Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the tolerance_zone_definition entity:

— conical_tolerance_zone_definition_constraint (See 5.2.4.68) ;

— tolerance_zone_definition_constraint (See 5.2.4.398) ;

— tolerance_zone_definition_with_per_unit_size_specification_constraint (See 5.2.4.399) .

5.2.3.2.90 tolerance_zone_form

The base definition of the tolerance_zone_form entity is given in ISO 10303-47. The following modifications apply to this Part of ISO 10303.

The definition of tolerance_zone_form is modified as follows:

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the tolerance_zone_form entity:

— tolerance_zone_form_constraint (See 5.2.4.401) .

5.2.3.2.91 versioned_action_request

The base definition of the versioned_action_request entity is given in ISO 10303-41. The following modifications apply to this Part of ISO 10303.

The definition of versioned_action_request is modified as follows:

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the versioned_action_request entity:

— change_request_unique_constraint (See 5.2.4.35) ;

— versioned_action_request_requires_approval (See 5.2.4.411) ;

— versioned_action_request_requires_date_or_date_and_time (See 5.2.4.412) ;

— versioned_action_request_requires_person_organization (See 5.2.4.413);

— versioned_action_request_requires_status (See 5.2.4.414);

— work_request_constraint (See 5.2.4.421);

— work_request_unique_constraint (See 5.2.4.422).

5.2.4 electronic_assembly_interconnect_and_packaging_design rule definitions

NOTE Informal propositions may use the curly brackets, { and }, or the square brackets, [and], to indicate an "AND" relationship. Informal propositions may also use parentheses, (and), to indicate an "OR" relationship.

5.2.4.1 acu_requires_security_classification

The acu_requires_security_classification rule assures that if an instance of assembly_component_usage is also an item of applied_security_classification assignment, then there is a corresponding security classification for the assembly_component_usage.

EXPRESS specification:

```
*)
RULE acu_requires_security_classification FOR
  (assembly_component_usage,
applied_security_classification_assignment);
WHERE

  WR1: SIZEOF (QUERY (acu <* assembly_component_usage |
    NOT (SIZEOF (QUERY (asca <*
    applied_security_classification_assignment |
    acu IN asca.items )) = 1 ))) = 0;
END_RULE;
(*
```


Formal propositions:

WR1: For each instance of assembly_component_usage there shall be exactly one instance of applied_-security_classification_assignment that contains the instance of assembly_component_usage in its set of items.

5.2.4.2 add_design_object_management_relationship_unique_constraint

The add_design_object_management_relationship_unique_constraint rule constrains the add_design_-object_assignment and add_design_object_request_assignment population members, that fill the role of ARM add_design_object_management_relationship, so that the combination of ARM current_design and ARM current_design_object attributes shall be unique within the population.

EXPRESS specification:

```
*)
RULE add_design_object_management_relationship_unique_constraint FOR
 ( add_design_object_assignment, add_design_object_request_assignment );
LOCAL
  pdr_bag : BAG OF product_definition_relationship := [];
  pd_bag : BAG OF product_definition := [];
  adoa_bag : BAG OF add_design_object_assignment;
  adora_bag : BAG OF add_design_object_request_assignment;
  pass : BOOLEAN := TRUE;
  mdo_bag : BAG OF managed_design_object;
END_LOCAL;

REPEAT i := 1 to SIZEOF(add_design_object_assignment) by 1;
  REPEAT j := 1 TO SIZEOF(add_design_object_assignment[i].items) by 1;
 IF ( ('ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
'PRODUCT_DEFINITION_RELATIONSHIP' IN
   TYPEOF(add_design_object_assignment[i].items[j]))
 AND (add_design_object_assignment[i].items[j].name =
         'design object addition') ) THEN
   IF EXISTS( add_design_object_assignment[i].items[j].
                        related_product_definition )
                                                        THEN
     IF( NOT( add_design_object_assignment[i].items[j].
                         related_product_definition
                                          IN pd_bag ) ) THEN
       pd_bag := pd_bag +
                 add_design_object_assignment[i].items[j].
                          related_product_definition;
     END_IF;
   END_IF;
 END_IF;
  END_REPEAT;
END_REPEAT;

REPEAT i := 1 to SIZEOF(add_design_object_request_assignment) by 1;
  REPEAT j := 1 TO
     SIZEOF(add_design_object_request_assignment[i].items) by 1;
 IF ( ('ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
'PRODUCT_DEFINITION_RELATIONSHIP' IN
        TYPEOF(add_design_object_request_assignment[i].items[j]))
AND (add_design_object_request_assignment[i].items[j].name =
                         'design object addition') ) THEN
   IF EXISTS(
        add_design_object_request_assignment[i].items[j].
                         related_product_definition )
                                                        THEN
     IF( NOT(
        add_design_object_request_assignment[i].items[j].
                           related_product_definition
```



```
                                        IN pd_bag ) ) THEN
       pd_bag := pd_bag +
          add_design_object_request_assignment[i].items[j].
                                  related_product_definition;
     END_IF;
   END_IF;
 END_IF;
  END_REPEAT;
END_REPEAT;

REPEAT i := 1 to SIZEOF(pd_bag) by 1;
  IF ( NOT pass ) THEN ESCAPE;
  END_IF;
  mdo_bag := [];
  pdr_bag := QUERY( pdr <* USEDIN(pd_bag[i],
         'ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
                       (pdr.name = 'design object addition') );
  REPEAT j := 1 to SIZEOF(pdr_bag) by 1;
 IF ( NOT pass ) THEN ESCAPE;
 END_IF;
 adoa_bag := QUERY( adoa <* add_design_object_assignment |
                                  (pdr_bag[j] IN adoa.items) );
 REPEAT k := 1 to SIZEOF(adoa_bag) by 1;
   IF ( NOT pass ) THEN ESCAPE;
   END_IF;
   REPEAT l := 1 to SIZEOF(adoa_bag[k].items) by 1;
     IF EXISTS( adoa_bag[k].items[l] ) THEN
       IF ( adoa_bag[k].items[l] IN mdo_bag ) THEN
         pass := FALSE;
         ESCAPE;
       ELSE
         mdo_bag := mdo_bag + adoa_bag[k].items[l];
       END_IF;
     END_IF;
   END_REPEAT;
 END_REPEAT;
  END_REPEAT;
  REPEAT j := 1 to SIZEOF(pdr_bag) by 1;
 IF ( NOT pass ) THEN ESCAPE;
 END_IF;
 adora_bag := QUERY( adora <* add_design_object_request_assignment |
                                        (pdr_bag[j] IN adora.items) );
 REPEAT k := 1 to SIZEOF(adora_bag) by 1;
   IF ( NOT pass ) THEN ESCAPE;
   END_IF;
   REPEAT l := 1 to SIZEOF(adora_bag[k].items) by 1;
     IF ( adora_bag[k].items[l] IN mdo_bag ) THEN
       pass := FALSE;
       ESCAPE;
     ELSE
       mdo_bag := mdo_bag + adora_bag[k].items[l];
```



```
     END_IF;
   END_REPEAT;
 END_REPEAT;
  END_REPEAT;
END_REPEAT;
WHERE

  WR1: pass;
END_RULE;
(*
```

Formal propositions:

WR1: Every instance of add_design_object_assignment and add_design_object_request_assignment that has a common product_definition that is the related_product_definition of a product_definition_relationship that has a name of 'design object addition' where the product_definition_relationship is a member of the items attribute of the add_design_object_assignment and add_design_object_request_assignment must have unique managed_design object entities in their items attributes.

5.2.4.3 adjacent_stratum_surface_definition_constraint

The adjacent_stratum_surface_definition_constraint rule constrains the related and relating shape_aspect of a shape_aspect_relationship when it is used as a adjacent stratum surface definition.

EXPRESS specification:

```
*)
RULE adjacent_stratum_surface_definition_constraint FOR
( shape_aspect_relationship );
WHERE

WR1: SIZEOF (QUERY (sar <* shape_aspect_relationship |
            (sar.name = 'adjacent stratum surface definition') AND
  (NOT(('ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
        'STRATUM_SURFACE' IN TYPEOF (sar.related_shape_aspect)) AND
(sar.related_shape_aspect.description IN ['secondary surface'])))) = 0;

WR2: SIZEOF (QUERY (sar <* shape_aspect_relationship |
            (sar.name = 'adjacent stratum surface definition') AND
  (NOT(('ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
  'STRATUM_SURFACE' IN TYPEOF (sar.relating_shape_aspect)) AND
 (sar.relating_shape_aspect.description IN ['primary surface'])))) = 0;
END_RULE;
(*
```


Formal propositions:

WR1: If shape_aspect_relationship.name = 'adjacent stratum surface definition' the shape_aspect_relationship.related_shape_aspect shall be a stratum_surface with a description of 'secondary surface'.

WR2: If shape_aspect_relationship.name = 'adjacent stratum surface definition' the shape_aspect_relationship.relating_shape_aspect shall be a stratum_surface with a description of 'primary surface'.

5.2.4.4 adjacent_stratum_surface_definition_unique_constraint

The adjacent_stratum_surface_definition_unique_constraint rule constrains shape_aspect_relationship population members, that fill the role of ARM adjacent_stratum_surface_definition, so that the ARM name shall be unique and that the combination of ARM precedent_surface and ARM subsequent_surface attributes shall be unique within the population.

EXPRESS specification:

```
*)
RULE adjacent_stratum_surface_definition_unique_constraint FOR
 ( shape_aspect_relationship );
LOCAL
  assd : BAG OF shape_aspect_relationship :=
   QUERY( sar <* shape_aspect_relationship
| (sar.description = 'adjacent stratum surface definition') );
  pass1 : BOOLEAN := TRUE;
  name_bag : BAG OF STRING := [];
  pss_bag : BAG OF stratum_surface := [];
  sar_bag : BAG OF shape_aspect_relationship;
  pass2 : BOOLEAN := TRUE;
  sss_bag : BAG OF stratum_surface;
END_LOCAL;

REPEAT i := 1 to SIZEOF(assd) by 1;
  IF EXISTS( assd[i].name ) THEN
 IF ( assd[i].name IN name_bag ) THEN
   pass1 := FALSE;
   ESCAPE;
 ELSE
   name_bag := name_bag + assd[i].name;
 END_IF;
  END_IF;
END_REPEAT;

REPEAT i := 1 to SIZEOF(assd) by 1;
  IF EXISTS( assd[i].relating_shape_aspect ) THEN
 IF ('ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
'STRATUM_SURFACE' IN TYPEOF(assd[i].relating_shape_aspect) ) THEN
   IF( NOT( assd[i].relating_shape_aspect IN pss_bag ) ) THEN
     pss_bag := pss_bag + assd[i].relating_shape_aspect;
   END_IF;
 END_IF;
  END_IF;
END_REPEAT;

REPEAT i := 1 to SIZEOF(pss_bag) by 1;
  IF ( NOT pass2 ) THEN ESCAPE;
  END_IF;
  sss_bag := [];
  sar_bag := QUERY( sar <* assd | (sar.relating_shape_aspect :=:
           pss_bag[i]) );
  REPEAT j := 1 to SIZEOF(sar_bag) by 1;
 IF EXISTS( sar_bag[j].related_shape_aspect ) THEN
   IF ('ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
'STRATUM_SURFACE' IN TYPEOF(assd[j].related_shape_aspect) ) THEN
     IF ( sar_bag[j].related_shape_aspect IN sss_bag ) THEN
       pass2 := FALSE;
```



```
        ESCAPE;
      ELSE
        sss_bag := sss_bag + sar_bag[j].related_shape_aspect;
      END_IF;
    END_IF;
  END_IF;
   END_REPEAT;
END_REPEAT;
WHERE

  WR1: pass1;

  WR2: pass2;
END_RULE;
(*
```

Formal propositions:

WR1: Every instance of shape_aspect_relationship with a description of 'adjacent stratum surface definition' must have a unique name attribute.

WR2: Every instance of shape_aspect_relationship with a description of 'adjacent stratum surface definition' must have a unique combination of relating_shape_aspect and related_shape_aspect attributes.

5.2.4.5 aggregate_connectivity_requirement_unique_constraint

The aggregate_connectivity_requirement_unique_constraint rule constrains product_definition_relationship population members, that fill the role of ARM aggregate_connectivity_requirement, so that the ARM design_definition_path attribute shall be unique within the population.

EXPRESS specification:

```
*)
RULE aggregate_connectivity_requirement_unique_constraint FOR
 ( product_definition_relationship );
LOCAL
  acr : BAG OF product_definition_relationship :=
                 QUERY( pdr <* product_definition_relationship |
                 (pdr.name = 'aggregate connectivity requirement') );
  pass : BOOLEAN := TRUE;
  pd_bag : BAG OF product_definition := [];
END_LOCAL;

REPEAT i := 1 to SIZEOF(acr) by 1;
  IF EXISTS( acr[i].related_product_definition ) THEN
 IF ( acr[i].id = 'design composition path' ) THEN
   IF ( acr[i].related_product_definition IN pd_bag ) THEN
     pass := FALSE;
     ESCAPE;
   ELSE
     pd_bag := pd_bag + acr[i].related_product_definition;
   END_IF;
 END_IF;
  END_IF;
END_REPEAT;
WHERE

  WR1: pass;
END_RULE;
(*
```


Formal propositions:

WR1: Every instance of product_definition_relationship with a name of 'aggregate connectivity requirement' must have a unique related_product_definition attribute that points to a product_definition with an id of 'design composition path'.

5.2.4.6 analytical_model_port_unique_constraint

The analytical_model_port_unique_constraint rule constrains the analytical_model_port population members, that fill the role of ARM analytical_model_port, so that the combination of ARM port_name and ARM accessed_analytical_model attributes shall be unique within the population.

EXPRESS specification:

```
*)
RULE analytical_model_port_unique_constraint FOR
 ( analytical_model_port );
LOCAL
  name_bag : BAG OF STRING := [];
  amp_bag : BAG OF analytical_model_port;
  rr_bag : BAG OF representation_relationship;
  pass : BOOLEAN := TRUE;
  am_bag : BAG OF analytical_model;
END_LOCAL;

REPEAT i := 1 to SIZEOF(analytical_model_port) by 1;
  IF EXISTS( analytical_model_port[i].name ) THEN
 IF( NOT( analytical_model_port[i].name IN name_bag ) ) THEN
   name_bag := name_bag + analytical_model_port[i].name;
 END_IF;
  END_IF;
END_REPEAT;

REPEAT i := 1 to SIZEOF(name_bag) by 1;
  IF ( NOT pass ) THEN ESCAPE;
  END_IF;
  amp_bag := QUERY( amp <* analytical_model_port |
       (amp.name = name_bag[i]) );
  am_bag := [];
  REPEAT j := 1 to SIZEOF(amp_bag) by 1;
 IF ( NOT pass ) THEN ESCAPE;
 END_IF;
 rr_bag := QUERY( rr <* USEDIN( amp_bag[j],
            'ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.'
+ 'REPRESENTATION_RELATIONSHIP.REP_2' ) | ((rr.name = 'access mechanism')
 AND
('ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.'
+ 'ANALYTICAL_MODEL' IN TYPEOF(rr.rep_1))) );
 REPEAT k := 1 to SIZEOF(rr_bag) by 1;
   IF EXISTS( rr_bag[k].rep_1 ) THEN
     IF ( rr_bag[k].rep_1 IN am_bag ) THEN
       pass := FALSE;
       ESCAPE;
     ELSE
       am_bag := am_bag + rr_bag[k].rep_1;
     END_IF;
   END_IF;
 END_REPEAT;
  END_REPEAT;
END_REPEAT;
WHERE
```



```
  WR1: pass;
END_RULE;
(*
```

Formal propositions:

WR1: Every instance of analytical_model_port must have a unique combination of name attribute and analytical_model entity assigned to the analytical_model_port by a representation_relationship with a name of 'access mechanism'.

5.2.4.7 analytical_model_vector_port_assignment_constraint

The analytical_model_vector_port_assignment_constraint rule constrains the definition of the property_definition_representation to ensure that there will be an ordered list of assigned_functional_unit_terminals.

EXPRESS specification:


```
*)
(*
RULE analytical_model_vector_port_assignment_constraint FOR
(property_definition_representation);
END_RULE;
*)
(*
```


Informal propositions:

IP1: IF the property_definition_representation.used_representation [is an analytical_model_port] [is an items of a group_assignment that has a name = ('vector port') ('digital vector port')] THEN property_definition_representation.definition.description shall be an integer greater than or equal to 0.

IP2: IF the property_definition_representation.used_representation [is an analytical_model_port] [is an items of a group_assignment that has a name = ('vector port') ('digital vector port')] THEN the combination of the property_definition_representation.used_representation and property_definition_representation.definition.description shall be unique.

5.2.4.8 angular_dimension_with_direction_vector_unique_constraint

The angular_dimension_with_direction_vector_unique_constraint rule constrains the angular_dimension_with_orientation population members, that fill the role of ARM angular_dimension_with_direction_vector, so that the ARM measurement_orientation attribute shall be unique within the population.

EXPRESS specification:

```
*)
RULE angular_dimension_with_direction_vector_unique_constraint FOR
 ( angular_dimension_with_orientation );
LOCAL
  p_bag : BAG OF property_definition;
  pdr_bag : BAG OF property_definition_relationship;
  pass : BOOLEAN := TRUE;
  pd_bag : BAG OF property_definition := [];
END_LOCAL;

REPEAT i := 1 to SIZEOF(angular_dimension_with_orientation) by 1;
  IF ( NOT pass ) THEN ESCAPE;
  END_IF;
  p_bag := QUERY( pd <* USEDIN(angular_dimension_with_orientation[i],
'ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
'PROPERTY_DEFINITION.DEFINITION') | (pd.description =
      'dimensional location property') );
  REPEAT j := 1 to SIZEOF(p_bag) by 1;
 IF ( NOT pass ) THEN ESCAPE;
 END_IF;
 pdr_bag := QUERY( pdr <* USEDIN(p_bag[j],
'ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN.' +
'PROPERTY_DEFINITION_RELATIONSHIP.RELATING_PROPERTY_DEFINITION') |
((pdr.name = 'measurement orientation') AND
(pdr.related_property_definition.description =
     'datum based vector orientation')) );
 REPEAT k := 1 to SIZEOF(pdr_bag) by 1;
   IF EXISTS(  pdr_bag[k].related_property_definition ) THEN
     IF ( pdr_bag[k].related_property_definition IN pd_bag ) THEN
       pass := FALSE;
       ESCAPE;
     ELSE
       pd_bag := pd_bag + pdr_bag[k].related_property_definition;
     END_IF;
   END_IF;
 END_REPEAT;
  END_REPEAT;
END_REPEAT;
WHERE


  WR1: pass;
END_RULE;
(*
```


Formal propositions:

WR1: Every instance of angular_dimension_with_orientation must have a unique property_definition entity with a name of 'datum based vector orientation' that is related to the angular_dimension_with_-orientation by a property_definition_relationship with a name of 'measurement orientation'.

5.2.4.9 angular_size_dimension_constraint

The angular_size_dimension_constraint rule constrains the use of an angular_size when it is used as an angular size dimension.

EXPRESS specification:

```
*)
(*
RULE angular_size_dimension_constraint FOR
  (angular_size);
END_RULE;
*)
(*
```


Informal propositions:

IP1: Each angular_size shall be represented by exactly zero shape_dimension_representation that contain at least one representation_item that has a name that is neither 'full angle' nor 'half angle'.

IP2: Each angular_size shall be represented by exactly zero shape_dimension_representation that contains a representation_item with the name of 'full angle' and a representation_item with a name of 'half angle'.

IP3: Each angular_size shall have a name of 'angular'.

5.2.4.10 application_context_requires_ap_definition

The application_context_requires_ap_definition rule assures that each instance of application_context has an application_protocol_definition with name equal to 'electronic_assembly_interconnect_and_packaging_design'.

EXPRESS specification:

```
*)
RULE application_context_requires_ap_definition FOR
  (application_context, application_protocol_definition);
WHERE
```



```
  WR1: SIZEOF (QUERY (ac <* application_context |
    NOT (SIZEOF (QUERY (apd <* application_protocol_definition |
    (ac :=: apd.application)
    AND
    (apd.application_interpreted_model_schema_name =
      'electronic_assembly_interconnect_and_packaging_design'
    ))) = 1 ))) = 0;
END_RULE;
(*
```

Formal propositions:

WR1: For each instance of application_context, there shall be exactly one instance of application_protocol_definition that references the instance of application_context as its application with a value of 'electronic_assembly_interconnect_and_packaging_design' as its application_interpreted_model_schema_name.

5.2.4.11 approval_requires_approval_date_time

The approval_requires_approval_date_time rule assures that each instance of approval has an associated date or time.

EXPRESS specification:


```
*)
RULE approval_requires_approval_date_time FOR (approval,
      approval_date_time);
WHERE

  WR1: SIZEOF(QUERY ( app <* approval |
    NOT (SIZEOF (QUERY (adt <* approval_date_time |
    app :=: adt.dated_approval)) = 1))) = 0;
END_RULE;
(*
```

Formal propositions:

WR1: For each instance of approval, there shall be exactly one instance of approval_date_time.

5.2.4.12 approval_requires_approval_person_organization

The approval_requires_approval_person_organization rule assures that for each instance of approval there is an organization or person that is responsible for that approval.

