OVERVIEW OF CIS/2 CONNECTIONS AND JOINTS: FOR DESIGN AND MANUFACTURING MODELS

Chuck Eastman

Introduction

In this note, I lay out the structure of connections and joints within the CIS/2. First, they are presented for a assembly_design, then for a assembly_manufacturing. For each cluster of Entities, I present the EXPRESS-G diagram, followed by the corresponding EXPRESS code, expanding the set of attributes of an Entity resulting from inheritance, into a flattened structure. This is followed by an example set of instance Entities in Part 21 file format. The flattened structure format is where the inherited attributes are identified by parentheses, followed by the inherited Entity name, followed by a colon, followed by the inherited attributes. These are often nested several deep. The format is close to that produced in a Part 21 file. At each level, I provide some discussion on the intended use of the model. With this redundancy and multiple presentations, the structure should become clear from a careful reading. It is assumed that readers of this report are familiar with EXPRESS and EXPRESS-G.

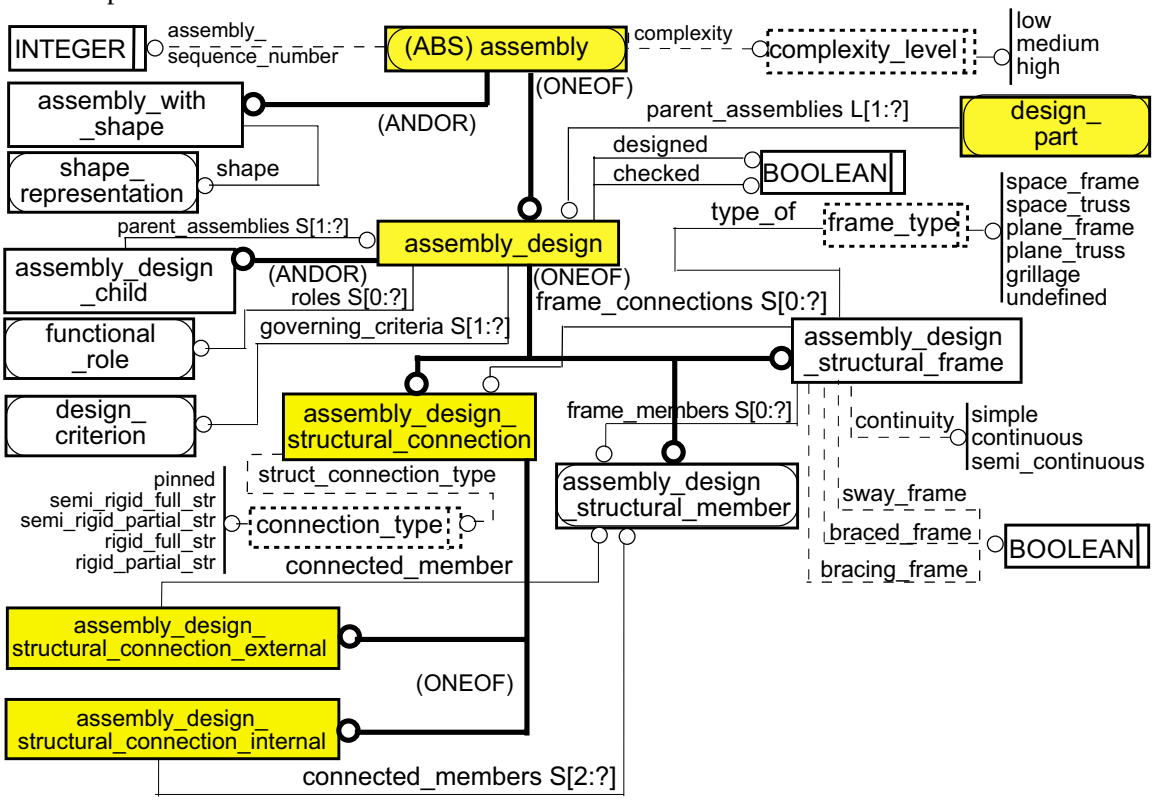

Figure One: Overview of the high-level structure that connections are part of.

Top Level Connection Definition – assembly_design

The top level description of connections is in the Design Model, defined by assembly_design_structural_connection. The Express-G model is shown in Figure One. This Entity inherits attributes from its parents: from assembly_design — designating its function, its design criterion and its completion status; from assembly— an identifier and complexity value; from structural_frame_product—life cycle data; and structural_frame_item — an identifier, name and optional description. Each connection carries a connection_type. Assembly_design_structural_connection is subclassed into either an external connection—for example, a column to the ground— or internal. An external connection only connects one member, while an internal connection is between two or more members. Notice that the emphasis of the connection Entity is a functional description. It does not describe "how" its functions are accomplished. The flattened Express code for internal connections is shown below.

```
ENTITY assembly_design_structural_connection_internal
       (assembly_design_structural_connection:
         (assembly_design:
            (assembly:
               (structural_frame_product:
                  (structural_frame_item:
                  item_number : INTEGER;
                  item_name : label;
                  item_description : OPTIONAL text;
                  );
               life_cycle_stage : OPTIONAL label;
               );
            assembly_sequence_number : OPTIONAL INTEGER;
            complexity : OPTIONAL complexity_level;
            );
          designed : BOOLEAN;
          checked : BOOLEAN;
          roles : SET [0:?] OF functional_role;
          governing_criteria : SET [0:?] OF design_criterion;
          );
      struc_connection_type : OPTIONAL connection_type;
      );
      connected_members : SET [2:?] OF
assembly_design_structural_member;
END_ENTITY;

ENTITY functional_role
SUPERTYPE OF (functional_role_documented);
        functional_role_name : label;
        functional_role_description : text;
INVERSE
        role_for_assemblies : SET [1:?] OF assembly_design FOR roles;
END_ENTITY;

ENTITY design_criterion
SUPERTYPE OF (design_criterion_documented);
        criterion_name : label;
        criterion_description : text;
        design_assumptions : OPTIONAL text;
INVERSE
        governed_assemblies : SET [1:?] OF assembly_design FOR
governing_criteria;
```

```
END_ENTITY;
```

Notice that although there are many Entities involved in assembly_design_structural_connection, it is one large Entity, with 5 optional and 7 required attributes. Of the 7 required attributes, two are lists that may be empty. It uses two complex properties to define the function of the connection: zero or more functional_roles and zero or more design_criterion. It also identifies the members being connected. Its structural is similar to assembly_design_structural_member, described in the *Design Tutorial*. Example P21 file listings are shown below.

```
#121= ASSEMBLY_DESIGN_STRUCTURAL_CONNECTION_INTERNAL(132,'A132',
$,$,$,$,.TRUE.,FALSE.,(#145),(#146),$,(#1451,#1455));
#145= FUNCTIONAL_ROLE('BAY FRAME CONNECTION','CONNECTION');
#146= DESIGN_CRITERION('TYPE 4 STRUCTURE','STORAGE BUILDING FOR
COMBUSTIBLE MATERIALS',$);
```

The two members connected are #1451 and #1455. `Functional_role` and `design_criterion`, though they could have been omitted, are included to show how they might be used.

Connections also may be part of assembly_design_structural_frame, as shown in Figure One. Each connection identifies the members it connects.

Physical Description – Joints

The response to the functional description of a connection is defined with a design_joint_system. Design_joint_system is shown in Figure Two. It provides a link between a generic joint system specification and its multiple instances at different locations. It refers to the generic joint, defined as a joint_system, to one or more parent_assemblies the joint_system is part of, optionally the location of these instance joint_systems, and a name.

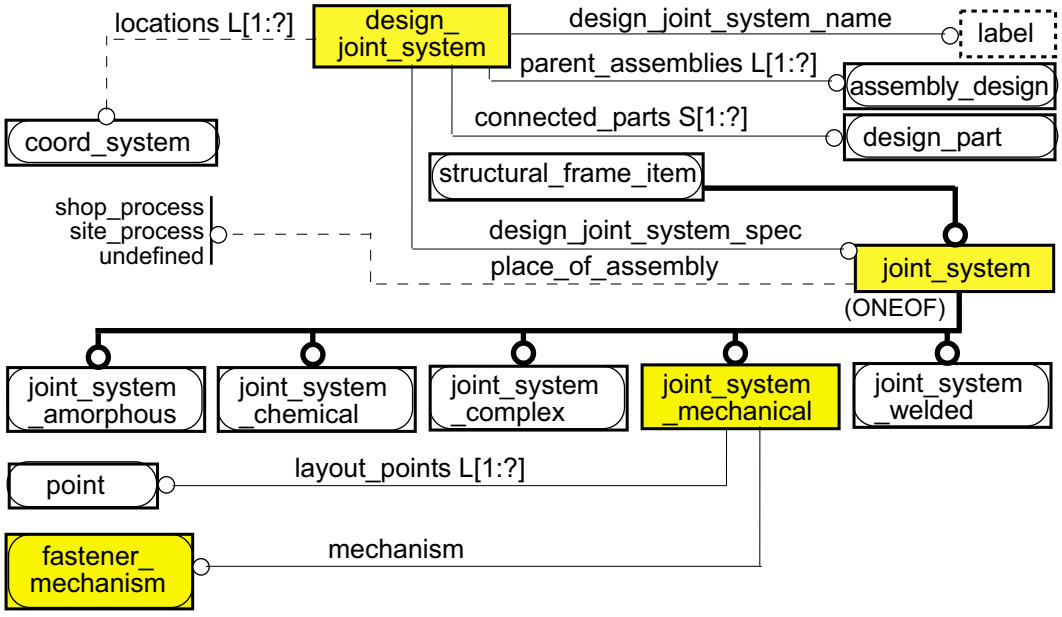

Figure Two: The physical definition of a connection is provided by a design_joint_system.

Design_joint_system has the following Express definition:

```
ENTITY design_joint_system;
      design_joint_system_name : label;
      design_joint_system_spec : joint_system;
      parent_assemblies : LIST [1:?] OF assembly_design;
      locations : OPTIONAL LIST [1:?] OF coord_system;
      connected_parts : SET [0:?] OF design_part;
WHERE
      WRD8 : NOT(EXISTS(locations) AND (SIZEOF(locations) <> SIZEOF
      (parent_assemblies)));
END_ENTITY;
```

Each location of a joint_system defined by design_joint_system has a corresponding parent_assembly. If the same joint_system has multiple locations within the same parent assembly, the new location is identified and a corresponding reference to the parent_assembly added. The WHERE clause requires that there be the same number of locations as parent_assemblys, so they can be matched.

The joint_system is a superclass of the different types of joints: mechanical (bolted), welded, chemical (bonded), amorphous (chemical with filler), or complex (welded studs, involving both welding and mechanical). Joint_system_mechanical will be examined here. It is also shown in Figure Two. Joint_system is subclassed into a type of joint. For the mechanical example, the flattened attributes come from joint_system and structural_frame_item. They are:

```
ENTITY joint_system_mechanical
   (joint_system:
      (structural_frame_item
      item_number : INTEGER;
      item_name : label;
      item_description : OPTIONAL text;
      );
    place_of_assembly : OPTIONAL shop_or_site;
    );
  layout_points : LIST [1:?] OF point;
  mechanism : fastener_mechanism;
END_ENTITY;
```

Below, both the design_joint_system and the subtype of joint_system, joint_system_ mechanical, are presented in the Part 21 file format.

```
#304= DESIGN_JOINT_SYSTEM(2014,#306,(#121),(#1656),$);
#1656= COORD_SYSTEM_CARTESIAN_3D('','assembly coordinate system',$,3,
#67);
#67= AXIS2_PLACEMENT_3D($,#68,#69,#70);
#68= CARTESIAN_POINT($,(712.05911,24.055,284.37498));
#69= DIRECTION($,(0.,0.,1.));
#70= DIRECTION($,(-0.64018438,-0.7682213,0.));
#306= JOINT_SYSTEM_MECHANICAL(2015,'connection',$,.SITE.,
(#1028,#1030,#1032,#1034,#1036,
#1038),#310);
```

Design_joint_system provides some identifiers, then references the joint detail (#306), followed by the functional description(s) of the connection it is a response to (#121), followed by the list of

coordinate systems where the design connection is located. Thus each design_joint_system identifies an assembly_design that it is part of and coordinate system (location plus orientation) where it is placed. The joint_system_mechanical provides an identifier, whether assembled on site and in shop. More important, it provides a list of layout_points that will be used in locating the fastener_mechanisms and their definition.

Fastener mechanisms are defined in Figure Three below.

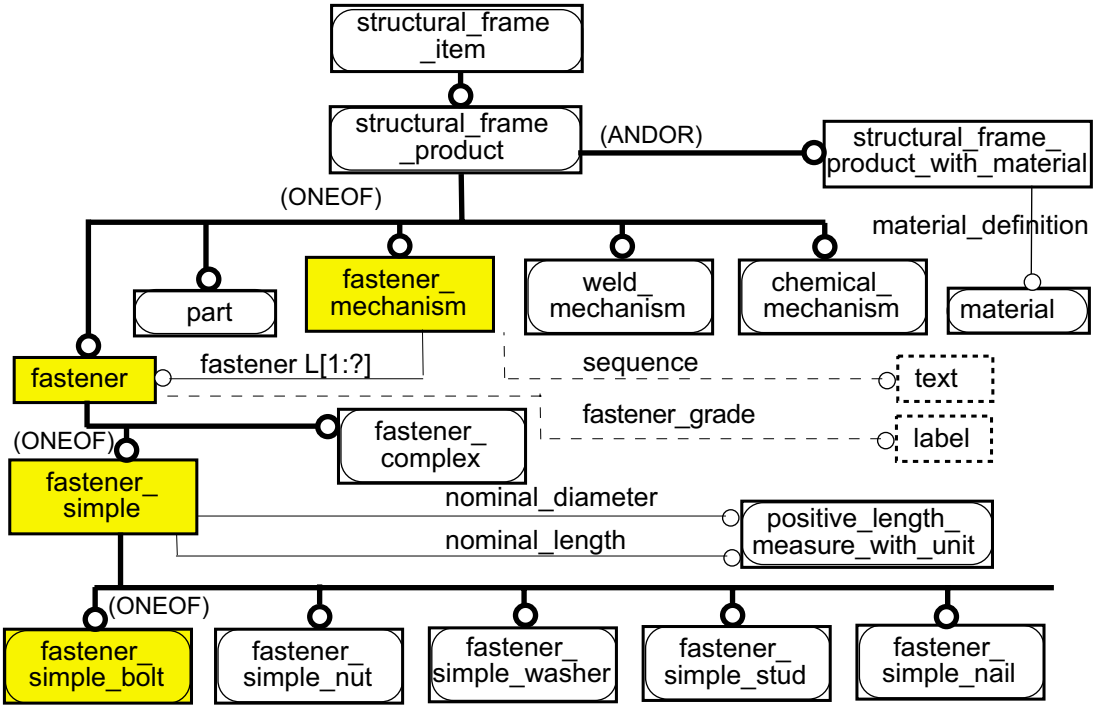

Figure Three: Fastener_mechanisms and fasteners used in joint_system_mechanical.

A fastener mechanism identifies the set of parts that go together to define it, such as bolts, washers and nuts, or washers and sheet metal screws. Thus fastener_mechanism identifies a list of the fasteners that comprise a mechanism. Notice that all the listed fasteners comprise a single mechanism; they are not an assemblies. Fastener mechanisms may be simple or complex. A fastener_mechanism_complex is defined through explicit geometry and seems to exist for special cases. All normal fastener mechanisms are considered simple, and include nuts and bolts, threaded studs, pins, nails and other driven connections, self-taping and other types of srews. Double-ended u-bolts are also supported. These can be defined at various levels of detail, up to very high levels of detail.

Fastener_mechanisms are assumed to be predefined, then referenced as needed for different joint_systems. A fastener_mechanism inherits a life_cycle attribute (e.g., “as planned, “as designed”, “as built”) from structural_frame_ product and has the following definition:

```
ENTITY fastener_mechanism
   (structural_frame_product:
      (structural_frame_item:
        item_number : INTEGER;
```

```
        item_name : Label;
        item_description : OPTIONAL Text;
        );
    life_cycle_stage : OPTIONAL label;
    );
      sequence : OPTIONAL text;
      fasteners : LIST [1:?] OF fastener;
END_ENTITY;
```

It inherits identifiers from structural_frame_product and an optional life cycle stage from structural_frame_product. The sequence of assembling the list of fasteners can optionally be defined in the sequence attribute.

A fastener may be defined without any detail, with diameter and length attributes, as shown in Figure Three. Alternatively, fastener_simple_bolt or fastener_simple_nut may be specified and somewhat more detail may be added. For bolts, details of the head can also be specified, if needed, through subclassing. The Express-G example of nuts and bolts are shown in Figure Four, below, with details for hexagonal and square headed bolts. It should be emphasized that the lowst level of detail for bolts should be specified only when needed for the job.

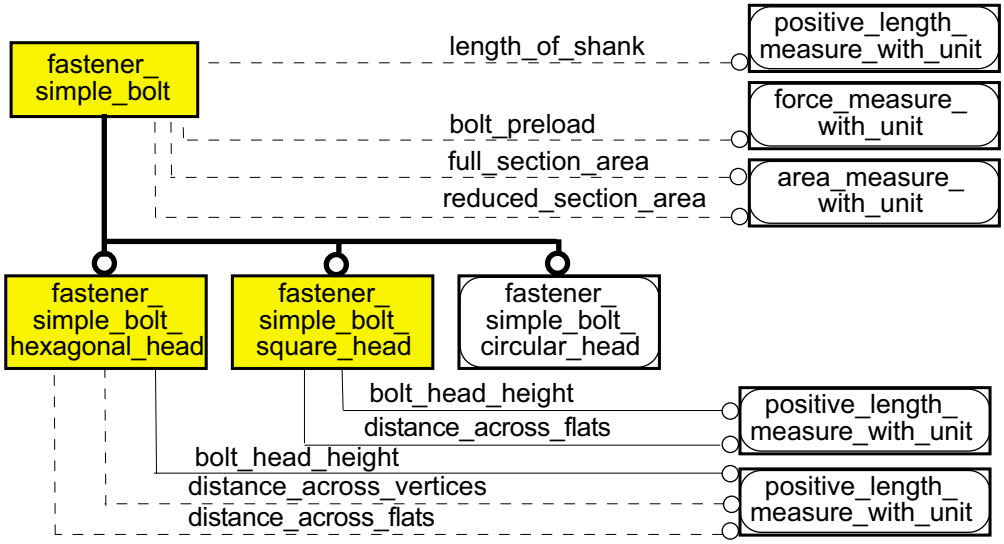

Figure Four: Details that can be specified for bolts.

Examples of flattened Express code for the three levels of bolt are shown below.

```
ENTITY fastener_simple
  (fastener:
      (structural_frame_product:
      (structural_frame_item:
        item_number : INTEGER;
        item_name : Label;
        item_description : OPTIONAL Text;
        );
      life_cycle_stage : OPTIONAL label;
      );
   fastener_grade : OPTIONAL label;
   );
```

```
 nominal_diameter : positive_length_measure_with_unit;
 nominal_length : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;
```

Fasteners_simple requires only a diameter attribute, but optionally provides for a length, life cycle stage and grade labels. Using the structural_frame_product_with_material, a full material spec may also be provided.

A second level of detail may be provided using fastener_simple_bolt. It adds four additional attributes, for length_of_shank, bolt_preload, and two bearing pressure areas. We also list fastener_simple_nut and fastener_simple_washer. The flattened Express code is:

```
ENTITY fastener_simple_bolt
  (fastener_simple:
      (fastener:
          (structural_frame_product:
            (structural_frame_item:
              item_number : INTEGER;
              item_name : Label;
              item_description : OPTIONAL Text;
              );
          life_cycle_stage : OPTIONAL label;
          );
      fastener_grade : OPTIONAL label;
      );
   nominal_diameter : positive_length_measure_with_unit;
   nominal_length : OPTIONAL positive_length_measure_with_unit;
   );
length_of_shank : OPTIONAL positive_length_measure_with_unit;
bolt_preload : OPTIONAL force_measure_with_unit;
full_section_area : OPTIONAL area_measure_with_unit;
reduced_section_area : OPTIONAL area_measure_with_unit;
END_ENTITY;

ENTITY fastener_simple_nut
SUBTYPE OF (fastener_simple);
  (fastener_simple:
      (fastener:
          (structural_frame_product:
            (structural_frame_item:
              item_number : INTEGER;
              item_name : Label;
              item_description : OPTIONAL Text;
              );
          life_cycle_stage : OPTIONAL label;
          );
      fastener_grade : OPTIONAL label;
      );
   nominal_diameter : positive_length_measure_with_unit;
   nominal_length : OPTIONAL positive_length_measure_with_unit;
   );
WHERE
      WRF28 : NOT('STRUCTURAL_FRAME_SCHEMA.FASTENER_SIMPLE_CURVED' IN
      TYPEOF(SELF));
END_ENTITY;
```

```
ENTITY fastener_simple_washer
  (fastener_simple:
      (fastener:
          (structural_frame_product:
            (structural_frame_item:
             item_number : INTEGER;
             item_name : Label;
             item_description : OPTIONAL Text;
             );
          life_cycle_stage : OPTIONAL label;
          );
      fastener_grade : OPTIONAL label;
      );
   nominal_diameter : positive_length_measure_with_unit;
   nominal_length : OPTIONAL positive_length_measure_with_unit;
   );

   washer_shape : OPTIONAL Text;
   inside_diameter : OPTIONAL Positive_Length_Measure_With_Unit;
   external_dimension : OPTIONAL Positive_Length_Measure_With_Unit;
WHERE
    WRF10 : NOT( (EXISTS (INSIDE_DIAMETER) AND EXISTS (EXTERNAL_
            DIMENSION)) AND (INSIDE_DIAMETER.VALUE_COMPONENT >
            EXTERNAL_DIMENSION. VALUE_COMPONENT) );
    WRF11 : NOT( (EXISTS (INSIDE_DIAMETER)) AND (INSIDE_DIAMETER.VALUE_
            COMPONENT < (SELF\Fastener_Simple.NOMINAL_DIAMETER. VALUE_ \
            COMPONENT)) );
END_ENTITY;
```

At this level of detail, the Part21 file entries look like:

```
#3024= FASTENER_MECHANISM(3083,’1/2” BOLT FASTENER’,$,$,(#3012,#3017,
#3021));
#3012= FASTENER_SIMPLE_BOLT(3084,'A325 BOLT 1/2"x1 3/4"',$,$,$,#3014,
#3015,$,$,$,$);
#3014= POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE
(0.5),#4);
#3015= POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(1.75),
#4);
#3017= FASTENER_SIMPLE_NUT($,'HEAVY HEX NUT 1/2"',$,$,$,#3019,$);
#3019= POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(0.5),
#4);
#3021= FASTENER_SIMPLE_WASHER($,'PLAIN WASHER 1/2"',$,$,$,#3023,$,$,$,$
);
#3023=
POSITIVE_LENGTH_MEASURE_WITH_UNIT(POSITIVE_LENGTH_MEASURE(0.5),#4
);
```

The most detailed spec for a bolt can be made using one of the bolt subclasses, such as the hexagonal headed bolt spec below. The hexagonal_head bolt adds three additional attributes

```
ENTITY fastener_simple_bolt_hexagonal_head
 (fastener_simple_bolt:
      (fastener_simple:
          (fastener:
```

```
            (structural_frame_product:
            life_cycle_stage : OPTIONAL label;
            );
          fastener_grade : OPTIONAL label;
          );
      nominal_diameter : positive_length_measure_with_unit;
      nominal_length : OPTIONAL positive_length_measure_with_unit;
      );
   length_of_shank : OPTIONAL positive_length_measure_with_unit;
   bolt_preload : OPTIONAL force_measure_with_unit;
   full_section_area : OPTIONAL area_measure_with_unit;
   reduced_section_area : OPTIONAL area_measure_with_unit;
   );
bolt_head_height : positive_length_measure_with_unit;
distance_across_vertices : OPTIONAL positive_length_measure_with_unit;
distance_across_flats : OPTIONAL positive_length_measure_with_unit;
WHERE
      WRF3 : EXISTS (distance_across_vertices) OR EXISTS
      (distance_across_flats);
      WRF4 : NOT( (distance_across_flats.value_component >
      distance_across_vertices.value_component) AND
      (EXISTS (distance_across_vertices) AND EXISTS
      (distance_across_flats)) );

END_ENTITY;
```

The WRF3 WHERE clause requires that either the distance_across_flats or distance_across_vertices exists. The WRF4 WHERE clause requires that if both exist, the distance across flats must be greater than distance_across_vertices (measured in the same units).

Top Level Connection Definition – assembly_manufacturing

The top level of assembly_manufacturing is different from assembly_design, while its lower level structures are the same. The top- level structure is shown in Figure Five, below.

In a manufacturing model, the top level definitions consist of the same set of identifiers and life cycle stage attributes. Its lower level properties, however, vary from design, with a number of optional attributes: assembly_sequence_number, complexity_level, surface_treatment, assembly_use, assembly_sequence, and place_of_assembly. These all provide production and shop and site assembly information. Flattened, the corresponding EXPRESS code for assembly_manufacturing might look like:

```
ENTITY assembly_manufacturing
SUPERTYPE OF (assembly_manufacturing_child)
      (assembly:
            (structural_frame_product:
                  (structural_frame_item:
                  item_number : INTEGER;
                  item_name : label;
                  item_description : OPTIONAL text;
                  );
            life_cycle_stage : OPTIONAL label;
            );
```

```
      assembly_sequence_number : OPTIONAL INTEGER;
      complexity : OPTIONAL complexity_level;
      );
       surface_treatment : OPTIONAL text;
       assembly_sequence : OPTIONAL text;
       assembly_use : OPTIONAL text;
       place_of_assembly : OPTIONAL shop_or_site;
END_ENTITY;
```

A Part 021 file of assembly_manufacturing might be:

```
#11= ASSEMBLY_MANUFACTURING(10,'B2','CONNECTION',$,$,.MEDIUM.,$,$,$,
     .SHOP_PROCESS.);
```

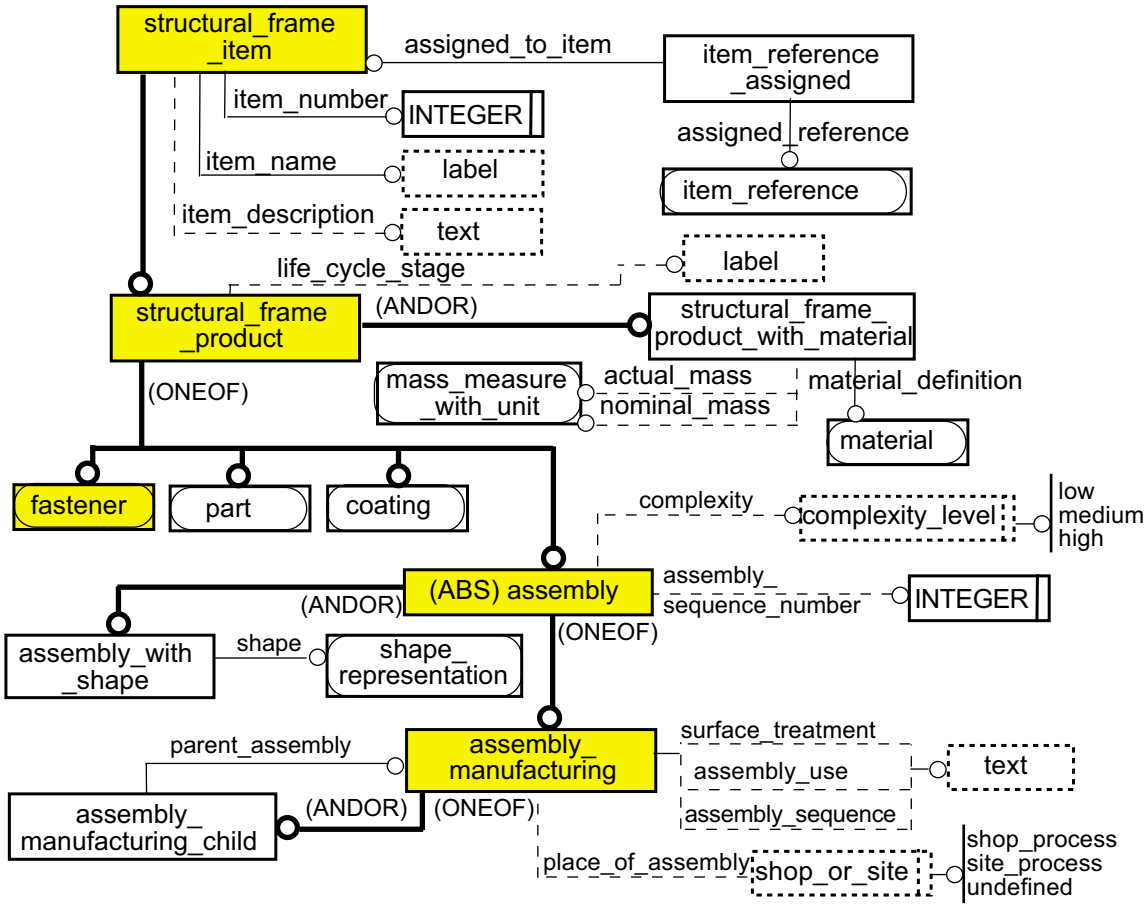

Figure Five: The top-level structure describing a manufacturing model.

These definitions provide broad definitions of the product, which are referenced in the more detailed definitions. The more detailed definitions are provided by located_joint_system, whose definition in EXPRESS-G is shown in Figure Six. It inherits high-level descriptors, a coordinate system location, a parent assembly and a reference to a detailed joint_system. Its flattened EXPRESS code is shown below. Located_assembly is also shown, as it is a required attribute, showing what assembly_ manufacturing the located_joint_system is part of.

```
ENTITY located_joint_system
      (located_item:
```

```
            (structural_frame_item:
            item_number : INTEGER;
            item_name : label;
            item_description : OPTIONAL text;
            );
      location : coord_system;
      );
    descriptive_joint_system : joint_system;
    parent_assembly : located_assembly;
UNIQUE
      URL8 : SELF\located_item.location, descriptive_joint_system,
parent_assembly;
WHERE
      WRL28 : SELF\located_item.location.coord_system_use =
            'Joint System Coordinate System';
      WRL29 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
            TYPEOF (SELF\located_item.location);
      WRL30 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING' IN
            TYPEOF (parent_assembly.descriptive_assembly);
      WRL31 : SELF\located_item.location.parent_coord_system
             :=: parent_assembly\located_assembly\located_item.location;
END_ENTITY;
```

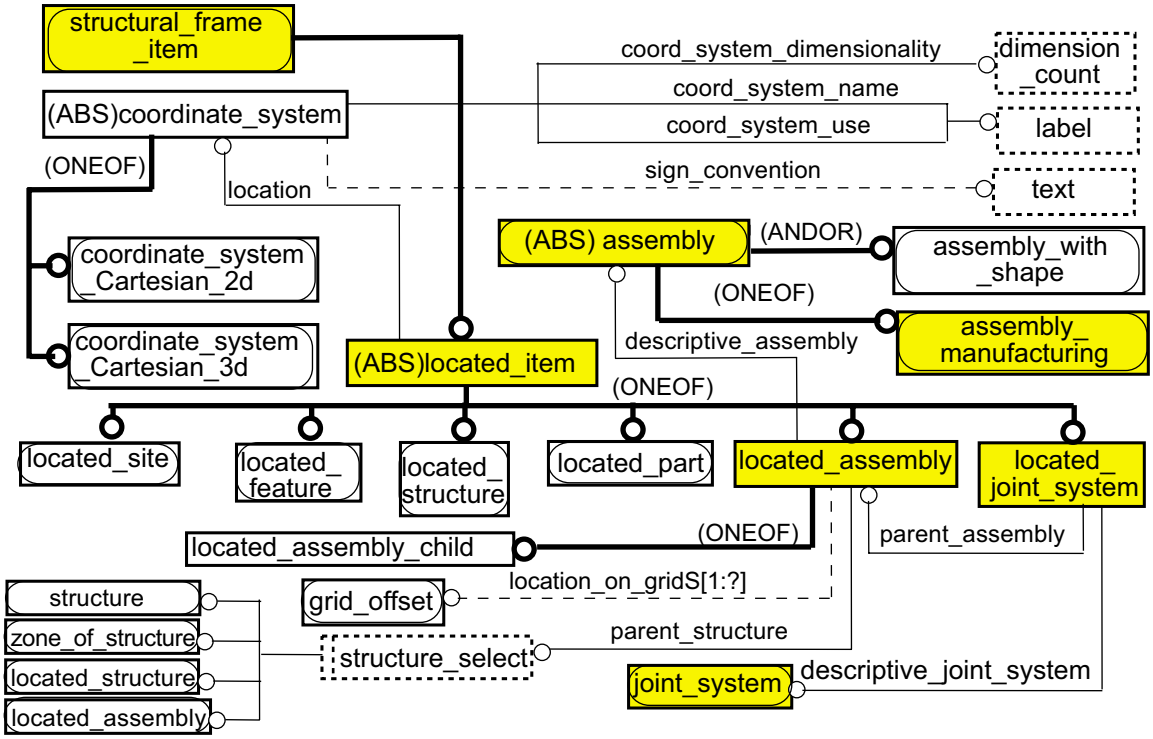

Figure Six: Detail description of a located_joint_system.

```
ENTITY located_assembly
SUPERTYPE OF (located_assembly_child)
      (located_item:
            (structural_frame_item:
            item_number : INTEGER;
```

```
            item_name : label;
            item_description : OPTIONAL text;
            );
      location : coord_system;
      );
    location_on_grid : OPTIONAL SET [1:?] OF grid_offset;
    descriptive_assembly : assembly;
    parent_structure : structure_select;
UNIQUE
      URL2 : SELF\located_item.location, descriptive_assembly,
parent_structure;
WHERE
      WRL22 : SELF\located_item.location.coord_system_use = 'Assembly
Coordinate System';
      WRL46 : parent_structure :<>: (SELF);
END_ENTITY;
```

The corresponding Part 21 file format instance data for a located_joint_system would be as follows:

```
#3044= LOCATED_JOINT_SYSTEM(3911,'1/2” bolt fastener',$,#3045,$,#306,
#761);
#3045= (COORD_SYSTEM('location of 3911','Joint System Coordinate
System',$,3)COORD_SYSTEM_CARTESIAN_3D(#3046));
#3046= AXIS2_PLACEMENT_3D($,#3047,#3048,#3049);
#3047= CARTESIAN_POINT($,(13.39591,5.5,-0.25));
#3048= DIRECTION(‘positive-z’,(0.,0.,1.));
#3049= DIRECTION(‘negative y’,(0.,-1.,0.));

#761= LOCATED_ASSEMBLY(159,'B3',$,#3045,$,#11,#10);
#10= STRUCTURE(10,'Sample Detailed Structure',$);
```

It provides high level descriptors, reference to a coordinate_system for the joint, a reference to a joint_system, and the located_assembly the joint is part of. The joint_system can be the same as, and has the same internal structure as joint_system used in an assembly_design. Thus there is a commonality of structure that can be used to carry the same joint structure, or to refine and replace, as needed. The located_assembly includes the standard identifiers and a coordinate system for the assembly, an optional grid location, a reference the assembly_manufacturing it is part of and its parent structure.

Conceptual Overview:

Given this review, one can see that there are strong similarities between located_joint_systems in a manufacturing model and design_joint_systems in a design model. Their differences and commonalities are shown in Figure Seven. In the design model, the design_joint_ system is the shared detailed description of a joint and its possibly multiple instances, each with their own location and the members it connects. Each design_joint_system then refers to a functional description that identifies its function, design criteria cycle status, and a more detail definition of fasteners, defined in the joint_system.

In a manufacturing model, the located_joint_system refers to the located_assembly it is part of. While an design_joint_system is typically separate from an assembly, a located_joint_ system is always assumed to be part of an located_assembly. Also, a located_joint_system is

always a reference to only a single joint, with one location (in contrast to a design_joint_system that can have many). The production information used in the located_assembly is defined by reference to assembly_ manufacturing. Like design_ joint_systems, a located_joint_ system references a joint_system, for its fasteners and/or welds and also higher level information.

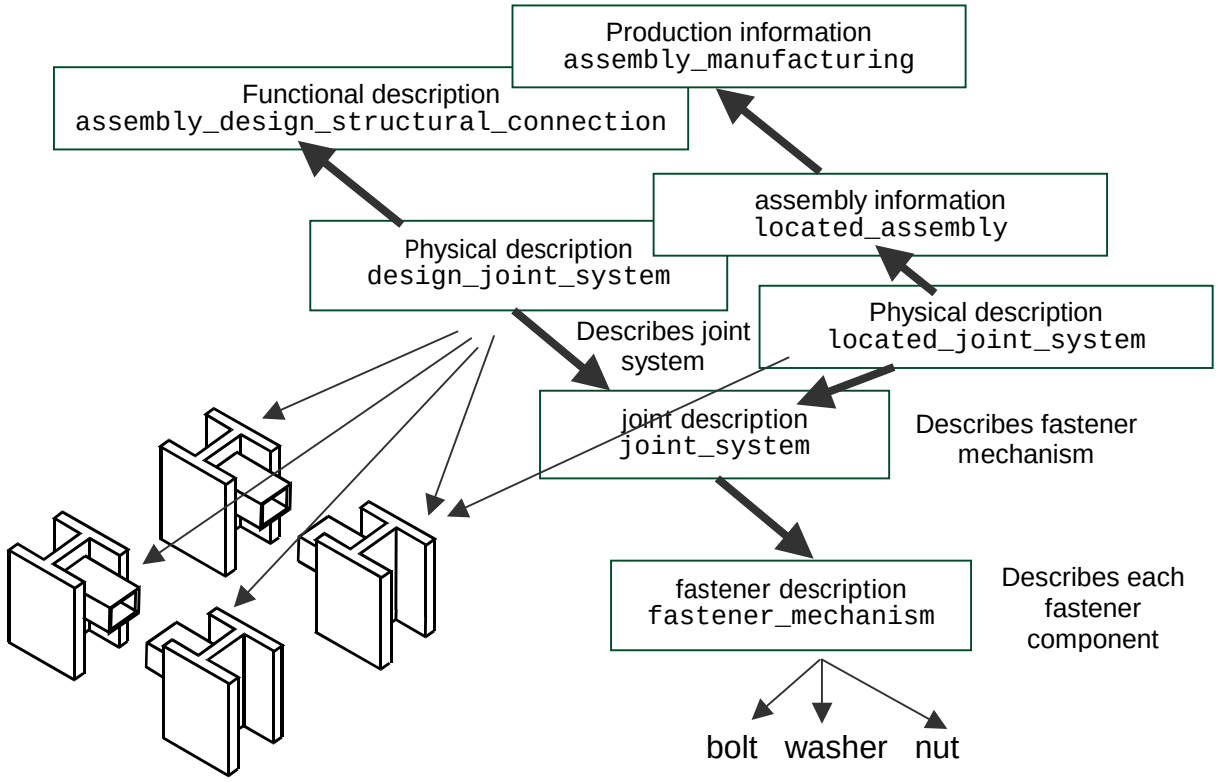

Figure Seven: Conceptual structure of joints and their functional and fasteners mechanisms.