 all_class_descriptions_reachable function

The **all_class_descriptions_reachable** function checks if the **dictionary_elements** that describe a class, referred by a **class_BSU,** and all its super-classes, can be computed in the inheritance tree defined by the class hierarchy.

EXPRESS specification:

```
*)
FUNCTION all_class_descriptions_reachable (cl:class_BSU): BOOLEAN;
IF SIZEOF(cl.definition) = 0
THEN
       RETURN(FALSE);
END_IF;
IF NOT (EXISTS(cl.definition[1]\class.its_superclass))
THEN
       RETURN (TRUE);
ELSE
       RETURN(all_class_descriptions_reachable(
       cl.definition[1]\class.its_superclass));
END_IF;
END_FUNCTION; -- all_class_descriptions_reachable
(*
```

### 5.9.12 compute_known_visible_properties function

The **compute_known_visible_properties** function computes the set of properties that are visible for a given class. When a definition is not available, it returns only the visible properties that may be computed.

EXPRESS specification:

```
*)
FUNCTION compute_known_visible_properties (cl: class_BSU):
       SET OF property_BSU;
LOCAL
s: SET OF property_BSU:=[ ];
END_LOCAL;
s:= USEDIN(cl,
       'ISO13584_IEC61360_DICTIONARY_SCHEMA.PROPERTY_BSU.NAME_SCOPE');
IF SIZEOF(cl.definition)=0
THEN
       RETURN(s);
```

```
ELSE
       IF EXISTS (cl.definition[1]\class.its_superclass) THEN
             s:= s + compute_known_visible_properties(
                   cl.definition[1]\class.its_superclass);
       END_IF;
       RETURN(s);
END_IF;
END_FUNCTION;
(*
```

### 5.9.13 compute_known_visible_data_types function

The **compute_known_visible_data_types** function computes the set of data_types that are visible for a given class. When a definition is not available, it returns only the visible data_types that may be computed.

EXPRESS specification:

```
*)
FUNCTION compute_known_visible_data_types (cl: class_BSU):
       SET OF data_type_BSU;
LOCAL
       s: SET OF data_type_BSU:=[ ];
END_LOCAL;
s:= USEDIN(cl,
       'ISO13584_IEC61360_DICTIONARY_SCHEMA.DATA_TYPE_BSU.NAME_SCOPE');
IF SIZEOF(cl.definition)=0
THEN
       RETURN(s);
ELSE
       IF EXISTS (cl.definition[1]\class.its_superclass) THEN
             s:= s + compute_known_visible_data_types(
                   cl.definition[1]\class.its_superclass);
       END_IF;
       RETURN(s);
END_IF;
END_FUNCTION;
(*
```

### 5.9.14 compute_known_applicable_properties function

The **compute_known_applicable_properties** function computes the set of properties that are applicable for a given class. When a definition is not available, it returns only the applicable properties that may be computed.

EXPRESS specification:

```
*)
FUNCTION compute_known_applicable_properties (cl: class_BSU):
       SET OF property_BSU;
LOCAL
       s: SET OF property_BSU:=[ ];
END_LOCAL;
IF SIZEOF(cl.definition)=0
THEN
       RETURN(s);
ELSE
       REPEAT i:=1 TO SIZEOF(cl.definition[1]\class.described_by);
             s:= s + cl.definition[1]\class.described_by[i];
       END_REPEAT;
       IF EXISTS (cl.definition[1]\class.its_superclass) THEN
             s:= s + compute_known_applicable_properties(
                   cl.definition[1]\class.its_superclass);
       END_IF;
```

```
        RETURN(s);
END_IF;
END_FUNCTION;
(*
```

### 5.9.15 compute_known_applicable_data_types function

The **compute_known_applicable_data_types** function computes the set of **data_types** that are applicable for a given class. When a definition is not available, it returns only the applicable **data_types** that may be computed.

EXPRESS specification:

```
*)
FUNCTION compute_known_applicable_data_types (cl: class_BSU):
        SET OF data_type_BSU;
LOCAL
        s: SET OF data_type_BSU:=[ ];
END_LOCAL;
IF SIZEOF(cl.definition)=0
THEN
        RETURN(s);
ELSE
        REPEAT i:=1 TO SIZEOF(cl.definition[1]\class.described_by);
              s:=s+cl.definition[1]\class.defined_types[i];
        END_REPEAT;
        IF EXISTS (cl.definition[1]\class.its_superclass) THEN
              s:=s+compute_known_applicable_data_types(
                    cl.definition[1]\class.its_superclass);
        END_IF;
        RETURN(s);
END_IF;
END_FUNCTION;
(*
```

### 5.9.16 list_to_set

The **list_to_set** function creates a SET from a LIST named **l**; the type of element for the SET will be the same as that in the original LIST.

EXPRESS specification:

```
*)
FUNCTION list_to_set(l: LIST [0:?] OF GENERIC:type_elem)
        : SET OF GENERIC: type_elem;
LOCAL
        s: SET OF GENERIC: type_elem:= [];
END_LOCAL;
REPEAT i:= 1 TO SIZEOF(l);
        s:= s + l[i];
END_REPEAT;
RETURN(s);
END_FUNCTION; -- list_to_set


END_SCHEMA; -- ISO13584_IEC61360_dictionary_schema
(*
```

## 6 IEC 61360 extended dictionary schema

This clause contains an IEC specific schema which contains additional provisions defined by IEC 61360-1 for modelling and structuring the terms that are involved in the definitions of data element types and associated classification scheme, component classes and terms.

This schema defines conformance class 2 of this part of IEC 61360. This conformance class includes conformance class 1 which is compatible with ISO 13584-42.
This schema uses all the resources defined in the ISO13584_IEC61360_dictionary_schema. It is therefore a superset of this schema.
It also contains references to other EXPRESS schemata which are used in the **IEC61360_extended_dictionary_schema**. Their source can be found in ISO 10303-41: STEP Part 41: "Fundamentals of Product Description and Support".

EXPRESS specification

```
*)
SCHEMA  IEC61360_extended_dictionary_schema;
USE FROM ISO13584_IEC61360_dictionary_schema;
REFERENCE FROM support_resource_schema (label);
(*
```

The following EXPRESS specification contains definitions for the representation of a **term.** A **term** is a conventional symbol for a concept, consisting of a word or a phrase. **Terms** in the context of IEC61360 define the **value_meaning** of a **value** of classifying data element types or define other words that may be ambiguous in understanding.

EXPRESS specification

```
*)
ENTITY term
SUBTYPE OF (item_names);
       abbreviated_names: SET OF value_code_type;
       identifier: label;
       definition: label;
       note: OPTIONAL label;
       remark: OPTIONAL label;
       figure: OPTIONAL graphics;
       formula: OPTIONAL mathematical_string;
       source_doc_of_definition: OPTIONAL document;
       related_terms: SET OF term;
       used_in_definition: SET OF class_and_property_elements;
WHERE
       WR1: NOT( identifier LIKE '* *');
END_ENTITY;
(*
```

Attribute definitions:

**abbreviated_names**[50]: representation of a term in a coded form.

**identifier**[51]: combination of characters used to identify a term uniquely within a term dictionary.

[50] See IEC 61360-1 clause 6.2.3 .

[51] See IEC 61360-1 clause 6.2.4 .

**definition**[52]: statement that describes the meaning of a term and permits its differentiation from all other terms.

**note**[53]: statement which provides further information on any part of the terminological record, which is essential to the understanding of that record.

**remark**[54]: explanatory text to further clarify the meaning of a terminological record.

**figure**[55]: illustration added to the definition of a term to clarify the meaning of that definition.

**formula**[56]: rule or statement in mathematical form expressing the semantics of a term.

**source_doc_of_definition**[57]: reference to the source document, generally an international standard, from which the term definition was derived.

**related_terms**[58]: term having some relation with the term under consideration.

**used_in_definition**: term used to make the definition of a data elemen type unambiguous.

<u>Formal propositions:</u>

**WR1:** The term identifier shall not contain any space.

```
*)
END_SCHEMA; -- IEC61360_extended_dictionary_schema
(*
```

## 7 ISO13584_IEC61360_language_resource_schema

The following schema provides resources for permitting strings in various languages. It has been extracted from the Dictionary schema, since it could be used in other schemata. It is largely based on the **support_resource_schema** from ISO 10303-41: STEP part 41: "Fundamentals of Product Description and Support", and can be seen as an extension to that. It allows for the usage of one specific language throughout an exchange context (Physical File) without the overhead introduced when multiple languages are used. See figure 12 - EXPRESS-G diagram of **ISO13584_IEC61360_language_resource_schema** and **support_resource_schema,** for a graphical depiction.

[52] See IEC 61360-1 clause 6.3.1 .

[53] See IEC 61360-1 clause 6.3.2 .

[54] See IEC 61360-1 clause 6.3.3 .

[55] See IEC 61360-1 clause 6.3.4 .

[56] See IEC 61360-1 clause 6.3.5 .

[57] See IEC 61360-1 clause 6.3.6 .

[58] See IEC 61360-1 clause 6.4.1 .

EXPRESS specification:

```
*)
SCHEMA ISO13584_IEC61360_language_resource_schema;

REFERENCE FROM support_resource_schema (identifier, label, text);

       (* from ISO 10303-41: STEP part 41: "Fundamentals of Product
       Description and Support" *)
(*
```

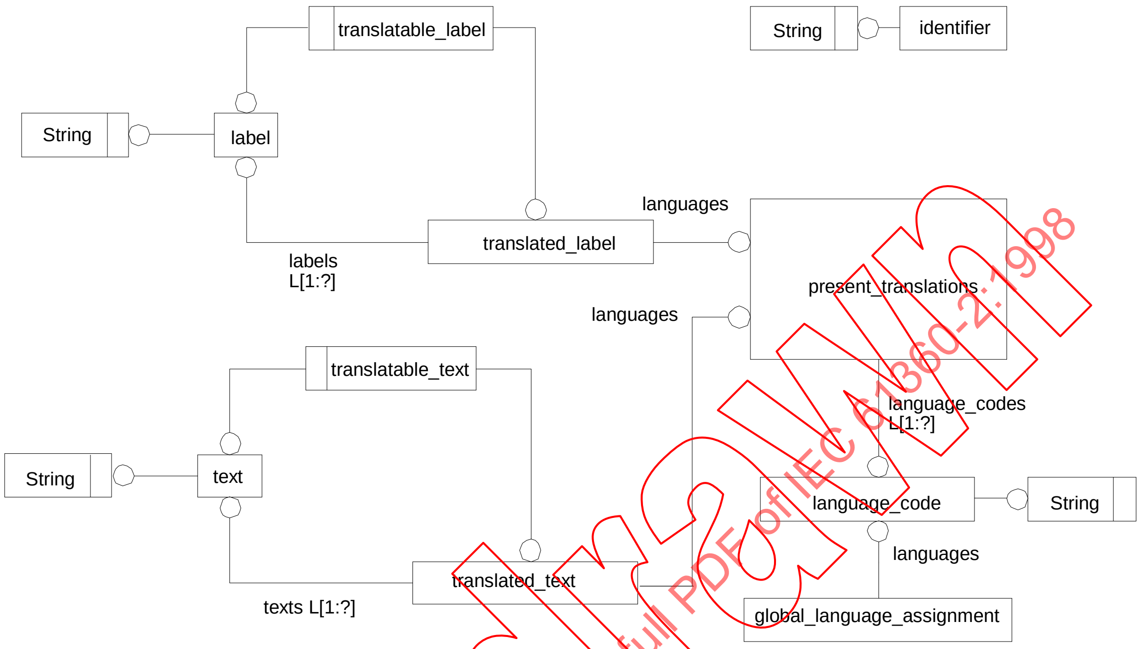

**Figure 12 - EXPRESS-G diagram of ISO13584_IEC61360_language_resource_schema and support_resource_schema**

## 7.1 ISO13584_IEC61360_language_resource_schema type and entity definitions

This subclause contains the EXPRESS type and entity definitions in the **ISO13584_IEC61360_language_resource_schema.**

### 7.1.1 language_code

The **language_code** type enables to identify a language according to ISO 639. Values are e.g. "EN" for English in general, "FR" for French, "RU" for Russian, "DE" for German, "en GB" for UK English, "en US" for US English, etc....

EXPRESS specification:

```
*)
TYPE language_code = identifier;
END_TYPE;
(*
```

### 7.1.2 global_language_assignment

The **global_language_assignment** entity specifies the language for **translatable_label** and **translatable_text,** if **label** and **text** are selected, respectively (i.e. without explicit language indication as is done in **translated_label** and translated_text).

EXPRESS specification:

```
*)
ENTITY global_language_assignment;
       language: language_code;
END_ENTITY;
(*
```

Attribute definitions:

**language:** the code of the assigned language

### 7.1.3 present_translations

The **present_translations** entity serves to indicate the languages used for **translated_label** and **translated_text.**

EXPRESS specification:

```
*)
ENTITY present_translations;
       language_codes: LIST [ 1: ? ] OF UNIQUE language_code;
UNIQUE
       UR1: language_codes;
END_ENTITY;
(*
```

Attribute definitions:

**language_codes:** the list of unique language codes corresponding to the language in which a translation is made.

Formal proposition:

**UR1**: for each list of **language_code** there shall be a unique instance of **present_translations**

### 7.1.4 translatable_label

A **translatable_label** defines a type of values that can be **labels** or **translated_labels.**

EXPRESS specification:

```
*)
TYPE translatable_label = SELECT (label, translated_label);
END_TYPE;
(*
```

### 7.1.5 translated_label

The **translated_label** entity defines the labels that are translated and the corresponding languages of translation.

EXPRESS specification:

```
*)
ENTITY translated_label;
       labels: LIST [ 1: ? ] OF label;
       languages: present_translations;
WHERE  WR1: SIZEOF (labels ) = SIZEOF (languages.language_codes);
END_ENTITY;
(*
```

Attribute definitions:

**labels:** the list of **labels** which are translated.

**languages**: the list of **languages** in which each label is translated.

Formal propositions:

**WR1:** The number of **labels** contained in the **labels** list shall be equal to the number of languages provided in the **languages.language_codes** attribute.

Informal propositions:

**IP1:** The content of **labels [i]** is in the language identified by **languages.language_codes [i]**.

### 7.1.6 translatable_text

A **translatable_text** defines a type of values that can be **texts** or **translated_texts**.

EXPRESS specification:

```
*)
TYPE translatable_text = SELECT ( text, translated_text);
END_TYPE;
(*
```

### 7.1.7 translated_text

The **translated_text** entity defines the **texts** that are translated and the corresponding **languages** of translation.

EXPRESS specification:

```
*)
ENTITY translated_text;
       texts: LIST [ 1: ? ] OF text;
       languages: present_translations;
WHERE
       WR1: SIZEOF (texts ) = SIZEOF (languages.language_codes);
END_ENTITY;
(*
```

Attribute definitions:

**texts:** the list of **texts** which are translated.

**languages:** the list of **languages** in which each text is translated.

Formal propositions:

**WR1:** The number of **texts** contained in the **texts** list shall be equal to the number of languages provided in the **languages.language_codes** attribute.

Informal propositions:

**IP1:** The content of **texts [ i ]** is in the language identified by **languages.language_codes [ i ]**.

## 7.2 ISO13584_IEC61360_language_resource_schema function definitions

This subclause contains functions which are referenced in WHERE clauses to assert data consistency.

### 7.2.1 check_label_length function

The **check_label_length** function checks that no label in **l** exceeds the length indicated by **l_length.**

EXPRESS specification:

```
*)
FUNCTION check_label_length (
             l: translatable_label;
             l_length: INTEGER): BOOLEAN;
IF 'ISO13584_IEC61360_LANGUAGE_RESOURCE_SCHEMA.TRANSLATED_LABEL'
       IN TYPEOF(l) THEN
       REPEAT i:=1 TO SIZEOF (l.labels);
             IF LENGTH(l.labels[i]) > l_length THEN
             RETURN (FALSE);
             END_IF;
       END_REPEAT;
       RETURN (TRUE);
ELSE (* the argument l is a single string *)
       RETURN (LENGTH(l) <= l_length);
END_IF;
END_FUNCTION; -- check_label_length
(*
```

### 7.2.2 check_text_length function

The **check_text_length** function checks that no text in **t** exceeds the length indicated by **t_length.**

EXPRESS specification:

```
*)
FUNCTION check_text_length (
             t: translatable_text;
             t_length: INTEGER): BOOLEAN;
IF 'ISO13584_IEC61360_LANGUAGE_RESOURCE_SCHEMA.TRANSLATED_TEXT'
       IN TYPEOF(t) THEN
       REPEAT i:=1 TO SIZEOF (t.texts);
             IF LENGTH(t.texts[i]) > t_length THEN
             RETURN (FALSE);
             END_IF;
       END_REPEAT;
       RETURN (TRUE);
ELSE (* the argument t is a single string *)
       RETURN (LENGTH(t) <= t_length);
END_IF;
END_FUNCTION; -- check_text_length
(*
```

## 7.3 ISO13584_IEC61360_language_resource_schema rule definition

The rule **single_language_assignment** asserts that only one language may be assigned to be used in **translatable_label** and **translatable_text,** respectively.

EXPRESS specification:

```
*)
RULE single_language_assignment FOR (global_language_assignment);
WHERE
       SIZEOF ( global_language_assignment ) <= 1;
END_RULE;

END_SCHEMA; -- ISO13584_IEC61360_language_resource_schema
(*
```

## 8 Example Physical File

### 8.1 Some example data

This subclause gives some fragments of a Physical File for exchanging the data of IEC 61360-4: "International Reference List of Standard Data Element Types for Electric Components". It is intended to show the use of the EXPRESS model in clause 5 "ISO13584_IEC61360_dictionary_schema" together with ISO 10303-21 to exchange corresponding data.

```
ISO-10303-21;
HEADER;
...
ENDSEC;
DATA;
```

#### 8.1.1 Supplier data

```
#1=    SUPPLIER_BSU('01122//61360-4',*,* );
#2=    SUPPLIER_ELEMENT(#1, #3, '01', *, #4, #5);
#3=    DATES('1994-09-16', '1994-09-16', $);
#4=    ORGANIZATION('IEC 61360-4', 'IEC 61360-4 Maintenance Agency', 'The
       IEC Maintenance Agency as described in IEC 61360-2: "Maintenance and
       Validation Procedures"');
#5=    ADDRESS('to be determined', $, $, $, $, $, $, $, $, $, $, $);
#10=   SUPPLIER_BSU('01123//-00', *, * );  /* ISO/IEC ICS */
```

#### 8.1.2 Root class data

The AAA000 IEC root class provides a name scope corresponding to the whole IEC 61360-4. It covers two trees, one for materials, one for components, therefore the class is defined as an item_class. It is a subtype of ICS root.

```
#90=   CLASS_BSU('00', '001', *, #10);      /* ICS root */
#100=  CLASS_BSU('AAA000', '001', *, #1);
#101=  ITEM_CLASS(#100, #3, '01', #102, 'IEC root class that provides a name
       scope corresponding to the whole IEC 61360-4 standard. It covers two
       trees, one for materials, one for components', $, $, $, *, #90,
       (#110), (), $, (#110), (), $);
#102=  ITEM_NAMES('IEC root', $, 'AAA000-001', $, $);
#110=  PROPERTY_BSU('AAE000', '001', *, #100);
#111=  NON_DEPENDENT_P_DET(#110, #3, '01', #112, 'the type of tree: material
       or component', $, $, $, *, $, (), $, $, #113, $);
#112=  ITEM_NAMES('type of tree', $, 'tree type', $, $);
#113=  NON_QUANTITATIVE_CODE_TYPE('A..8', #114);
#114=  VALUE_DOMAIN((#120, #122), $, $, ());
#120=  DIC_VALUE('MATERIAL', #121, $);
#121=  ITEM_NAMES('material tree', $, 'mat tree', $, $);
#122=  DIC_VALUE('COMPONS', #123, $);
#123=  ITEM_NAMES('component tree', $, 'comp tree', $, $);
```