Modelling of Utility Function and Constraints in Camel Model 2.0

Description

This part describes the general approach to UF modeling in the Camel release 2.0 with specific examples and explanations of supported features.

Utility Function Formula

The formula of the UF can be provided in the application's Requirement Model: i.e. in the Optimisation Requirement section:

```
optimisation requirement maxUtility{
            variable FCRMetricModel.Utility
        }
```

As you can see above, the UF itself is modeled as Variable, which then can contain the actual UF definition:

```
//utility function

variable Utility{
 template UtilityTemplate
 formula:('0.5 * (AppActCardinality * RT_AVG/AppCardinality/100)) + 0.5 * ((AppActCardinality/AppActPrice)/
(AppCardinality*AppPrice))')
 }
```

The definition consists of the following parts:

- **template**

  This is actually not really used within the UF code, but must be provided in the Camel as a mandatory part. Example:

```
template UtilityTemplate{
        attribute utility
    unit FCRUnitModel.Double
    direction 1
    value type FCRTypeModel.DoubleRange
}
```

- **formula**

  The main part of the UF definition. Contains the mathematical formula for the Function. The arguments of the function can be various types (metrics, variables, attributes of Node Candidates). Supported list of arguments can be found in section Supported arguments

It is necessary to use **annotations** from MetaDataSchema.camel concepts to indicate a type of variable/Node Candidate attribute and also to indicate a kind of requirement.

melodic-mms.camel

Example:

```
measurable attribute cardinality [ MetaDataModel.MELODICMetadataSchema.melodic_requirements.
m_cardinality]
```

Constraints

Constraints can be used in a requirement model section of Camel Model as slo.

```
slo MinCoresSlo constraint GenomConstraintModel.WorkerCoresGreaterThanMinimumCores
```

and can be modeled in the a constraint model section of Camel Model.

```
constraint model GenomConstraintModel{
        variable constraint WorkerCoresGreaterThanMinimumCores : GenomMetricModel.
WorkerCoresGreaterThanMinimumCores > 0.0}
```

It is possible to create, among others, a variable constraint and a metric constraint. The variable constraint will be the converted to the constraint in the CP Model and it contains the variable with a formula.

```
variable WorkerCoresGreaterThanMinimumCores{
        template CoresTemplate
        formula: ('WorkerCardinality * WorkerCores - MinimumCores')
}
```

The metric constraint is a constraint that is used in the EMS to trigger reconfiguration. The "main" part of a metric constraint is a composite metric context:

```
composite metric context AvgMemoryUtilisationContext{
        metric MetricTemplateCamelModel.MetricTemplateModel.AverageRAMUtilisation
        grouping per-instance
        window AvgCPUUtilWindow
        schedule AvgCPUUtilSchedule
        object context SmartDesignObjectContext
        composing contexts [RawMemoryUtilisationContext]
}
```

Supported arguments for the UF/Constraints

- **Common variable**

  It represents variable from CP Model and it may be used both in variable constraints and in utility function formula. It must have indicated a type by an annotation.

```
variable AppCardinality{
         template CardinalityTemplate
         component FCRDeployment.Component_App
 }

 template CardinalityTemplate{
      attribute cardinality
      unit FCRUnitModel.Integer
      value type FCRTypeModel.IntRange
 }
  measurable attribute cardinality [ MetaDataModel.MELODICMetadataSchema.melodic_requirements.
m_cardinality]
```

Supported types of variables (VariableType in upperware-metamodel module):

- CORES (CPU concept in MetaDataSchema)
- RAM (RAM)
- STORAGE (Storage)
- CARDINALITY (Cardinality)
- OS (m_os)

- **Raw Metric**

  It represents a metric from CP Model and the Metasolver will update its value.

```
raw metric RT_AVG{
        template AvgResponseTimeTemplate
}

template AvgResponseTimeTemplate{
  attribute responseTime
  unit FCRUnitModel.Integer
  value type FCRTypeModel.IntRange
}
```

- **Composite Metric**

  It connects many metrics into one. It is important to use only metrics in the formula. In CP Model it will be one common metric and its value will by calculated by EMS and updated by the Metasolver.

```
composite metric AverageResponseTime{
        template ResponseTimeTemplate
        formula: ('mean(RawResponseTime)')
}

template ResponseTimeTemplate{
        attribute ResponseTime
        unit UnitTemplateCamelModel.UnitTemplateModel.Seconds
        direction 0
        value type TypeTemplateCamelModel.TypeTemplateModel.ZeroToPositiveInfinityDouble
}
```

- **Common variable with current-config flag**

  It represents values of variables for currently deployed solution. It will be represented as a metric in CP Model and the Metasolver will be responsible for updating these values (from solution in the CP Model). It can be used like common metric.

```
variable AppActCardinality{
        template CardinalityTemplate
        component FCRDeployment.Component_App
        current-config
}

template CardinalityTemplate{
        attribute cardinality
    unit FCRUnitModel.Integer
    value type FCRTypeModel.IntRange
}
measurable attribute cardinality [ MetaDataModel.MELODICMetadataSchema.melodic_requirements.
m_cardinality]:
```

- **Variable with on candidates flag**
  It is supposed to represent a list of indicated attribute for whole Node Candidates offers. For example, a list for all possible values of ram. It is currently not supported, because vectors as variables for function are not supported in MathParser (which is a tool used to create and evaluate a formula of the utility function).
- **Variable with an attribute of Node Candidate**

  It represents an attribute of Node Candidate. Possible type of attributes of Node Candidates is currently only PRICE (m_nc_price).

  This variable will not be a part of the CP Model. It may be used only in a formula of the utility function.

```
variable AppPrice{
        template PriceTemplate
        component FCRDeployment.Component_App
}


template PriceTemplate{
        attribute price
        unit FCRUnitModel.Double
        value type FCRTypeModel.DoubleRange
}

measurable attribute price [ MetaDataModel.MELODICMetadataSchema.melodic_requirements.m_nc_price]:
```

- **Variable with an attribute of Node Candidate with current-config flag**

  It represents values of attributes of Node Candidates for currently deployed solution.

  This variable will not be a part of the CP Model. It may be used only in a formula of the utility function.

```
variable AppActPrice{
        template PriceTemplate
        component FCRDeployment.Component_App
        current-config
}

template PriceTemplate{
        attribute price
        unit FCRUnitModel.Double
        value type FCRTypeModel.DoubleRange
}
measurable attribute price [ MetaDataModel.MELODICMetadataSchema.melodic_requirements.m_nc_price]:
```