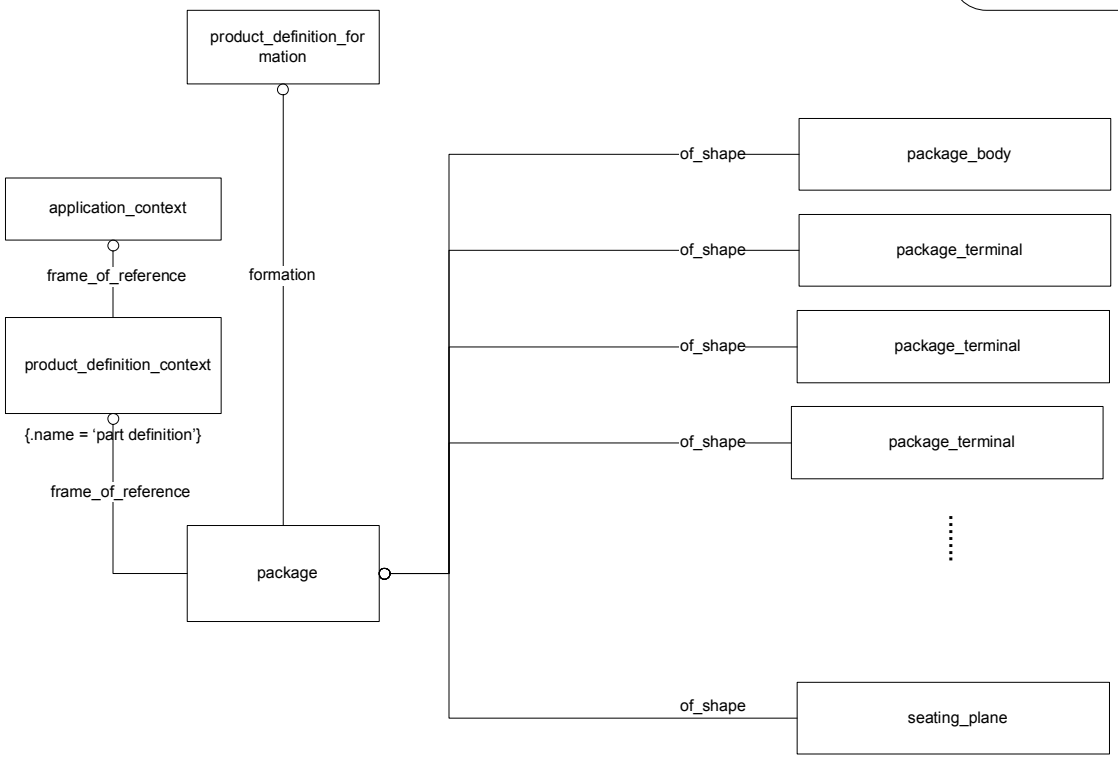
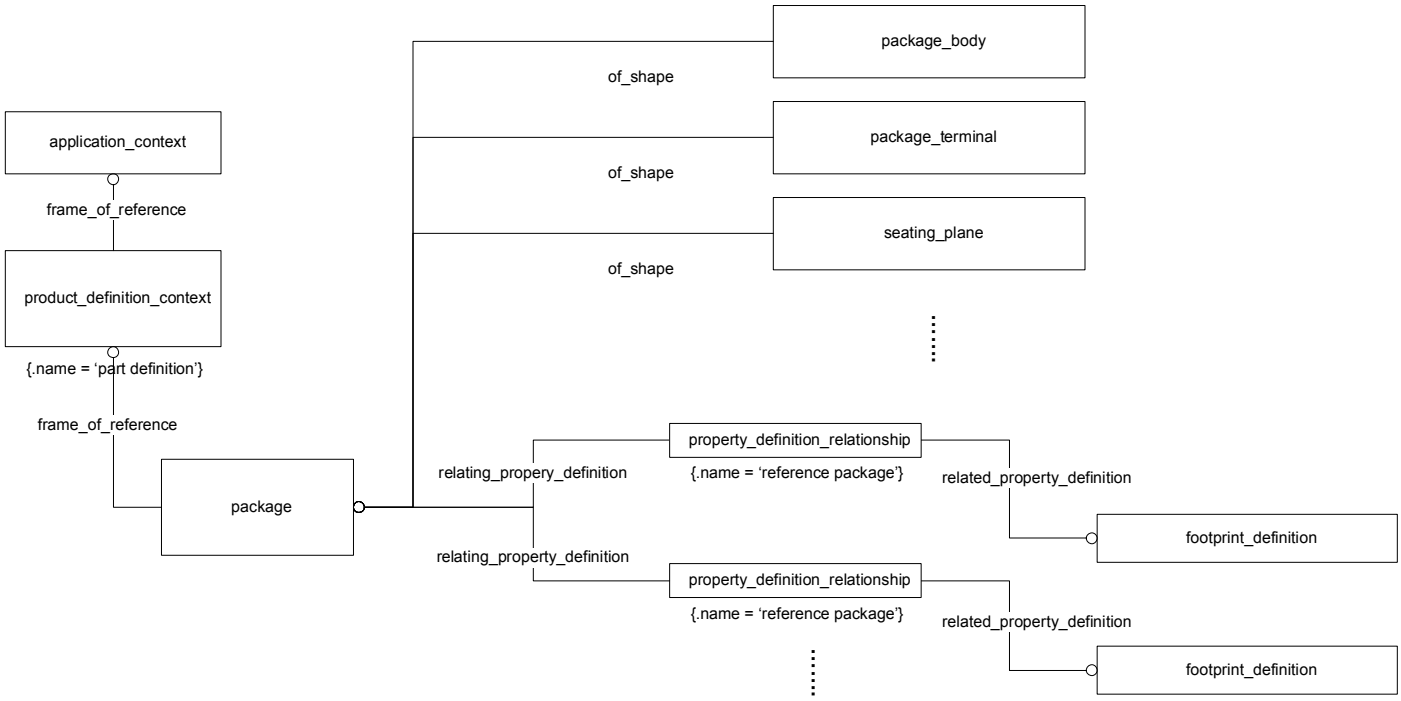
getTerminalsOfPackage	Returns all package_terminals for the given package.
getFootprintsOfPackage	Returns all footprint definitions associated with the given package
getTemplateOfPackageTerminal	Returns the package_terminal_template_definition for the given package_terminal
getShapeRepresentationOfPackageTerminalTemplate	Returns the shape_representation of the given package_terminal_template_definition
getShapeRepresentationOfPackageTerminal	Returns the shape_representation for the given package_terminal
getQualifiedShapeRepresentationOfPackageWithSpecifiedPurpose	Returns a qualified shape_representation with a particular purpose.
get2dDesignShapeRepresentationOfPackage	Returns a shape_representation associated with the given package that satisfies the mapping requirements for a ARM AO Physical_unit_planar_shape_model with a shape purpose of 'design.'
get3dDesignShapeRepresentationOfPackage	Returns a shape_representation associated with the given package that satisfies the mapping requirements for a ARM AO Physical_unit_3d_shape_model with a shape purpose of 'design.'
getShapeRepresentationOfShapeAspect	Finds the shape_representation associated with a shape_aspect of the package such as the package_body or seating_plane.
getPlacementOfShapeAspect	Returns the placement used to position the shape_representation of a shape_aspect with respect to the shape_representation of its containing shape
getBodyOfPackage	Returns the package_body for the given package
getSeatingPlaneOfPackage	Returns the seating_plane for the given package.
getAllTTLISTforST	Returns all template placements composing the given structured_template. These assembly_component_usage reflect the MIM mapping of the ARM AO Template_location_in_structured_template
getPartFeatureForTLIST	Returns a shape_aspect representing the ARM AO Part_feature associated with the given assembly_component_usage corresponding to a Part_feature_based_template_location subtype of Template_location_in_structured_template
getParametricAttributesOfPackage	Returns a specific set of read-only parameters associated with the given package. Provides a simplified interface for accessing the MIM mappings of the parametric attributes of the ARM Package AO.
getParametricAttributesOfTerminalTemplate	Returns a specific set of read-only parameters associated with the given package_terminal_template_definition. Provides a simplified interface for accessing the MIM mappings of the parametric attributes of the ARM Package_terminal_template_definition AO

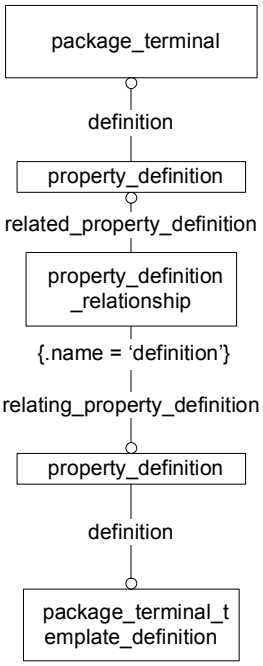



```
// Returns all package_terminals for the given package.
// Returns an empty aggregate if no terminals are found.

Aggregate<package_terminal> getTerminalsOfPackage(package pkg)
{
    Aggregate<package_terminal> a_pkgTerminals = referencingEntitiesOp(pkg)
    where  {package_terminal t}
           {pkg <- t.of_shape}

    return a_pkgTerminals
}
```

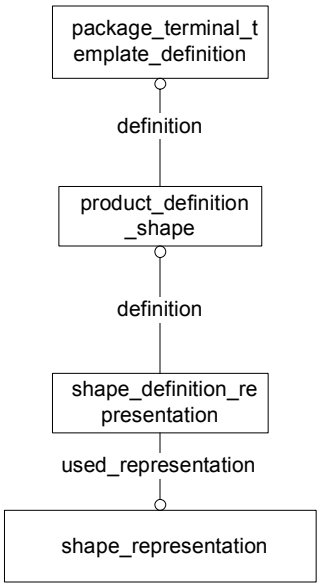
getFootprintsOfPackage



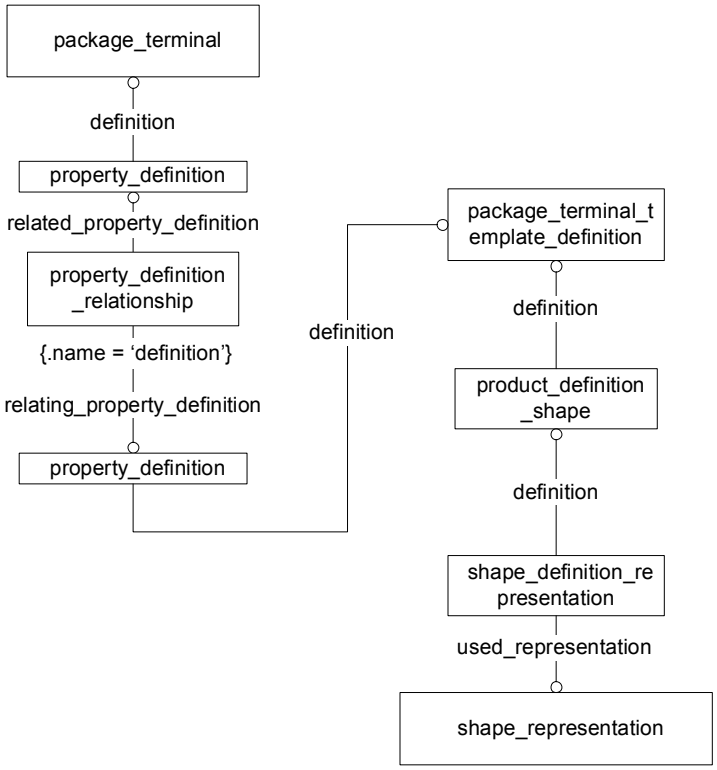

```
// Returns all footprint definitions associated with the given package.
// Returns an empty aggregate if no footprints are found.

Aggregate<footprint_definition> getFootprintsOfPackage(package pkg)
{
    Aggregate<footprint_definition> a_fd = relatedEntitiesOp(pkg)
        where  {footprint_definition e_fd}
            {property_definition_relationship e_pdr}
            {pkg <- e_pdr.relating_product_definition}
            {e_pdr.name = 'reference package'}
            {e_pdr.related_property_definition -> e_fd}

    return a_fd
}
```



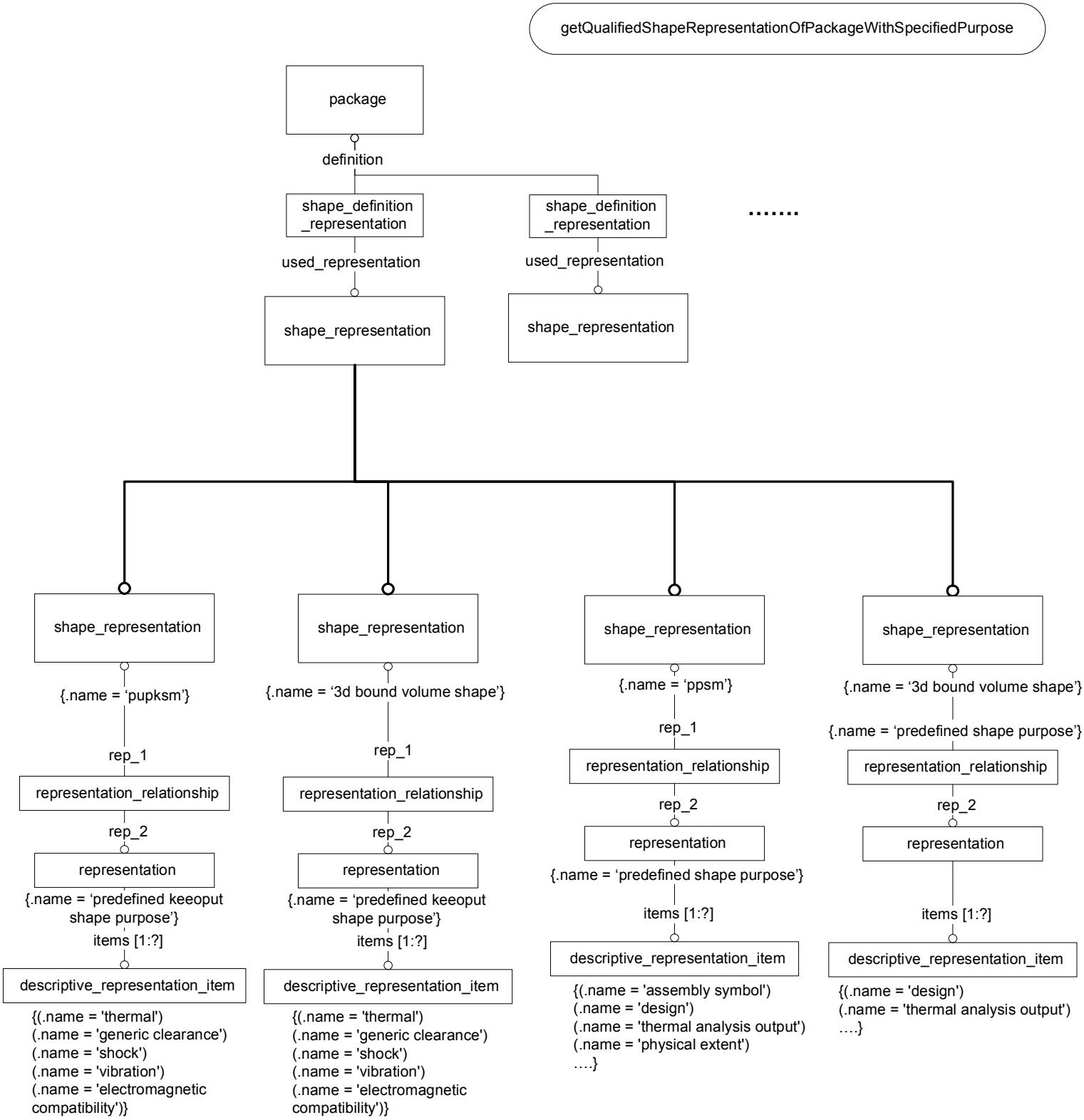

```
// Returns the package_terminal_template_definition for the given package_terminal or
// null if no associated terminal template is found

package_terminal_template_definition getTemplateOfPackageTerminal(package_terminal pt)
{
    Aggregate <property_definition> a_pd1 = referencingEntitiesOp(pt)
        where {property_definition pd}
                {pt <- pd.definition}

    For each property_definition e_pd1 in a_pd1
    {
            property_definition e_pd2 = relatedEntityOp(e_pd1)
                where {property_definition_relationship e_pdr}
                        {e_pd1 <- e_pdr.related_property_definition}
                        {e_pdr.name = 'definition'}
                        {e_pdr.relating_property_definition -> e_pd2}

            if (e_pd2 != null)
            {
                package_terminal_template_definition e_pttd = referencedEntityOp(e_pd2)
                    where {e_pd2.definition -> e_pttd}

                return e_pttd
            }
    }
    return null;
}
```



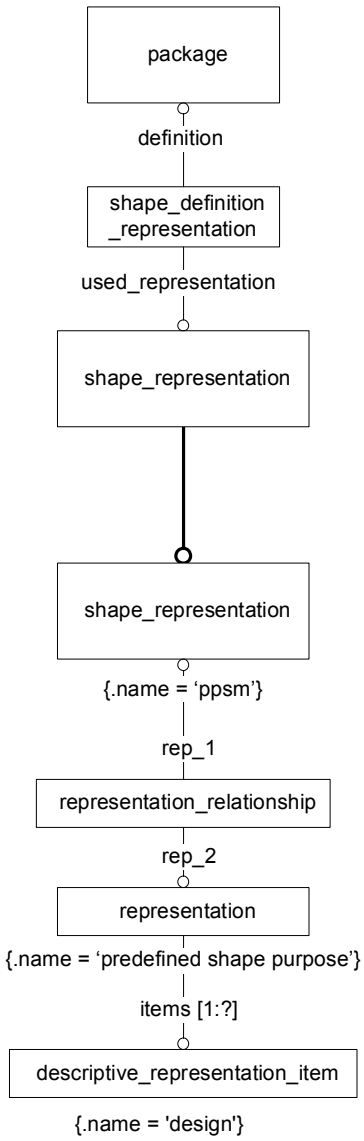

```
// Returns the shape_representation of the given package_terminal_template_definition.
// Implementation is based on the assumption that there is only one shape_representation associated with the template definition.
// Returns null if no associated shape_representation is found

shape_representation getShapeRepresentationOfPackageTerminalTemplate(package_terminal_template_definition e_pttd)
{
	product_definition_shape e_pds = referencingEntityOp(e_pttd)
		where	{ e_pttd <- e_pds.definition }

	if (e_pds != null)
	{
		shape_representation e_sr = relatedEntityOp(e_pds)
			where	{shape_definition_representation e_sdr}
				{e_pds <- e_sdr.definition}
				{e_sdr.used_representation -> e_sr}
		return e_sr;
	}
	return null;
}
```



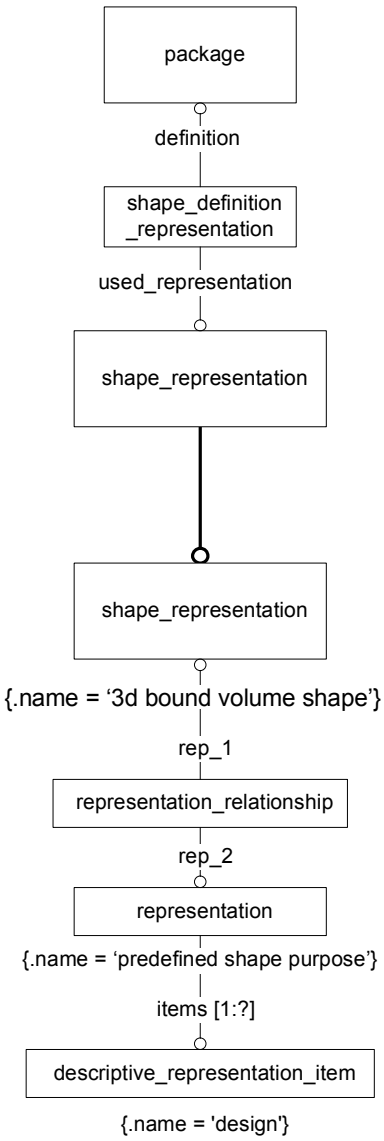

```
// Returns the shape_representation for the given package_terminal.
// Implementation is based on the assumption that there is only one shape_representation associated with the template definition.
// Returns null if no associated shape_representation is found

shape_representation getShapeRepresentationOfPackageTerminal(package_terminal e_pt)
{
    package_terminal_template_definition e_pttd = getTemplateOfPackageTerminal(e_pt);

    if (e_pttd == null)
        return null;

    return getShapeRepresentationOfPackageTerminalTemplate(e_pttd);
}
```


getQualifiedShapeRepresentationOfPackageWithSpecifiedPurpose
package
definition
shape_definition _representation
shape_definition _representation
.....
used_representation
used_representation
shape_representation
shape_representation
shape_representation
shape_representation
shape_representation
shape_representation
{.name = 'pupksm'}
{.name = '3d bound volume shape'}
{.name = 'ppsm'}
{.name = '3d bound volume shape'}
rep_1
rep_1
rep_1
{.name = 'predefined shape purpose'}
representation_relationship
representation_relationship
representation_relationship
representation_relationship
rep_2
rep_2
rep_2
rep_2
representation
representation
representation
representation
{.name = 'predefined keeoput shape purpose'}
{.name = 'predefined keeoput shape purpose'}
{.name = 'predefined shape purpose'}
items [1:?]
items [1:?]
items [1:?]
items [1:?]
descriptive_representation_item
descriptive_representation_item
descriptive_representation_item
descriptive_representation_item
{(.name = 'thermal')
(.name = 'generic clearance')
(.name = 'shock')
(.name = 'vibration')
(.name = 'electromagnetic compatibility')}
{(.name = 'thermal')
(.name = 'generic clearance')
(.name = 'shock')
(.name = 'vibration')
(.name = 'electromagnetic compatibility')}
{(.name = 'assembly symbol')
(.name = 'design')
(.name = 'thermal analysis output')
(.name = 'physical extent')
….}
{(.name = 'design')
(.name = 'thermal analysis output')
….}

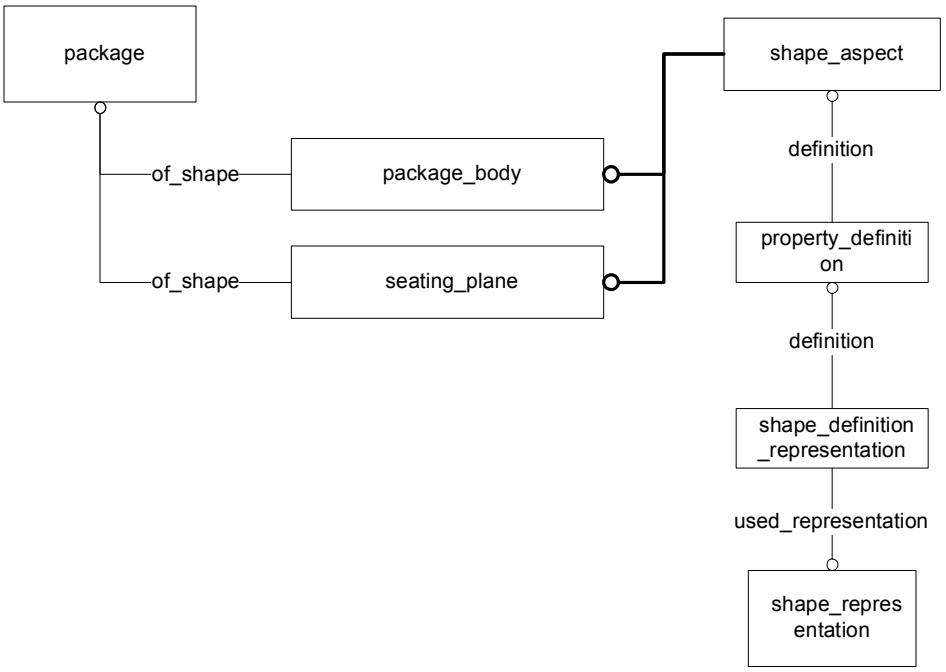
```
// Returns a qualified shape_representation with a particular purpose.
// The qualification on the name of the shape_representation supports the mapping requirements of the subtypes of the
// ARM Geometric_model AO, including the commonly implemented subtypes for 2D and 3D shape models and keepouts:
// Physical_unit_planar_shape_model
// Physical_unit_planar_keepout_shape_model
// Physical_unit_3d_shape_model
// Physical_unit_3d_keepout_shape_model
// The shapePurpose argument supports the mapping of the shape_purpose attribtue of the Physical_unit_3d_shape_model,
// Physical_unit_keepout_shape_model, and Physical_unit_planar_shape_model AOs.
// The method will return a shape_representation with shape_representation.name = qualifyingName
// and shape purpose equal to the given shapePurpose parameter.

shape_representation getQualifiedShapeRepresentationOfPackageWithSpecifiedPurpose(package e_p,
    String qualifyingName, String shapePurpose)
{
    Aggregate<shape_representation> a_sr = relatedEntitiesOp(e_p)
        where {shape_representation e_sr}
              {shape_definition_representation e_sdr}
              {e_pds <- e_sdr.definition}
              {e_sdr.used_representation -> e_sr}

    For each shape_representation e_sr in a_sr
    {
        if (e_sr.name != qualifyingName}
            remove e_sr from a_sr
    }

    return getShapeRepresentationWithSpecifiedPurpose(a_sr, shapePurpose);
}
```



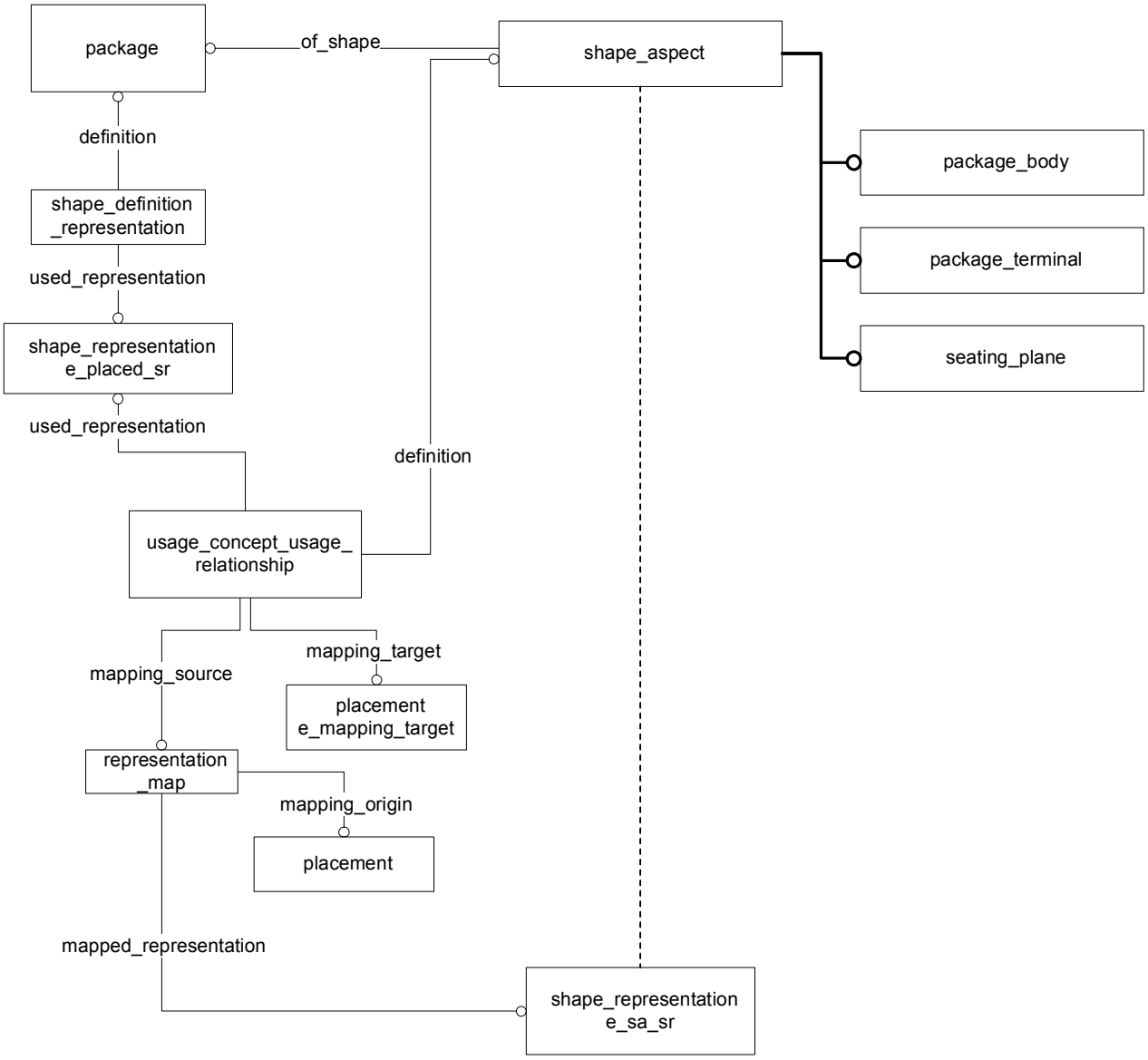

```
// Returns a shape_representation associated with the given package that satisfies the mapping requirements for a
// ARM AO Physical_unit_planar_shape_model with a shape purpose of ‘design.’

shape_representation get2dDesignShapeRepresentationOfPackage(package e_p)
{
    return getQualifiedShapeRepresentationOfPackageWithSpecifiedPurpose(e_p, ‘ppsm’, ‘design’)
}
```



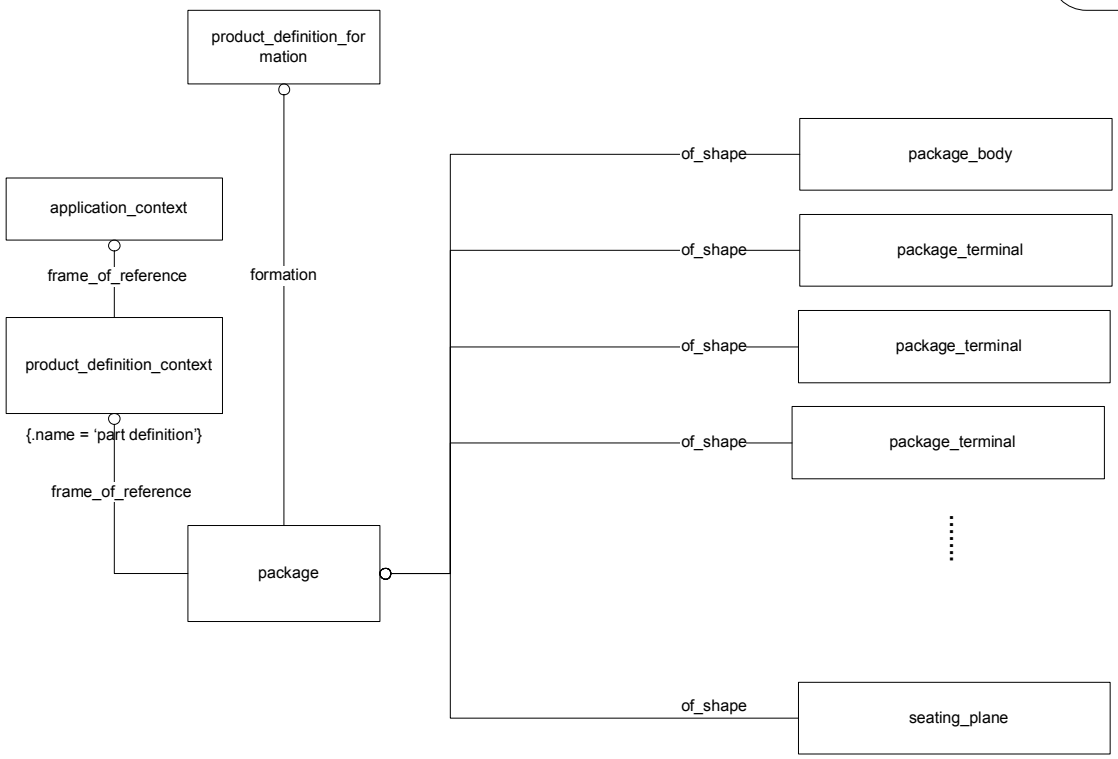

```
// Returns a shape_representation associated with the given package that satisfies the mapping requirements for a
// ARM AO Physical_unit_3d_shape_model with a shape purpose of ‘design.’

shape_representation get2dDesignShapeRepresentationOfPackage(package e_p)
{
    return getQualifiedShapeRepresentationOfPackageWithSpecifiedPurpose(e_p, ‘3d bound volume shape’, ‘design’)
}
```



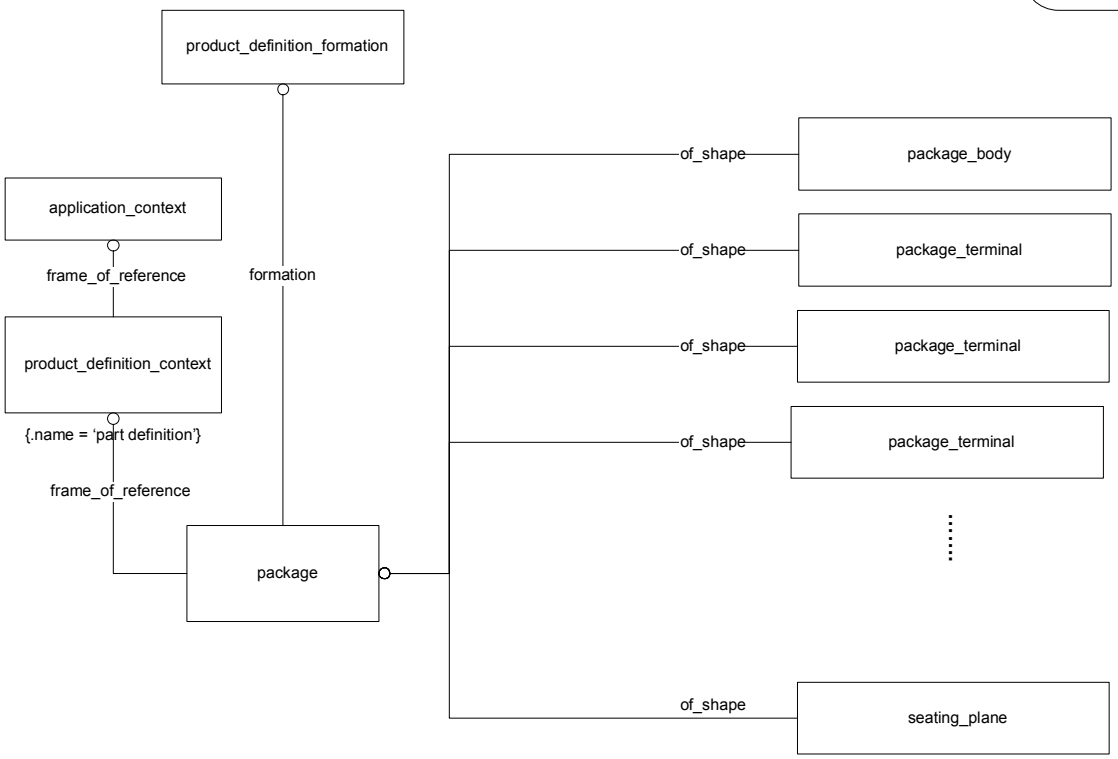

```
// Finds the shape_representation associated with a shape_aspect of the package such as the package_body or seating_plane.
// Assumes a single shape_representation of the shape_aspect.
// Returns null if no shape_representation is found.

shape_representation getShapeRepresentationOfShapeAspect(shape_aspect e_sa)
    {
        Aggregate<property_definition> a_pd1 = referencingEntitiesOp(e_sa)
            where  {property_definition e_pd}
                   {e_sa <- e_pd.definition}

        For each property_definition e_pd in a_pd1
        {
            shape_representation e_sr = relatedEntityOp(e_pd)
                where  {shape_definition_representation e_sdr}
                       {e_pd <- e_sdr.definition}
                       {e_sdr.used_representation -> e_sr}
            if (e_sr != null)
            {
                return e_sr;
            }
        }
        return null;
    }
```


package
of_shape
shape_aspect
definition
shape_definition
_representation
used_representation
shape_representation
e_placed_sr
used_representation
definition
usage_concept_usage_
relationship
mapping_source
mapping_target
placement
e_mapping_target
representation
_map
mapping_origin
placement
mapped_representation
shape_representation
e_sa_sr
package_body
package_terminal
seating_plane

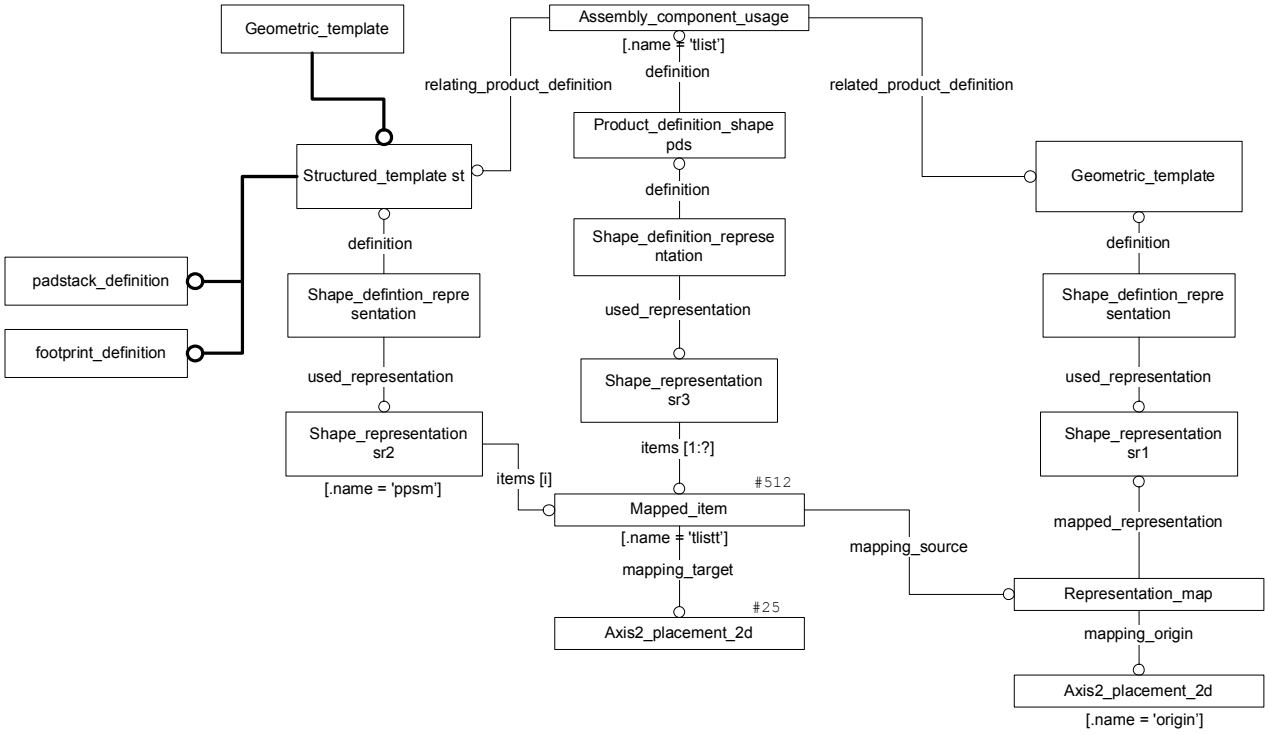
```
// Returns the placement used to position the shape_representation of a shape_aspect with respect to the shape_representation
// of its containing shape. In the context of a package model, the shape_aspect (e_sa_sr) may be a package_terminal,
// package_body, or seating_plane. The containing shape (e_placed_sr) would typically be the design shape of the package.
// The method must be qualified by both the package shape representation and the shape representation of the shape_aspect.
// If the applicable shape_representation are 3d, the returned placement will be of type axis2_placement_3d. If the applicable
shape_representations are planar shape models, the returned placement will be of type axis2_placement_2d

placement getPlacementOfShapeAspect(shape_aspect e_sa, shape_representation e_sa_sr, shape_representation e_placed_sr)
{
    Aggregate<usage_concept_usage_relationship> a_ucur = referencingEntitiesOp(e_sa)
        where {usage_concept_usage_relationship e_ucur}
              {e_sa <- e_ucur.definition}

    usage_concept_usage_relationship e_ucur = referencingEntityOp(a_ucur, e_placed_sr)
        where {usage_concept_usage_relationship e_ucur contained in a_ucur}
              {e_placed_sr <- e_ucur.used_representation}

    if (e_ucur == null)
        return null;

    representation_map e_rep_map = e_ucur.mapping_source
    representation e_mapped_representation = e_rep_map.mapped_representation

    if (e_mapped_representation != e_sa_sr)
        return null;

    representation_item e_mapped_target = e_ucur.mapping_target

    if (e_mapped_target is subtype of placement)
        return e_mapped_target;

    return null;
}
```



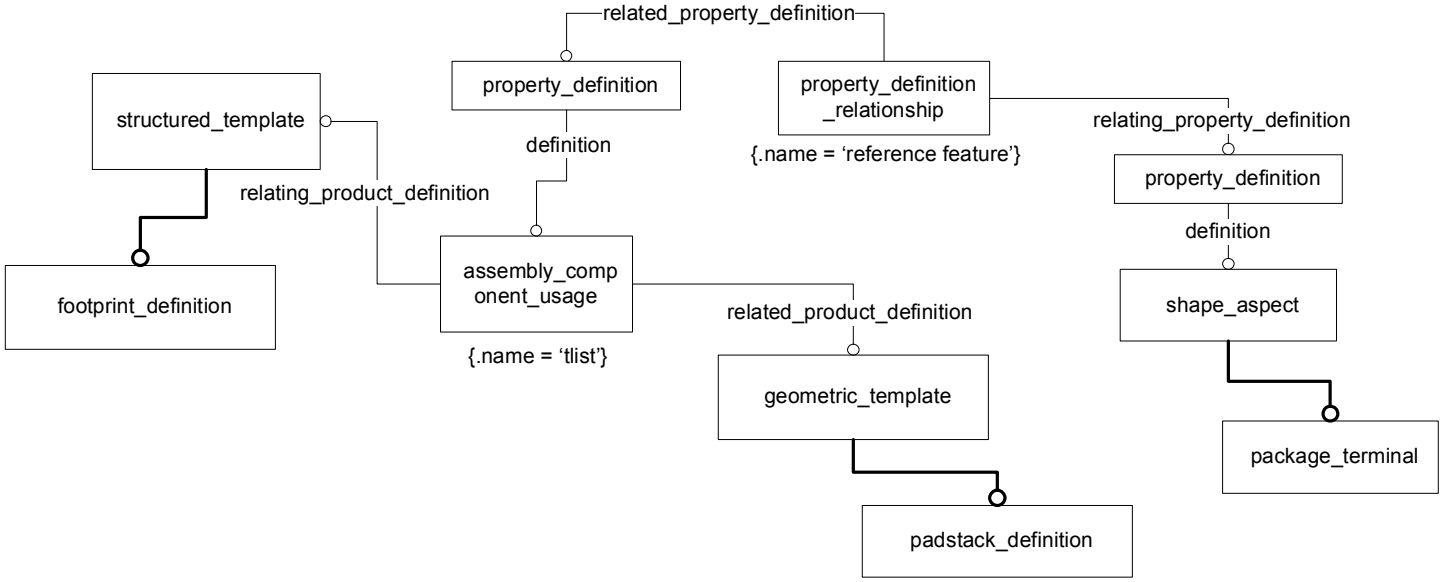

```
// Returns the package_body for the given package.
// Returns null if no package_body is found.

package_body getBodyOfPackage(package pkg)
{
    package_body e_pb = referencingEntityOp(pkg)
    where {pkg <- e_pb.of_shape}

    return e_pb
}
```



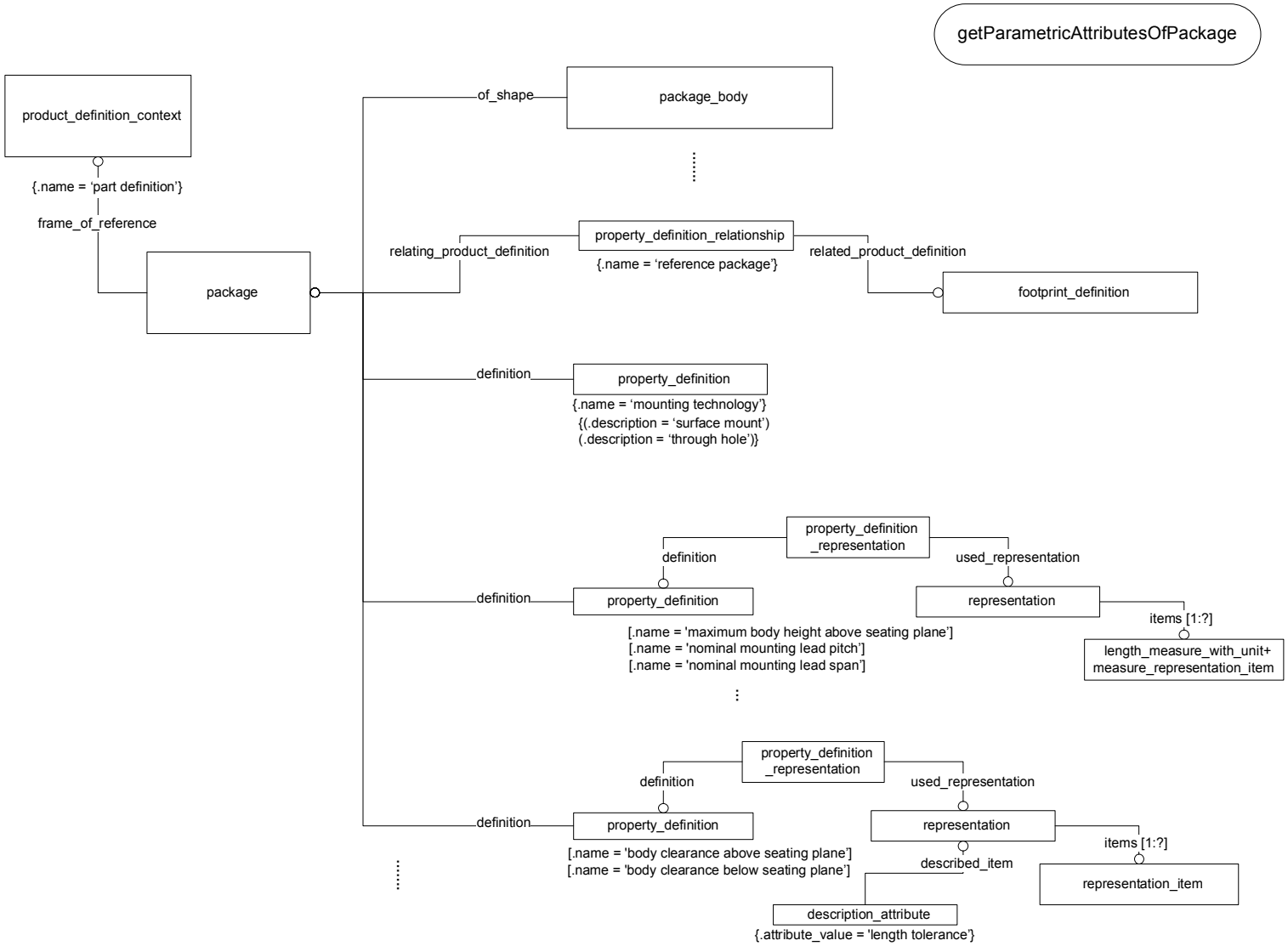

```
// Returns the seating_plane for the given package.
// Returns null if no package_body is found.

package_body getSeatingPlaneOfPackage(package pkg)
{
    seating_plane e_sp = referencingEntityOp(pkg)
    where {pkg <- e_sp.of_shape}

    return e_sp
}
```



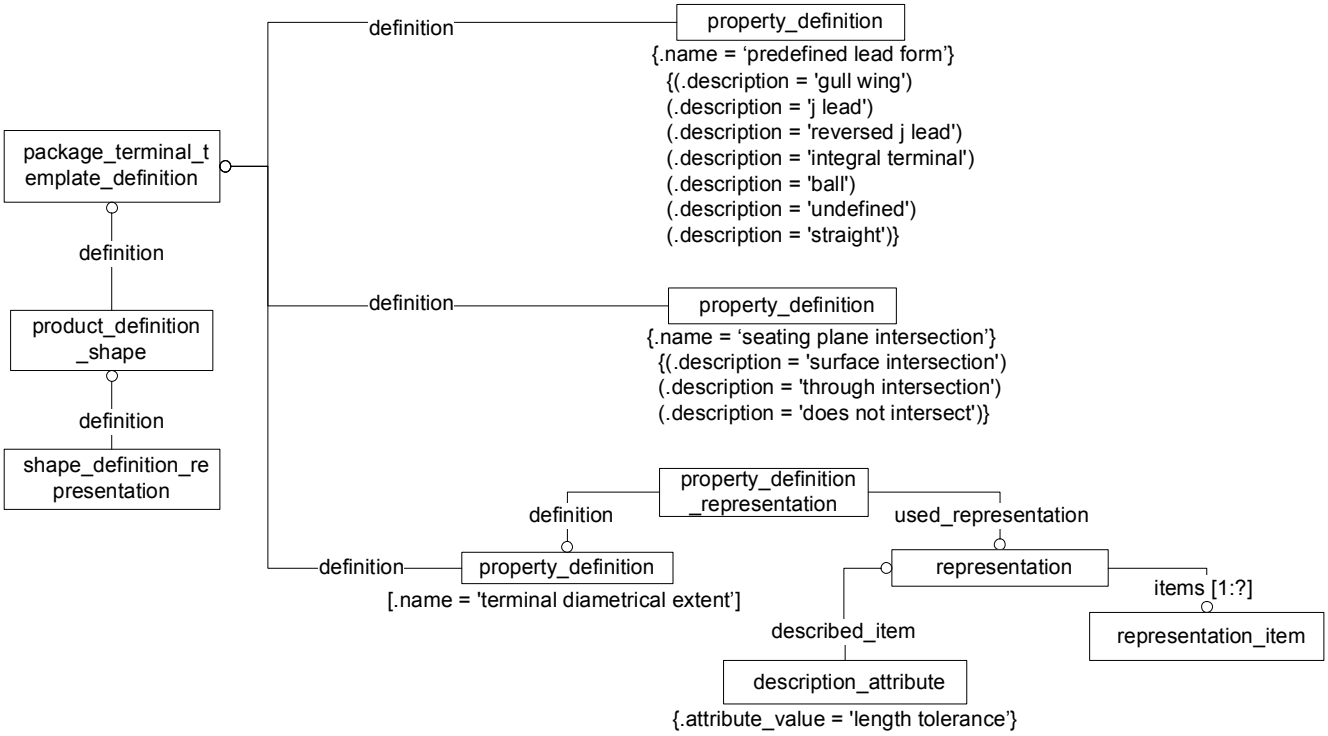

```
// Returns all template placements composing the given structured_template.
// These assembly_component_usage reflect the MIM mapping of the
// ARM AO Template_location_in_structured_template

Aggregate<assembly_component_usage> getAllTLISTforST(structured_template e_st)
{
    Aggregate<assembly_component_usage> a_tlist = referencingEntitiesOp(e_st)
    where  {assembly_component_usage e_acu}
            {e_st <- e_acu.relating_product_definition}
            {e_acu.name = 'tlist'}

    return a_tlist
}
```




```
// Returns a shape_aspect representing the ARM AO Part_feature associated with the given
// assembly_component_usage corresponding to a Part_feature_based_template_location subtype of
// Template_location_in_structured_template.
// In the context of a package model, this query is most commonly used to obtain the package_terminal
// associated with a placement of a padstack_definition within a footprint_definition.

shape_aspect getPartFeatureForTLIST(assembly_component_usage  e_tlist)
{
    Aggregate<property_definition> a_pd = referencingEntitiesOp(e_tlist)
        where  {property_definition e_pd}
               {e_tlist <- e_pd.definition}

    For each property_definition e_pd1 in a_pd
        {
            property_definition e_pd2 = relatedEntityOp(e_pd1)
                where  {property_definition_relationship e_pdr}
                       {e_pd1 <- e_pdr.related_property_definition}
                       {e_pdr.name = 'reference feature'}
                       {e_pdr.relating_property_definition -> e_pd2}

            if (e_pd2 != null)
            {
                EShape_aspect e_sa = (EShape_aspect) e_pd2.getDefinition(null);
                return e_sa;
            }
        }
        return null;
}
```




```
// Returns a specific set of read-only parameters associated with the given package.
// Provides a simplified interface for accessing the MIM mappings of the parametric attributes of the
// ARM Package AO.
// The set of package parameters to be returned is initialized through the addPackageParam method.
// See addPackageParam for a list of supported attributes.
// An empty set is returned if none of the relevant parameters are found.
// The attributes of the ARM Package AO are mapping to instances of property_definition with identifying
// names. Values are extracted directly from the property_definition description in the case of enumeration
// attributes, and from the associated representation in the case of physical measurements.

Aggregate<Param> getParametricAttributesOfPackage(package e_p)
    {
        Aggregate<property_definition> a_pd = referencingEntitiesOp(e_p)
            where {property_definition e_pd}
                  {e_p <- e_pd.definition}

        For each property_definition e_pd in a_pd
        {
            if (e_pd.name is in set of package attributes to be extracted)
            {
                Param p = parametricAttributeForPropertyDefinition(e_pd);
                if not(p == null)
                    add p to set
            }
        }
        return set;
    }
```




```
// Returns a specific set of read-only parameters associated with the given package_terminal_template_definition.
// Provides a simplified interface for accessing the MIM mappings of the parametric attributes of the
// ARM Package_terminal_template_definition AO.
// The set of terminal template parameters to be returned is initialized through the addTerminalParam method.
// See addTerminalParam for a list of supported attributes.
// An empty set is returned if none of the relevant parameters are found.
// The attributes of the ARM Package_terminal_template_definition AO are mapping to instances of property_definition with identifying
// names. Values are extracted directly from the property_definition description in the case of enumeration
// attributes, and from the associated representation in the case of physical measurements.

Set<Param> getParametricAttributesOfTerminalTemplate(package_terminal_template_definition e_pttd)
    {
        Aggregate<property_definition> a_pd = referencingEntitiesOp(e_pttd)
            where  {property_definition e_pd}
                   {e_p <- e_pttd.definition}

        For each property_definition e_pd in a_pd
        {
            if (e_pd.name is in set of terminal template attributes to be extracted)
            {
                Param p = parametricAttributeForPropertyDefinition(e_pd);
                if not(p == null)
                    add p to set
            }
        }
        return set;
    }
```