MIMqueries - Parameters

parametricAttributeForPropertyDefinition

Extracts parametric data from the most commonly used property_definition representations for boolean, textual, count, and measure based parametric data.

allParametersForProduct

Extracts parametric data from the most commonly used implementations of parameter_assignment related to a given product through a product_specific_parameter_value_assignment.

parameterForRepresentationItem

Extracts parametric data from the certain commonly used representation_item representations for boolean, textual, count, and measure based parametric data

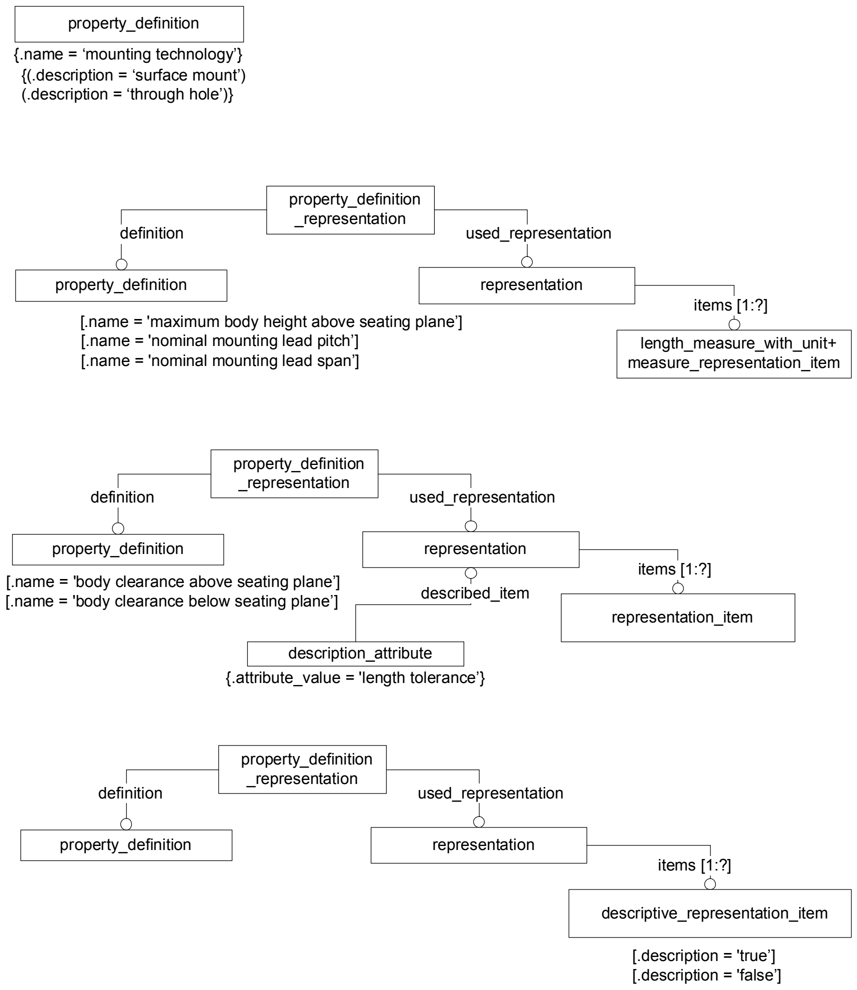
measureWithUnitParameter

Extracts parametric data from certain commonly used measure_with_unit subtypes and representations for integral parameters, area measures, length measures, time measures, and temperature measures.

otherMeasureWithUnitParameter

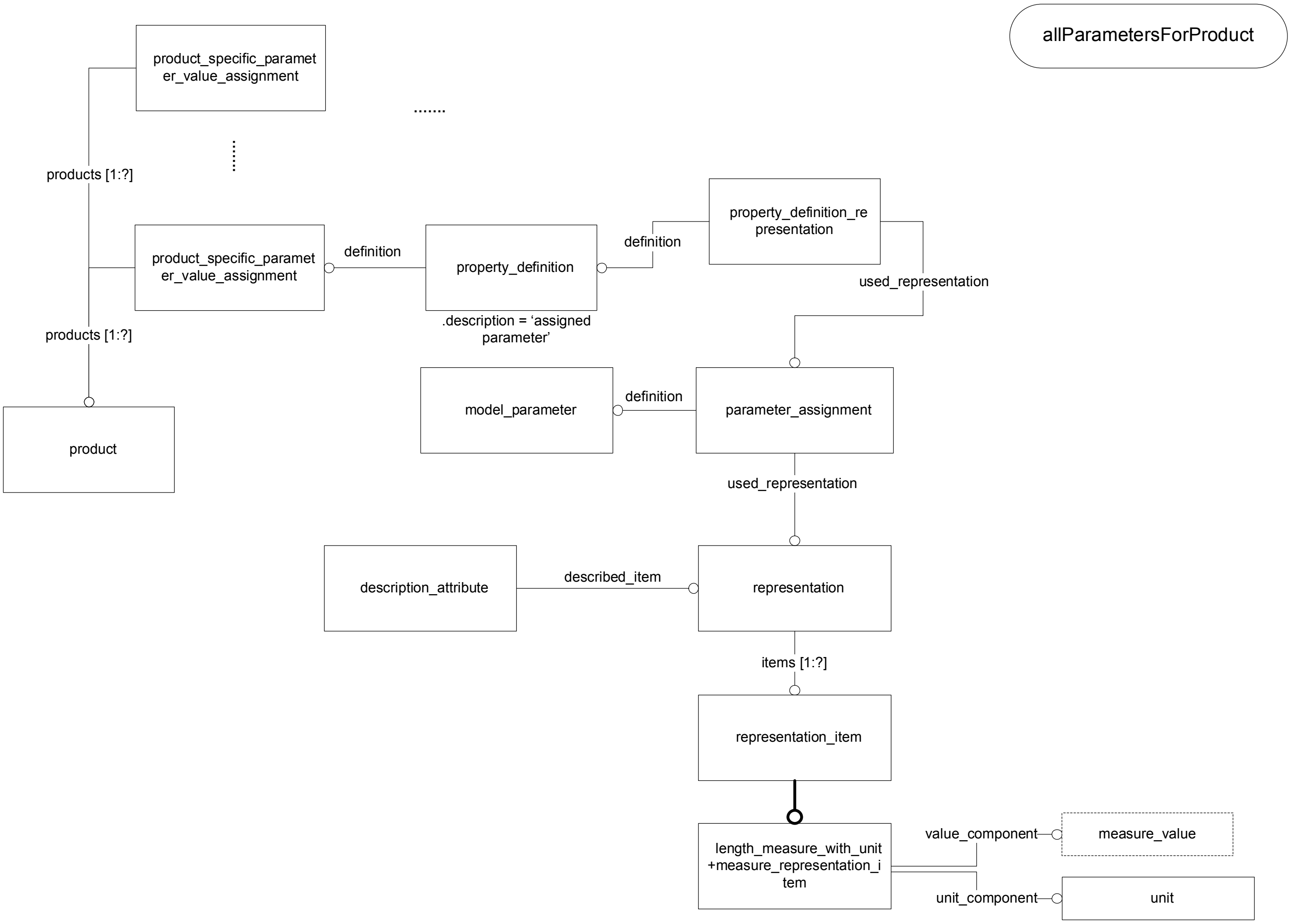
Extracts parametric data from certain time and temperature representations. Present implementation only supports second and degree celcius measures.

parametricAttributeForPropertyDefinition


property_definition
{.name = ‘mounting technology’}
{(.description = ‘surface mount’)
(.description = ‘through hole’)}
property_definition _representation
definition
used_representation
property_definition
representation
items [1:?]
[.name = 'maximum body height above seating plane’]
[.name = 'nominal mounting lead pitch’]
[.name = 'nominal mounting lead span’]
length_measure_with_unit+ measure_representation_item
property_definition _representation
definition
used_representation
property_definition
representation
items [1:?]
[.name = 'body clearance above seating plane’]
[.name = 'body clearance below seating plane’]
described_item
representation_item
description_attribute
{.attribute_value = 'length tolerance’}
property_definition _representation
definition
used_representation
property_definition
representation
items [1:?]
descriptive_representation_item
[.description = 'true’]
[.description = 'false’]

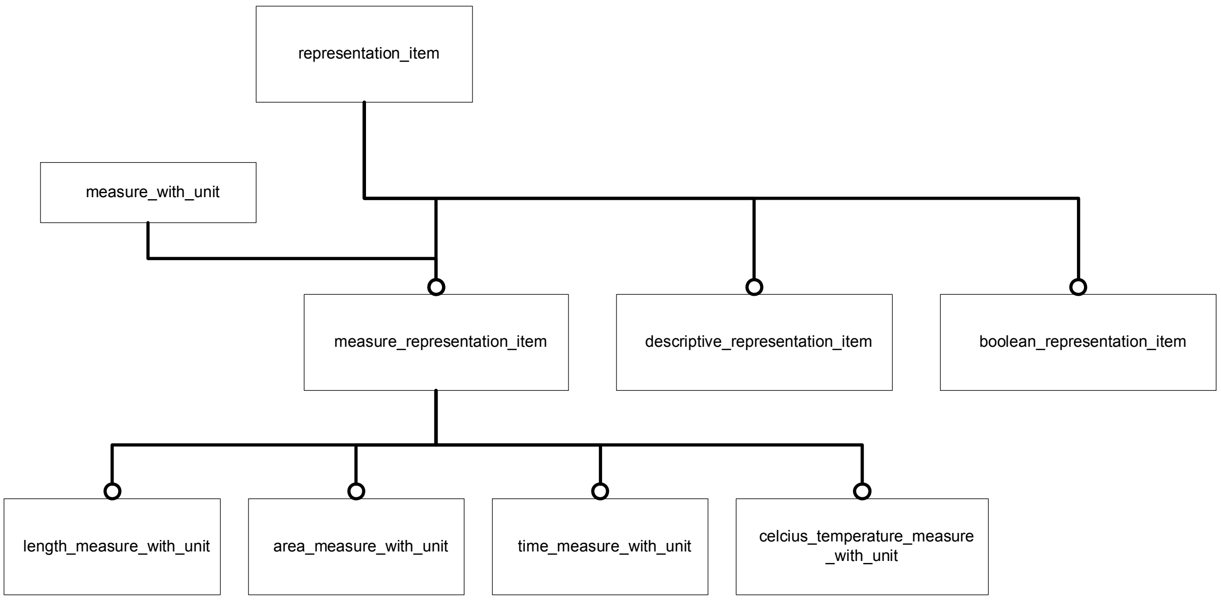
```
// Extracts parametric data from the most commonly used property_definition representations
// for boolean, textual, count, and measure based parametric data. Returns the extracted name and value through
// an implementing class of the Param interface.

Param parametricAttributeForPropertyDefinition(property_definition e_pd)
    {
        String propertyName = e_pd.name
        property_definition_representation e_pdr = referencingEntityOp(e_pd)
            where  {e_pd <- e_pdr.definition}

        if (not (e_pdr == null))
        {
            representation e_rep = e_pdr.used_representation
            representation_item e_ri = e_rep.items[1]
            p = parameterForRepresentationItem(propertyName, e_ri);
        }
        else
        {
            propertyDescription = e_pd.description
            p = new StringParam(propertyName, propertyDescription);
        }
        return p;
    }
```

allParametersForProduct



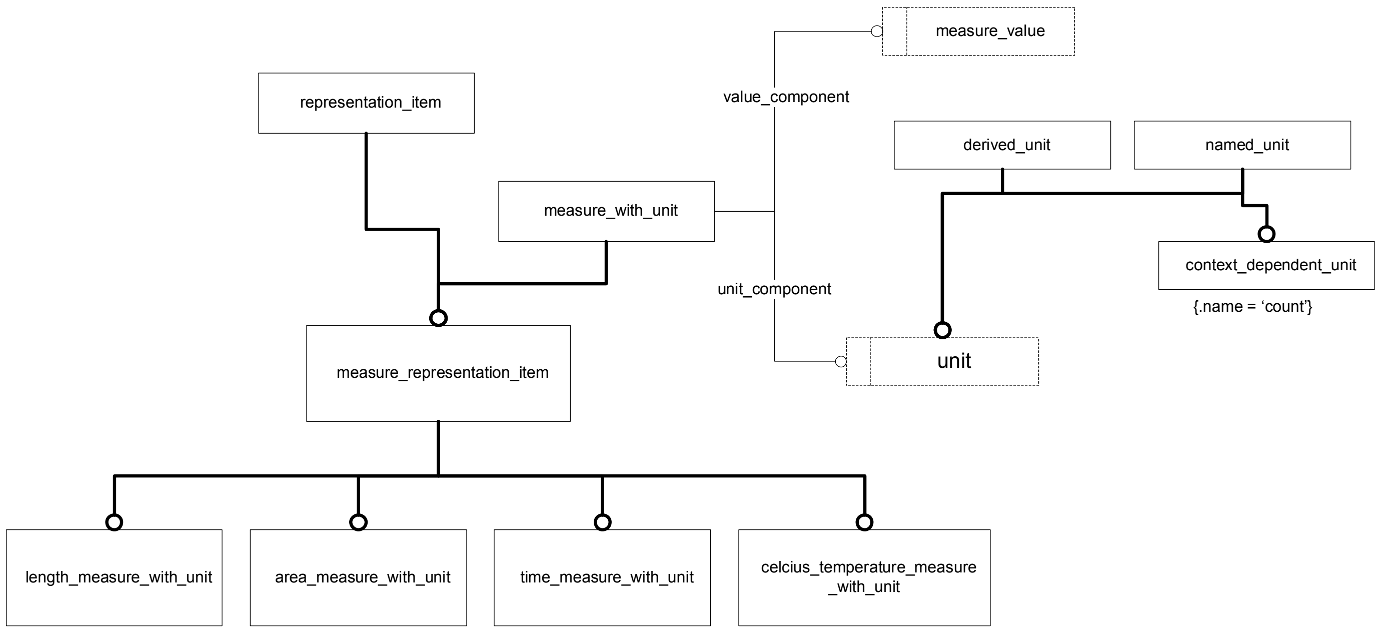
```
// Extracts parametric data from the most commonly used implementations of parameter_assignment related to a
// given product through a product_specific_parameter_value_assignment.
// Implementation supports boolean, textual, count, and measure based parametric data.
// Returns a set of associated product parameters through an implementing class of the Param interface.

Set<Param> allParametersForProduct(product e_p)
  {
    set = new Set<Param>
    Aggregate<parameter_assignment> a_pa = getAllParameterAssignmentsForProduct(e_p)

    For each parameter assignment e_pa in a_pa
    {
      if (e_pa.definition is instance of model_parameter)
      {
        model_parameter e_mp = e_pa.definition
        String parameterName = e_mp.name

        representation e_rep = e_pa.used_representation
        representation_item e_ri = e_rep.items[1]

        if (not (e_ri == null))
        {
          p = parameterForRepresentationItem(parameterName, e_ri);
        }
        else
        {
          p = new StringParam(parameterName, "Unknown");
        }
        add p to set
      }
    }
    return set;
  }
```

parameterForRepresentationItem



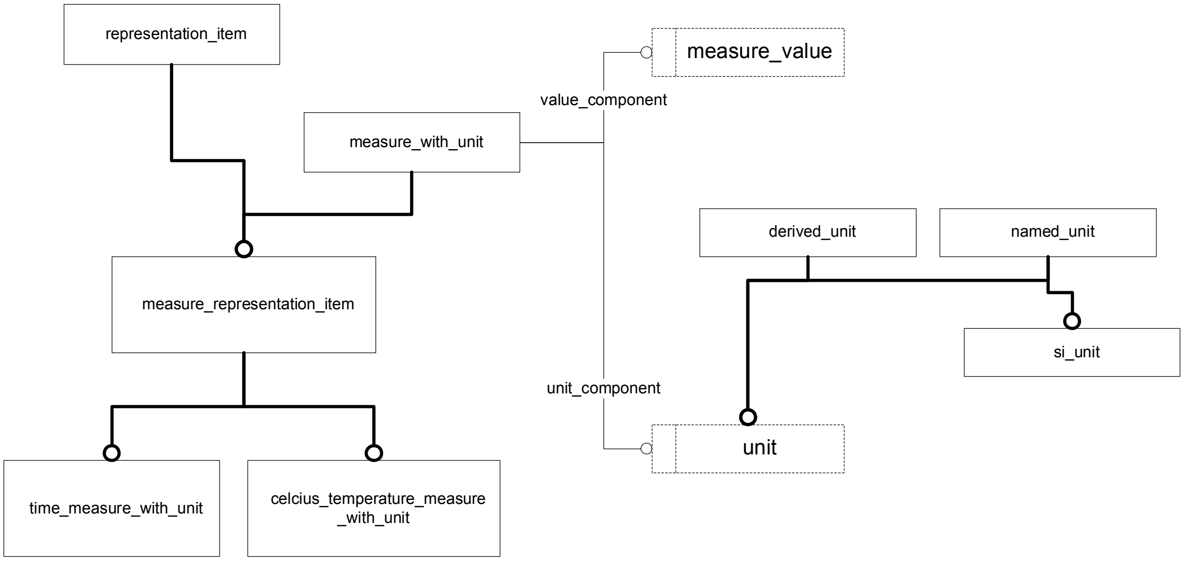
```
// Extracts parametric data from the certain commonly used representation_item representations
// for boolean, textual, count, and measure based parametric data. Returns the extracted name and value through
// an implementing class of the Param interface.

Param parameterForRepresentationItem(String paramName, representation_item e_ri)
    {
        if (e_ri is instance of measure_with_unit)
            return measureWithUnitParameter(paramName, e_ri)
        else if (e_ri is instance of descriptive_representation_item)
            return descriptive_representation_itemParameter(paramName, e_ri
        else if (e_ri is instance of boolean_representation_item)
            return booleanParameter(paramName, e_ri)
        else
            return new StringParam(paramName, "Unknown")
    }
```

measureWithUnitParameter

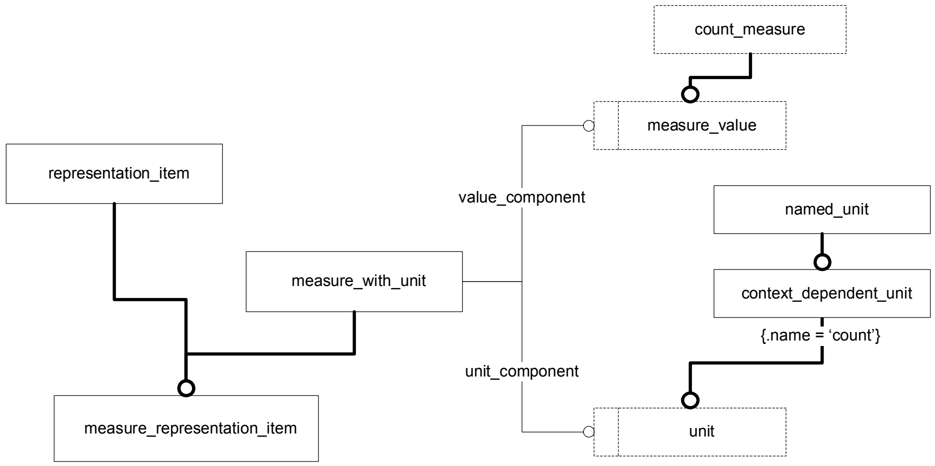


```
// Extracts parametric data from certain commonly used measure_with_unit subtypes and representations
// for integral parameters, area measures, length measures, time measures, and temperature measures.
// Returns the extracted name and value through an implementing class of the Param interface.

Param measureWithUnitParameter(String paramName, measure_with_unit e_mwu)
    {
        if (e_mwu.unit_component is instance of context_dependent_unit)
        {
            context_dependent_unit e_cdu = e_mwu.unit_component
            String unitName = e_cdu.name
            if (unitName == 'count')
                return integerParameter(paramName, e_mwu)
            else
                return otherMeasureWithUnitParameter(paramName, e_mwu)
        }
        else if (e_mwu is instance of length_measure_with_unit)
            return new MeasureParam(paramName, lengthMeasureWithUnitInMM(length_measure_with_unit e_mwu),
                                                                                Units.MILLIMETERS)
        else if (e_mwu is instance of area_measure_with_unit)
            return new MeasureParam(paramName, areaMeasureWithUnitInSqMM(area_measure_with_unit e_mwu,
                                                                                Units.SQUARE_MILLIMETERS)
        else
            return otherMeasureWithUnitParameter(paramName, e_mwu)
    }
```

otherMeasureWithUnitParameter



```
// Extracts parametric data from certain time and temperature representations.
// Present implementation only supports second and degree celcius measures.
// Returns the extracted name and value through an implementing class of the Param interface.

Param otherMeasureWithUnitParameter(String paramName, measure_with_unit e_mwu)
    {
        named_unit e_u = e_mwu.unit_component

        if (not (e_u is instance of si_unit.class))
            throw Exception("Unsupported measure_with_unit encountered")

        if (e_mwu is instance of time_measure_with_unit)
        {
            measure_value = e_mwu.value_component

            if ((e_u.name == SECOND) and e_u.prefix == null))
                return new MeasureParam(paramName, measure_value, Units.SECONDS)
            else
                throw Exception("Unsupported measure_with_unit encountered")
        }
        else if (e_mwu is instance of celsius_temperature_measure_with_unit)
        {
            measure_value = e_mwu.value_component

            if ((e_u.name == DEGREE_CELCIUS) and e_u.prefix == null))
                return new MeasureParam(paramName, measure_value, Units.DEG_CELSIUS)
            else
                throw Exception("Unsupported measure_with_unit encountered")
        }
        else
        {
            throw Exception("Unsupported measure_with_unit encountered")
        }
    }
```


count_measure
measure_value
representation_item
value_component
named_unit
measure_with_unit
context_dependent_unit
{.name = 'count'}
unit_component
measure_representation_item
unit