Plates and Decking in CIS/2

By Frank Wang and Chuck Eastman

Purpose

Although most steel structures are made up of prismatic linear parts, decking and plates also are often components of a steel structure. This tutorial outlines the use of surface elements in analysis and manufacturing models. Decking and plates are defined generally as a plane surface, with three or more connections in the plane. Any decking profile, such as corrugations, consists of a cross section centered on that plane. Decking profiles are not defined in a flavor cross section file of AISC, thus they must be defined explicitly. The tutorial is offered as an illustration of one of the several approaches of how to define a decking member in CIS/2.

Like other Georgia Tech tutorials, we describe some of the Express code definitions in “flattened” form. This means that all inherited attributes are presented, within parentheses, in the order they would be in a Part 21 file.

Decking in Analysis Model

In an analytical model, decking can be specified as an *element_surface_profiled* entity, which has a subtype of *element_surface_complex* and is inherited from *element_surface* through *element*. The inheritance structure of *element_surface_profiled* is shown Figure One below.

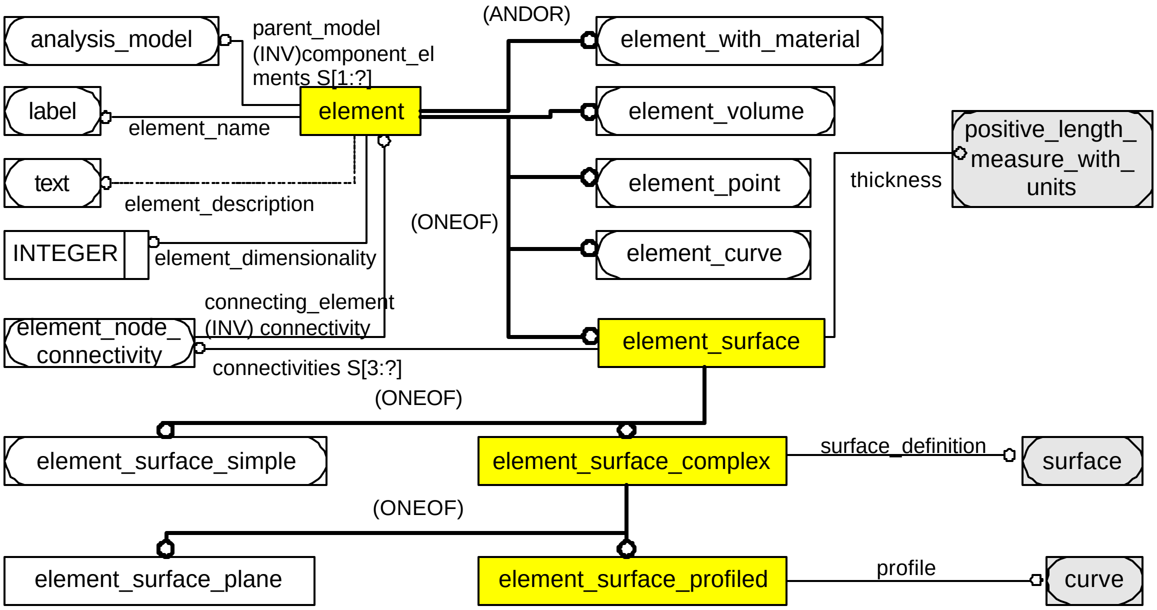

Figure One: EXPRESS-G for element and element_surface.

A flattened EXPRESS code is also listed to have a better understanding of the inherited relationship between *element_surface_complex* and *element.*

Flattened EXPRESS code for element_surface_profiled

```
ENTITY element_surface_profiled
(element_surface_complex:
       (element_surface:
              (element:
                    element_name : label;
                    element_description : OPTIONAL text;
                    parent_model : analysis_model;
                    element_dimensionality : INTEGER;
                    INVERSE
                       connectivity : SET [1:?] OF element_node_connectivity FOR
                       connecting_element;
                    UNIQUE
                       URE1 : element_name, parent_model;
                    );
             thickness : positive_length_measure_with_unit;
             DERIVE
                  connectivities : SET [3:?] OF element_node_connectivity :=
                  bag_to_set(USEDIN(SELF,
                  'STRUCTURAL_FRAME_SCHEMA.ELEMENT_NODE_CONNECTIVITY.CONNECTING_ELEMENT'
             );
      );
      surface_definition : surface;
);
profile : curve;
END_ENTITY;
```

Each *element* carries an inverse relation that associates the *element_node_connectivity* and indirectly the *nodes* the *element* is connected to. *Element_surface*, derives an aggregated set of these in attribute *connectivities*, which requires at least three associated *element_node_ connectivities*. Another attribute of *element_surface* is a thickness of the plate or decking member. This thickness is specified in the x-axis direction, with the decking laid out in the y-z plane.

It is assumed that for the purposes of analysis, all plates and most decking will be defined as a plane, using *element_surface_simple*. However, means are provided for structural analysis of corrugated or other forms of bent metal decking. The *element_surface_complex* a subtype of *element_surface*, carries the attribute *surface,* an entity that is defined explicitly in STEP Part 42 – "Geometric and topological representation (ISO 10303-42 The EXPRESS-G structure for surface and its subtypes is shown in Figure Six.

For purposes of analysis, the relevant plate or decking is bounded by the polygon defined by its connections (which may not be its true shape). The first connecting node is assumed to be the sheet piece's local origin (without eccentricities). The y-axis of the sheet is assumed to be in the direction from the first connection toward the second connection. The third connection defines the yz_plane. The origin and axes will shift if eccentricities are applied. The *element_surface_profiled* allows an implementer to define a non-planar cross-section by referencing a curve as a profile. Entities curve and surface are defined and discussed later in the following tutorial.

The STEP code shown as below is an example of defining an *element_surface_profiled* instance (#15). It includes a non-planar profile. The third variable in instance #15 is referenced to an *analysis_model* instance at #30, the fourth variable is an integer that specifies the dimensionality of an element as an integer, the fifth variable pointsto a *positive_length measure_with_unit* instance (#16) that indicates the thickness of the element, the sixth variable points to a surface instance (#17) where the element surface is carried and the last variable points to a curve instance that defines the profile (#18).

```
#15=ELEMENT_SURFACE_PROFILED('SURFACE_NAME','SURFACE_DESCRIPTION',#30,2,#16,#17,
#18);
#16=POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(0.125),#28);
/* surface */
#17=PLANE('NAME_OF_PLANE', #610);
#610=AXIS2_PLACEMENT_3D('AXES FOR PLANE',#620,#630,#640);
```

```
#620=CARTESIAN_POINT('ORIGIN FOR PLANE',(0.,0.,0.));
#630=DIRECTION('local x',(0.,1.,0.));
#640=DIRECTION('local z',(1.,0.,0.));
/* curve */
#18=COMPOSITE_CURVE('PROFILED_CURVE',(#200),.F.);
/* prototype curve */
#200=COMPOSITE_CURVE_SEGMENT(.CONTINUOUS.,.T.,#100);
/* polyline */
#100=POLYLINE('POLYLINE',(#110,#120,#130,#140,#150,#160));
#110=CARTESIAN_POINT('point-1',(0.,0.,0.));
#120=CARTESIAN_POINT('point-2',(1.,3.,0.));
#130=CARTESIAN_POINT('point-3',(3.,3.,0.));
#140=CARTESIAN_POINT('point-4',(5.,-3.,0.));
#150=CARTESIAN_POINT('point-5',(7.,-3.,0.));
#160=CARTESIAN_POINT('point-5',(8.,0.,0.));
#30=ANALYSIS_MODEL('default 3d','default 3d',.SPACE_FRAME.,$,3);
```

Decking in Manufacturing Model

In a manufacturing model, *part_sheet_profiled* is a sub-type of a *part* that can be used to define a decking member in CIS/2. A detailed relationship among *part_sheet_profiled* and its inherited entities are shown in Figure Two below.

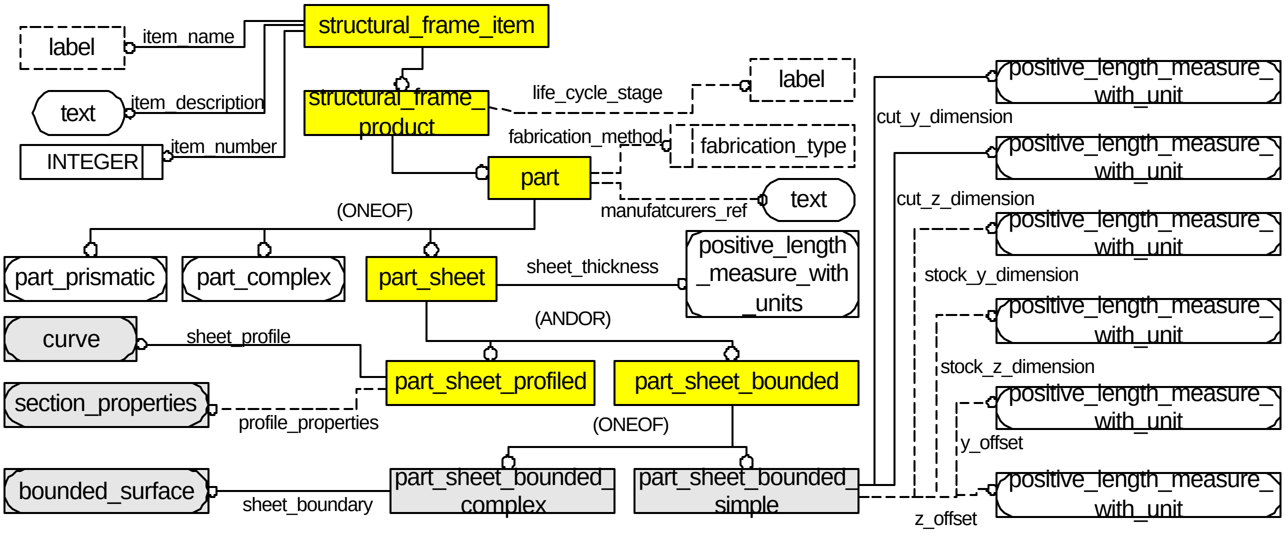

Figure Two: Part_sheet and its subtypes.

The flattened EXPRESS code of *part_sheet_profiled* is also given as follow:

```
ENTITY part_sheet_profiled
(part_sheet:
       (part:
             (structural_frame_product:
                    (structural_frame_item:
                           item_number : INTEGER;
                           item_name : label;
                           item_description : OPTIONAL text;
                    )
                    life_cycle_stage : OPTIONAL label;
             )
             fabrication_method : fabrication_type;
             manufacturers_ref : OPTIONAL text;
```

```
            );
        sheet_thickness : positive_length_measure_with_unit;
        );
        sheet_profile : curve;
        profile_properties : OPTIONAL section_properties;
END_ENTITY;
```

The way that CIS/2 handles a physical decking member in the manufacturing model is similar to the analysis model; the thickness of a decking is indicated as an attribute of entity *part_sheet* with a p*ositive length measure with units* value. Same as the analysis model, thickness is normal to the local yz-plane.

Eventually, *part_sheet_profiled* and *part_sheet_bounded* are two subtypes that are inherited from the entity *part_sheet*, within an *ANDOR* relationship that can completely represent the geometry of a decking member. The *part_sheet_profiled* will specify the cross-section profile of a sheet member by referencing a *curve* entity, an entity defined in Part 42 (ISO 10303-42). (A common depiction of this curve is presented in the next section.)The entity *part_sheet_bounded* will be used to define the outer boundary of a physical sheet member. The boundary of a sheet member can be interpreted in two ways; the first way is using a *part_sheet_bounded_simple* instance to define a surface boundary of a sheet member. This is often used when the sheet has a rectangular boundary. There are six attributes, all of them are defined as type of *positive_length_measure_with_unit*, which can be categorized into three pairs of parameters: 1. A cutting dimension on both y and z axes. 2. An optional stocking dimension on both y and z axes. 3. An optional offset distance on both y and z axes. The *cut_y_dimension* and *cut_z_dimension* are variables to specify the length and width as a finite boundary for a bounded sheet. Notice that the thickness of the physical decking is already defined as an offset along the local x-axis.
If the boundary of a decking member has a more irregular shape rather than a rectangular boundary, as in the above case, a *part_sheet_bounded_complex* can be used to customize an irregular bounded surface for the physical decking member. The boundary of this surface will be defined as a *bounded_surface*, an entity also defined in Part 42 (ISO 10303-42), to define the outer boundary of a physical sheet member.

If *part_sheet_bounded_simple* is used, and the ANDOR *part_sheet_profiled* is included, the simple case for defining a rectangular shape member, is given in an example as follows:

```
/* simple shape */
#400=(STRUCTURAL_FRAME_ITEM(5, 'ITEM LABEL','ITEM DESCRIPTION')
      STRUCTURAL_FRAME_PRODUCT($)
      PART(.UNDEFINED.,'MANUFACTURES_REF')
      PART_SHEET(#430)
      PART_SHEET_BOUNDED_SIMPLE(#410,#420,$,$,$,$)
      PART_SHEET_PROFILED(#300,$)
 );
#410=POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(100.0),#11);
#420=POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(80.0),#11);
#430=POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(0.125),#11);
```

An example for a *curve* is given on page 7.

An example of *part_sheet_bounded_complex* to define a decking surface is as follows:

```
/* complex shape */
#400=(STRUCTURAL_FRAME_ITEM(5, 'ITEM LABEL','ITEM DESCRIPTION')
      STRUCTURAL_FRAME_PRODUCT($)
      PART(.UNDEFINED.,'MANUFACTURES_REF')
      PART_SHEET(#430)
      PART_SHEET_BOUNDED_COMPLEX(#700)
      PART_SHEET_PROFILED(#300,$)
 );
#700=RECTANGULAR_TRIMMED_SURFACE('SURFACE_NAME',#600,72.0,72.0,72.0,72.0,.F.,.F.
);
#430=POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(0.125),#11);
```

In the simple case, the instance #400 is presenting as an external mapping for an *ANDOR* relation between *part_sheet_bounded_simple* and *part_sheet_profiled*. Instances #410 and #420 are represented as the length and width of the part in measurable length units. Instance #430 indicates the thickness of the part. In the last line of #400, the *part_sheet_profiled* points to the #300, a *bounded_curve* instance, which is the profile of the decking. This example will also be given in the following section.

In the complex case, #400 will point to #700 as a sheet boundary. This boundary will be defined by a bounded surface that shown as *rectangular_trimmed_surface*.

Geometry Representation Items in Part 42

Curve

Curve is an abstract geometry entity and there are various ways to define a curve for a cross-section profile in Part 42. In this tutorial, we will introduce one way to define a customized cross-section profile. In basic idea is:
1. Define a polyline as a prototyped curve, marked as the first curve.
2. Reference the polyline as a parent curve of *composite_curve_segment*. The replicated polyline will composite a bounded curve, which will be noted as the second curve and it is the curve to form the cross-section profile.

Figure 5 and figure 6 show a detailed definition of a curve and inherited relationship of curve in part 42.

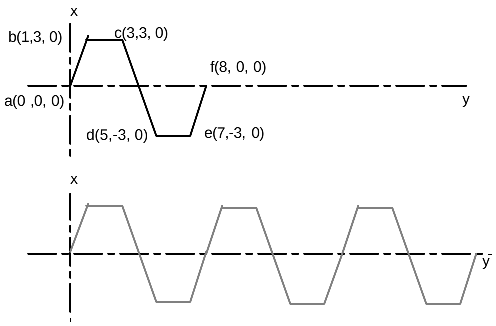

Figure Three: Compositing a Profiled Curve

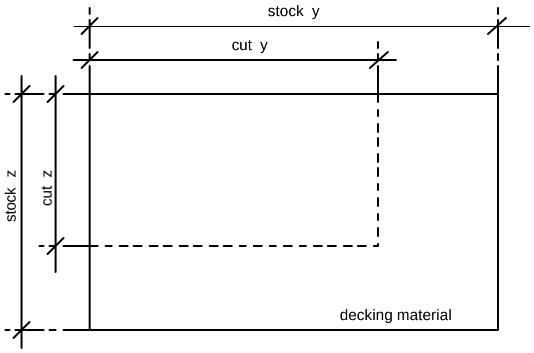

Figure Four: Definea trimmed decking by using part_boubnded_simple

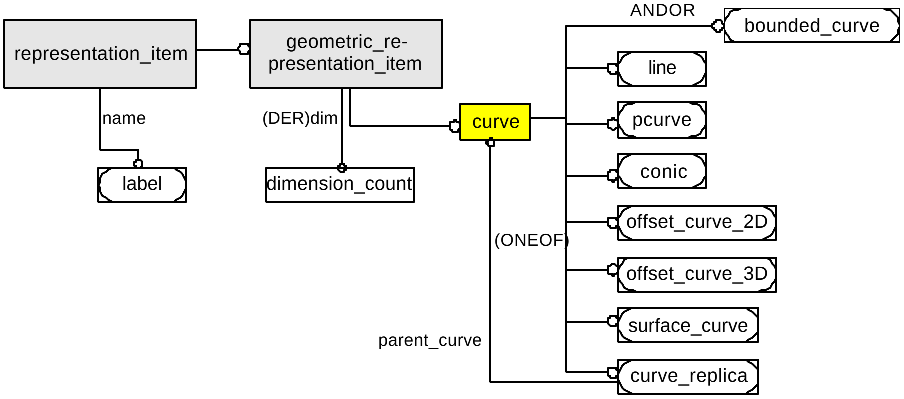

Figure Five: Definition of Curve in Part 42

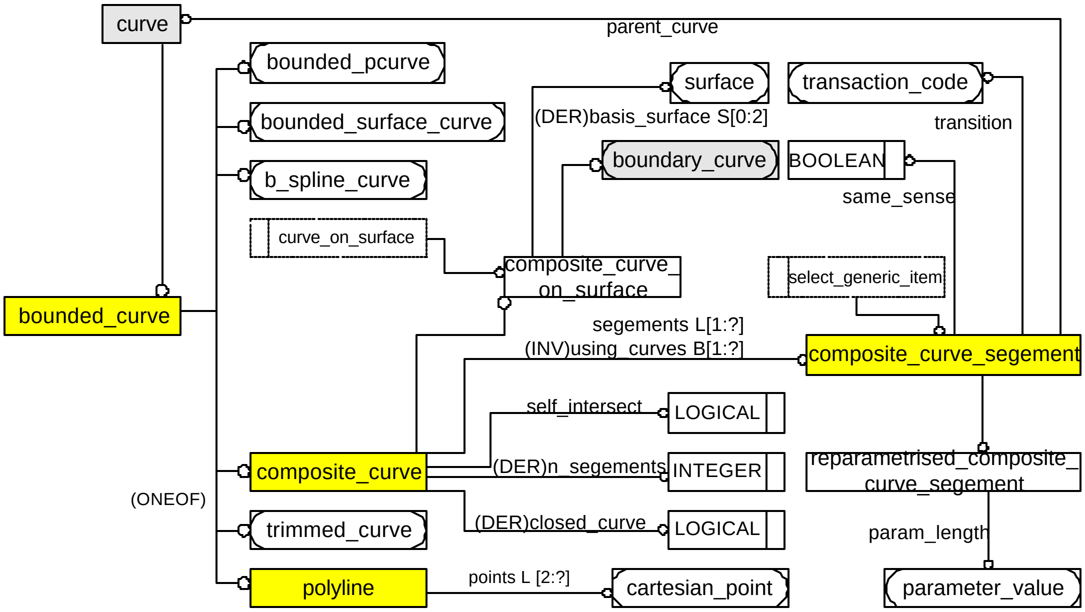

Figure Six: Detail Definition of cure in Part 42

Flattened EXPRESS code for composite_curve

```
ENTITY composite_curve
(bounded_curve:
       (curve:
              (geometric_representation_item:
                     (represetation_item:
                            name : label;
                     )
              DERIVE
              dim : dimension_count := dimension_of(SELF);
              )
       )
)
   segments : LIST [1:?] OF composite_curve_segment;
   self_intersect : LOGICAL;
DERIVE
        n_segments : INTEGER := SIZEOF(segments);
        closed_curve : LOGICAL := segments[n_segments].transition <> discontinuous;
```

```
WHERE
        WR1:
        NOT closed_curve AND (SIZEOF( QUERY (temp <* segments| (temp.transition =
        discontinuous))) = 1) OR closed_curve AND (SIZEOF( QUERY (temp <* segments|
        (temp.transition = discontinuous))) = 0);
END_ENTITY;
```

The *bounded_curve*, a finite arc with two ends defined, is an *ANDOR* subtype of *curve* and it also is a supertype of both *composite_curve* and *polyline*, which will geometrically define the profile of a decking member and the prototyped curve. There exists an inverse relationship between entities *composite_curve_segment* and *composite_curve*. The aggregated list of segments defines a collection of curve segments, which assemble the *composite_curve*. The inverse relation between *composite_curve* and *composite_curve_segment* defines that a curve is composed by replicated segments of a curve.

The following P21 instances are the example of the prototyped curve and the profiled curve.

```
/* PROFILED CURVE */
#300=COMPOSITE_CURVE('PROFILED_CURVE',(#200),.F.);

/* PROTOTYPE CURVE */
#200=COMPOSITE_CURVE_SEGMENT(.CONTINUES.,.T.,#100);

/* POLYLINE */
#100=POLYLINE('POLYLINE',(#110,#120,#130,#140,#150,#160));
#110=CARTESIAN_POINT('point-1',(0.,0.,0.));
#120=CARTESIAN_POINT('point-2',(1.,3.,0.));
#130=CARTESIAN_POINT('point-3',(3.,3.,0.));
#140=CARTESIAN_POINT('point-4',(5.,-3.,0.));
#150=CARTESIAN_POINT('point-5',(7.,-3.,0.));
#160=CARTESIAN_POINT('point-5',(8.,0.,0.));
```

Instance #300 is a *composite_curve* instance that indicates the profile curve for the decking. The first variable within it is a named label of the instance, the second variable is an aggregated list of segments that reference the previous *composite_curve_segment* instance at #200 and the last variable is a logical flag to determine if the curve has intersected to itself.

The first variable in #200 is the enumerated transaction code, a *CONTINUES* type is selected to identify that the curve is replicated. The second variable is a Boolean parameter, which indicate whether or not the direction of the segment agrees with the parent curve. If the sense is “false”, a point that has a higher value will become the start point of the curve. The last variable is the reference to a parent curve that defines the polyline in #100.

The prototyped curve is composed by a polyline that continuously linked by a series *cartesian point*, shows from line #110 to line #160.

Bounded Surface

Plane

The plane is an entity, which has no attribute associated within it, and it is also a subtype of *surface*. The location of a *plane* will be specified in its super type, the *elementary_surface*, which is a type of *axis2_placement_3d*. *Plane* is also defined by a point on a plane plus a normal direction of the plane in part 42. The plane is an

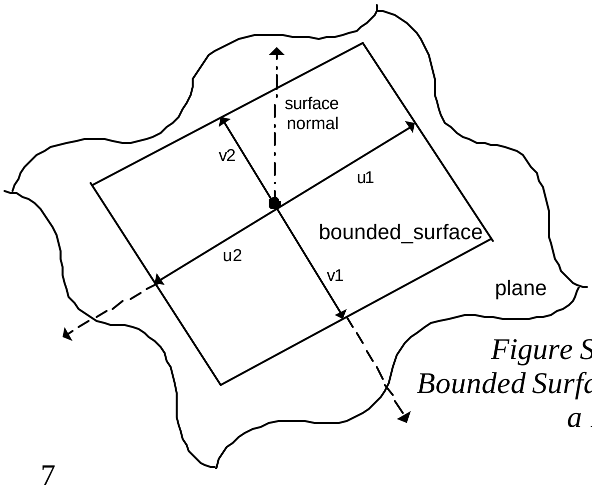

Figure Seven: Bounded Surface on a Plane unbounded surface where the parameterization ranges are infinite.

(If surface => s =(u,v), then plane => s =(u,v), when -8<u,v<8)

One of the ways to represent a *bounded_surface* is applying a *rectangle_trimmed_surface* entity, a subtype of *bounded_surface*, which defines the boundary of a plane.
The parameters u1, u2, v1 and v2 define the dimension of the boundary (see figure 7) as a value of parameter. The *parameter_value* is referenced externally to Part 41 "Fundamentals of product description and support", defined as a measure schema. Another attribute of *rectangular_trimmed_surface* is a *basis_surface* that points to the unbounded plane.

The following EXPRESS-G figure is a detailed definition of *bounded_surface* in Part 42.

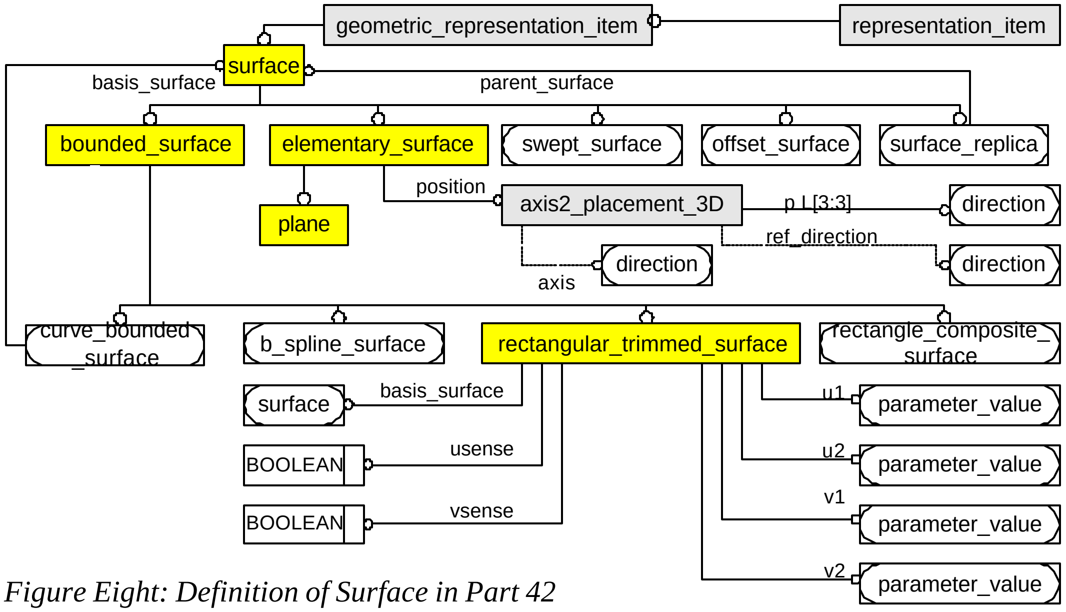

Figure Eight: Definition of Surface in Part 42

Flattened EXPRESS code for plane

```
ENTITY plane
(elementary_surface:
       (surface:
              (geometric_representation_item:
                     name: label;
              )
       )
       position : axis2_placement_3d;
)
END_ENTITY;
```

Flattened EXPRESS code for rectangle_trimmed_surface

```
ENTITY rectangle_trimmed_surface
(bounded_surface:
              (surface:
                     (geometric_representation_item:
                            name: label;
                     )
              )
```

```
        )
        basis_surface : surface;
        u1 : parameter_value;
        u2 : parameter_value;
        v1 : parameter_value;
        v2 : parameter_value;
        usense : BOOLEAN;
        vsense : BOOLEAN;
)
END_ENTITY;
```

In the instance level, there are two parameters associated with defining a plane. The first parameter is a label that indicates the name of the plane. The plane is an infinite surface with no boundary specified and carries the prospected bounded surface of a decking member.
The second is the position of a plane by pointing to an *axis2_placement_3d* instance. It defines an origin and two vectors in the relevant coordinate system, defining the coordinate system for the surface.

Instance #700 shows the way to define a bounded surface by creating a *rectangular_trimmed_surface* instance. The first parameter is a name label of the instance; the second parameter is the basis surface where the surface is going to be located on as pointing to #600. The parameters that are represented in real numbers are values of u1, u2, v1 and v2. The last two Boolean parameters are senses to verify if the second parameter values, u2 and v2, consist in an opposite direction of u1 and v1.

```
#600=PLANE('NAME_OF_PLANE', #610);
#610=AXIS2_PLACEMENT_3D('AXES FOR PLANE',#620,#630,#640);
#620=CARTESIAN_POINT('ORIGIN FOR PLANE',(0.,0.,0.));
#630=DIRECTION('local x',(0.,1.,0.);
#640=DIRECTION('local z',(1.,0.,0.);
#700=RECTANGULAR_TRIMMED_SURFACE
(‘SURFACE_NAME’,#600,72.0,72.0,72.0,72.0,.T.,.T.);
```