IFC 2x2 Measure and Unit Examples

In the IFC Object Model there may be a number of ways to use the measure defined types (for measure values) and their units. The purpose of this document and the examples is to promote common understanding of how measure types and related units should be used.

Introduction

In different types of units there are five basic cases:

- Basic *SI-units*, which cover a number of fundamental units of mainly physical quantities defined by ISO-1000+A1,1992 & 1998.; e.g. meter or millimeter as unit for length measure or square meter as a unit for area measure. The unit may have a scaling prefix (milli, kilo, etc.).
- Derived *SI-units*, which are defined as a derivation of the basic SI-units, e.g. Newton (kg m / s^2) as a unit of force. Both basic and derived SI-units are in the IFCs represented by IfcSIUnit.
- *Conversion based units*, which can be derived from SI-units by a scaling factor; e.g. inch which can be defined using SI-unit for length measure, i.e. an inch is 25.4 millimeters.
- *Derived units*, which can be defined as a derivation or combination of a number of basic units. In a derived unit each of the basic unit "component" has a dimensional exponent in defining the derived unit; e.g. kg / m^2, where kilogram (kg) has dimensional exponent 1 and meter (m) has exponent -2.
- *Context dependent units*, which cannot be defined as conversion based unit using SI-units.

With regard to the usage of the measure defined types (IfcLengthMeasure, IfcTimeMeasure, etc.) as attribute datatypes in the IFC Object Model, there three basic cases:

1) The datatype of an attribute of an entity type is a measure defined type as such without possibility on an instance level to define the unit of the measure value; E.g.

```
ENTITY IfcBoundingBox
    SUBTYPE OF(IfcGeometricRepresentationItem);
      Corner : IfcCartesianPoint;
      XDim   : IfcPositiveLengthMeasure;
      YDim   : IfcPositiveLengthMeasure;
      ZDim   : IfcPositiveLengthMeasure;
  END_ENTITY;
```

and

```
ENTITY IfcScheduleTimeControl
    SUBTYPE OF(IfcControl);
      ActualStart                : OPTIONAL IfcCalendarDate;
      EarlyStart                 : OPTIONAL IfcCalendarDate;
      LateStart                  : OPTIONAL IfcCalendarDate;
      ScheduleStart              : OPTIONAL IfcCalendarDate;
      ActualFinish               : OPTIONAL IfcCalendarDate;
      EarlyFinish                : OPTIONAL IfcCalendarDate;
      LateFinish                 : OPTIONAL IfcCalendarDate;
      ScheduleFinish             : OPTIONAL IfcCalendarDate;
      ScheduleDuration           : OPTIONAL IfcTimeMeasure;
      ActualDuration             : OPTIONAL IfcTimeMeasure;
      RemainingTime              : OPTIONAL IfcTimeMeasure;
      FreeFloat                  : OPTIONAL IfcTimeMeasure;
      TotalFloat                 : OPTIONAL IfcTimeMeasure;
      IsCritical                 : OPTIONAL BOOLEAN;
      StatusTime                 : OPTIONAL IfcCalendarDate;
      StartFloat                 : OPTIONAL IfcTimeMeasure;
      FinishFloat                : OPTIONAL IfcTimeMeasure;
      Completion                 : OPTIONAL IfcPositiveRatioMeasure;
  END_ENTITY;
```

In this case, it is the global unit assignment for the corresponding unit for the measure type that defines the unit for all the usages of this defined measure type (except for cases 2 and 3 below).

2) The datatype of an attribute is IfcMeasureWithUnit, which allows for definition of unit per instance of that entity type, independent and possibly overriding the global unit assignment; E.g.

```
ENTITY IfcConstructionMaterialResource
    SUBTYPE OF(IfcResource);
      Suppliers      : OPTIONAL SET [1:?] OF IfcActorSelect;
      OrderQuantity  : OPTIONAL IfcMeasureWithUnit;
      DesignMaterial : OPTIONAL SET [1:?] OF IfcMaterial;
  END_ENTITY;
```

In this case the relevant measure defined type (from the IfcMeasureWithUnit. ValueComponent : IfcValue select list) is not exactly defined by the schema, but implied by the context.

3) The entity type has a separate "unit" attribute which allows for defining the unit for another attribute of the entity type for representing the actual value; E.g.

```
ENTITY IfcPropertySingleValue
 SUBTYPE OF (IfcSimpleProperty);
   NominalValue : IfcValue;
   Unit : OPTIONAL IfcUnit;
END_ENTITY;
```

Although in the cases 2 and 3 different units could be used for different instances of the same entity type or for the same measure type in attributes of different entity types, it is recommended not to mix different units for same measure defined types, if it can be avoided.

Below some examples of each of the above basic cases are given.

Note: In the example instantiations in the form of IFC data exchange files, mainly the measure and unit -relevant attributes are given the values; the other attributes are given no values (in the form of $-sign) independent of whether they should actually have values because of being non-optional attributes.

Besides this document, it may be useful to have a look at measure and unit summary document which gives an overview of the correspondence of various units, corresponding defined measure types etc.

Global unit assignment

The global unit assignment of the project defines the global units for measures and values when the units are not otherwise defined more specifically using entity type IfcMeasureWithUnit as attribute's datatype.

Basic SI-units as global units

An example where a project's global basic length, area, volume and time units are defined as SI units:

```
#1=IFCPROJECT ('00ZhrqZYLBcgy$rVVaiu2A', $, 'Example project', $,
               $, $, $, $, #2);
#2=IFCUNITASSIGNMENT ((#3, #4, #5, #6));
#3=IFCSIUNIT (*, .LENGTHUNIT., .MILLI., .METRE.);
#4=IFCSIUNIT (*, .AREAUNIT., $, .SQUARE_METRE.);
#5=IFCSIUNIT (*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#6=IFCSIUNIT (*, .TIMEUNIT., $, .SECOND.);
```

Note: In the examples the '*' character as the first value in the IfcSIUnit-instances is due to the fact that in the IfcSIUnit the inherited attribute .Dimensions is redefined as a derived attribute. In the IFC Object Model schema there is a function that returns the derived dimensional exponent values for SI units. In the exchange file the derived attribute values are not exchanged, and in the case of redefinition the value is replaced by '*'.

Use of defined measure type with global unit assignment

An example where the attribute `IfcScheduletimeControl.ScheduleDuration` is of datatype `IfcTimeMeasure` without possibility of using specific unit from `IfcMeasureWithUnit`. Then the global unit for time measure applies; in this case seconds (the example's 172800 seconds would be equivalent to 2 days):

```
#1=IFCPROJECT ('00ZhrqZYLBcgy$rVVaiu2A', $, 'Example project', $,
               $, $, $, $, #2);
#2=IFCUNITASSIGNMENT ((#3, #4, #5, #6));
#3=IFCSIUNIT (*, .LENGTHUNIT., .MILLI., .METRE.);
#4=IFCSIUNIT (*, .AREAUNIT., $, .SQUARE_METRE.);
#5=IFCSIUNIT (*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#6=IFCSIUNIT (*, .TIMEUNIT., $, .SECOND.);
#7=IFCSCHEDULETIMECONTROL('00ZhrqZYLBcgy$rVVaiu2B', $, 'Time for
      Task-1', $, $, $, $, $, $, $, $, $, $, 172800., $, $, $, $,
      $, $, $, $, $);
```

Conversion based Imperial units as global units

An example where a project's global basic length, area and volume units are defined as imperial units (inches, square feet and cubic feet), which are further defined as conversion based units relative to SI units millimeter, square meter and cubic meter:

```
#1=IFCPROJECT ('00ZhrqZYLBcgy$rVVaiu2B', $, 'Example project', $,
               $, $, $, $, #2);
#2=IFCUNITASSIGNMENT ((#6, #7, #10, #13));
#3=IFCSIUNIT (*, .LENGTHUNIT., .MILLI., .METRE.);
#4=IFCSIUNIT (*, .AREAUNIT., $, .SQUARE_METRE.);
#5=IFCSIUNIT (*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#6=IFCSIUNIT (*, .TIMEUNIT., $, .SECOND.);
#7=IFCCONVERSIONBASEDUNIT(#9, .LENGTHUNIT., 'INCH', #8);
#8=IFCMEASUREWITHUNIT(IFCLENGTHMEASURE(25.4), #3);
#9=IFCDIMENSIONALEXPONENTS(1, 0, 0, 0, 0, 0, 0);
#10=IFCCONVERSIONBASEDUNIT(#11, .AREAUNIT., 'SQUARE_FEET', #12);
#11=IFCDIMENSIONALEXPONENTS(2, 0, 0, 0, 0, 0, 0);
#12=IFCMEASUREWITHUNIT(IFCAREAMEASURE(0.09290304), #4);
#13=IFCCONVERSIONBASEDUNIT(#14, .VOLUMEUNIT., 'CUBIC_FEET', #15);
#14=IFCDIMENSIONALEXPONENTS(3, 0, 0, 0, 0, 0, 0);
#15=IFCMEASUREWITHUNIT(IFCVOLUMEMEASURE(0.0283168466), #5);
```

In a conversion based unit the scaling factor, like 0.0283168466 for cubic feet, expresses how many base units make up one conversion based unit.

Another example of a conversion based unit would be the definition of temperature in degrees of Fahrenheits. A conversion based unit should be used for this purpose, because degrees of Fahrenheits can in principle be defined using degrees of Celsius. However, with IfcConversionBasedUnit one cannot define their relationship fully because the zero degrees offset between them cannot be defined. So, it is assumed that applications can solve the dependency between degrees Fahrenheit and Celsius.

Yet another example of a conversion based unit would be degrees and gradians as a unit for plane angle measure. These can be defined as a conversion based unit based on radian unit.

Use IfcMeasureWithUnit and specific unit per instance

An example where the `IfcConstructionMaterialResource.OrderQuantity` of datatype `IfcMeasureWithUnit` is given a value in litres (value 10 litres e.g. as a quantity value for paint) as a conversion based unit based on SI unit cubic meter. Here an “overriding” specific unit can be defined on the instance level independent of the global unit for volume measure (here SI unit cubic meter):

```
#1=IFCPROJECT('00ZhrqZYLBcgy$rVVaiu2A', $, 'Example project', $,
              $, $, $, $, #2);
#2=IFCUNITASSIGNMENT((#3, #4, #5, #6));
#3=IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#4=IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#5=IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#6=IFCSIUNIT(*, .TIMEUNIT., $, .SECOND.);
#7=IFCCONSTRUCTIONMATERIALRESOURCE('00ZhrqZYLBcgy$rVVaiu2B', $,
      'Paint resource', $, $, $, $, $, #8, $);
#8=IFCMEASUREWITHUNIT(10., #9);
#9=IFCCONVERSIONBASEDUNIT(#10, .VOLUMEUNIT., 'LITRE', #11);
#10=IFCDIMENSIONALEXPONENTS(3, 0, 0, 0, 0, 0, 0);
#11=IFCMEASUREWITHUNIT(0.001, #5);
```

A derived unit

An example definition of a unit for specific heat capacity (Joule / kg Kelvin), which is defined as a derived unit based on basic SI units. In the example the attribute IfcThermal-MaterialProperties.SpecificHeatCapacity is of datatype IfcSpecificHeatCapacityMeasure:

```
#1=IFCPROJECT('00ZhrqZYLBcgy$rVVaiu2A', $, 'Example project', $,
              $, $, $, $, #2);
#2=IFCUNITASSIGNMENT((#3, #4, #5, #6, #7, #8, #9, #10));
#3=IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#4=IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#5=IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#6=IFCSIUNIT(*, .TIMEUNIT., $, .SECOND.);
#7=IFCSIUNIT(*, .ENERGYUNIT., $, .JOULE.);
#8=IFCSIUNIT(*, .MASSUNIT., .KILO., .GRAM.);
#9=IFCSIUNIT(*, .THERMODYNAMICTEMPERATUREUNIT., $, .KELVIN.);
#10=IFCDERIVEDUNIT((#11, #12, #13), .SPECIFICHEATCAPACITYUNIT.,
                   $);
#11=IFCDERIVEDUNITELEMENT(#7, 1);
#12=IFCDERIVEDUNITELEMENT(#8, -1);
#13=IFCDERIVEDUNITELEMENT(#9, -1);
#15=IFCTHERMALMATERIALPROPERTIES($, 123., $, $, $);
```

✧