AACHENER VERFAHRENSTECHNIK

Technical Report LPT-2008-28

Mathematical Models

J. Morbach, A. Yang, W. Marquardt

July 2008

Enquiries should be addressed to:

RWTH Aachen University
Aachener Verfahrenstechnik
Process Systems Engineering
52056 Aachen

Tel.: +49 (0) 241 80 - 94668
Fax: +49 (0) 241 80 - 92326
E-Mail: secretary.pt@avt.rwth-aachen.de

RWTH

Table of Contents

List of Figures

1. Mathematical Models

The partial model **mathematical_model** is concerned with the description of mathematical models. Fig. 1 gives an overview of the ontology modules of **mathematical_model** and their interrelations. The main module, *mathematical_model*, introduces the basic concepts for mathematical modeling, including model quantities and items pertaining to sub-models and their connections; CapeML (von Wedel, 2002) was taken as an important source. The ontology module *equation_system* further specifies the characteristics of the model equations that constitute a *mathematical model*. Based on these characteristics, an appropriate numerical solver can be selected, which is the concern of the ontology module *numerical_solution_strategy*. The modules *process_model* and *cost_model* describe two particular types of mathematical models: *process models* model the behavior of *process units* and *materials*, while *cost models* predict the cost of *chemical process systems*.

The *process_model* module is extended on the Application-Oriented Layer: The ontology module *laws* establishes models for a number of physical laws that are common in the context of chemical engineering (e.g., the law of energy conservation); *property_models* provides correlations for designated *physical quantities*, such as vapor pressure correlations or activity coefficient models; finally, the module *process_unit_model* establishes customary mathematical models for process units, such as ideal reactor models or tray-by-tray models for distillation columns.

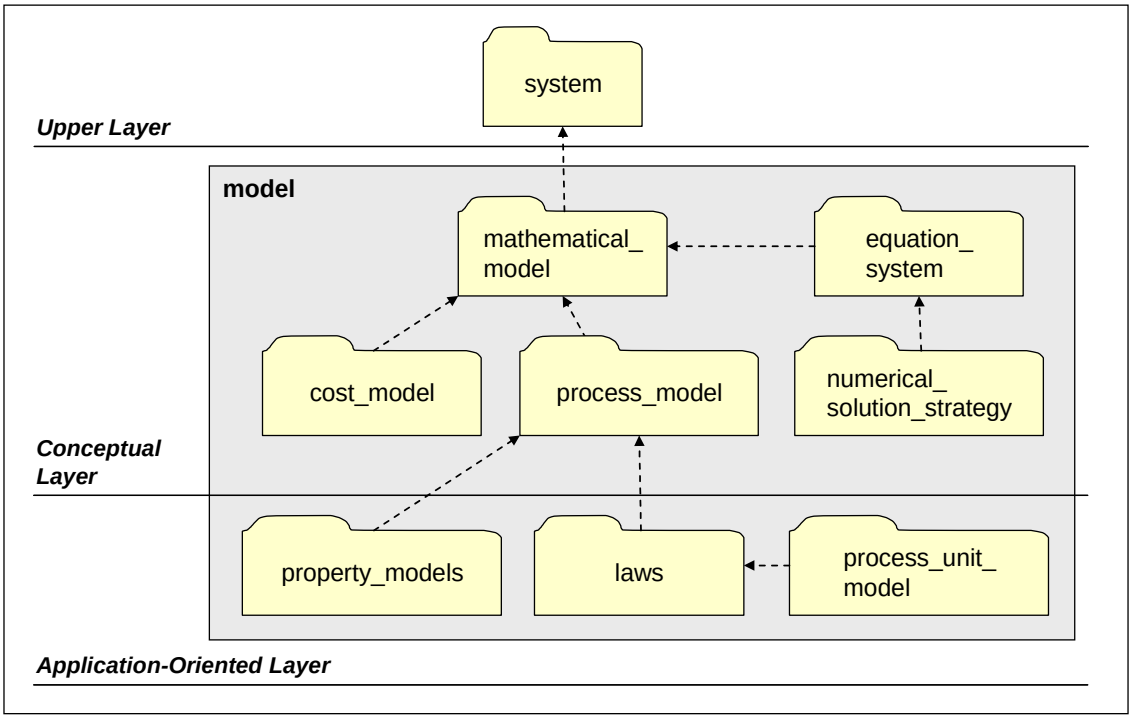

Fig. 1: Overview on partial model **mathematical_model**

1.1. Mathematical Model

A *mathematical model* is a special type of *model*, which uses mathematical language to describe the behaviour of a *system*. A *mathematical model* has a number of properties, the most important of which are *model quantities*. As indicated in Fig. 2, a *model quantity* is a subclass of *physical quantity* that is linked to the model via the relation hasVariable (a specialization of hasProperty, cf. lower right corner of Fig. 2). Like any *physical quantity*, a *model quantity* has a particular *physical dimension* and can be either a *scalar quantity* or a *tensor quantity*. The *value* of a *model quantity* is represented by the class *model quantity specification*; each *model quantity* has exactly one *model quantity specification*.

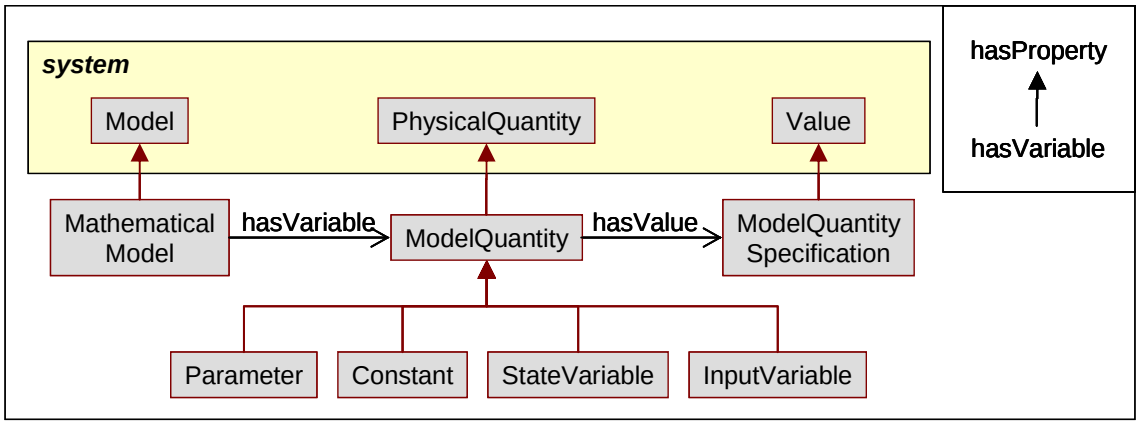

Fig. 2: *Mathematical model*, *model quantity*, and *model quantity specification*

A *model quantity* can be one of the following types: a *constant*, a *parameter*, a *state variable* or an *input variable*, depending on the intended specification of its value: *Constants* and *parameters* constitute the fixed set of specified variables. *Input variables* represent time of spation dependent inputs, which have to be specified for dynamic and/or spatial distributed systems. Finally, *state variables* constitute the fixed set of unknown variables which have to be computed by the model. The *model quantity specification* indicates the numericalValue of the *model quantities*. Unlike *constants*, *parameter* and *input variables* may have different *model quantity specifications* in different simulation runs. If the *model quantity* is of type *parameter* or *state variable*, the *model quantity specification* may indicate their *upper limit* and *lower limit* (cf. Fig. 4).

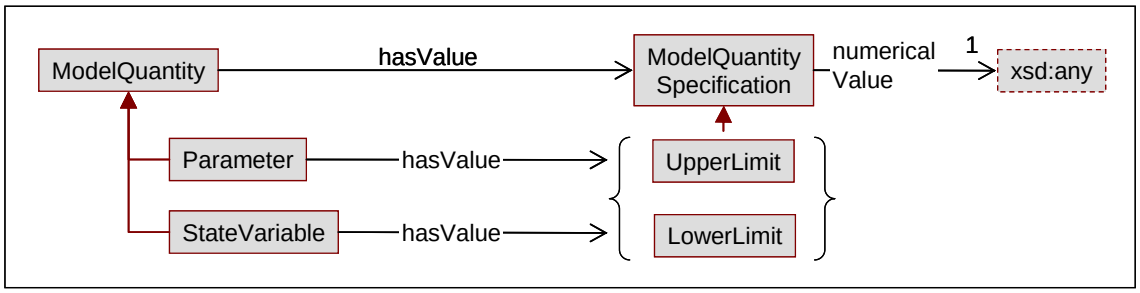

Fig. 3: Assignment of particular *model quantity specifications* to *model quantites*

A *system* which is the target of a models relation is classified as a *modeled object*. The correspondences between the *model quantity* and the *physical quantity* of the *modeled object* can be explicitly represented by means of the relation corresponds-ToQuantity (cf. Fig. 4).

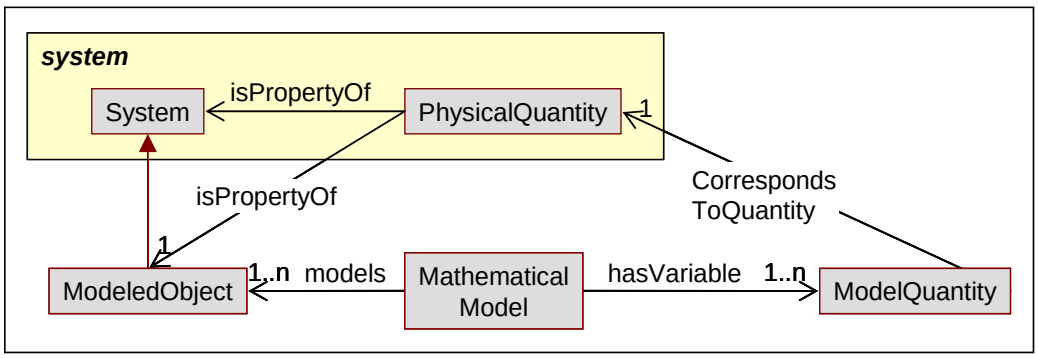

Fig. 4: Correspondence between a *model quantity* and a *physical quantity* of the *modeled object*

Like any *system*, a *mathematical model* can be decomposed into *subsystems*, which are called *submodels*. The *submodel* models the same *system* as its superordinate *mathematical model*; consequently, there is no need to specify the models relation between *submodel* and *system* explicitly. However, such a relation may be indicated if the *submodel* models a designated *subsystem* of the overall *system*.

The different *submodels* of a *mathematical model* are coupled via their *model quantities*, as will be explained in the following:

At first, the concept of a *model port* is introduced. A *model port* is a special type of *property set*, which comprises *model quantities* that can participate in a connection with another model. Thus, a *model port* has the function to identify and to bundle the 'public' variables of a *mathematical model*.

Next, the concept of a *coupling* is established. A *coupling* is a property of the overall *mathematical model*, which defines a connection between two of its *submodels* by linking their respective *model ports*. The *coupling* implicitly defines equality constraints between the *model quantities* in the two *model ports* and must be treated as such (e.g., during degrees of freedom analysis of a complex model). It may be used to connect *mathematical models* both 'horizontally' (i.e., on the same level of decomposition) and 'vertically' (i.e., across levels of decomposition).

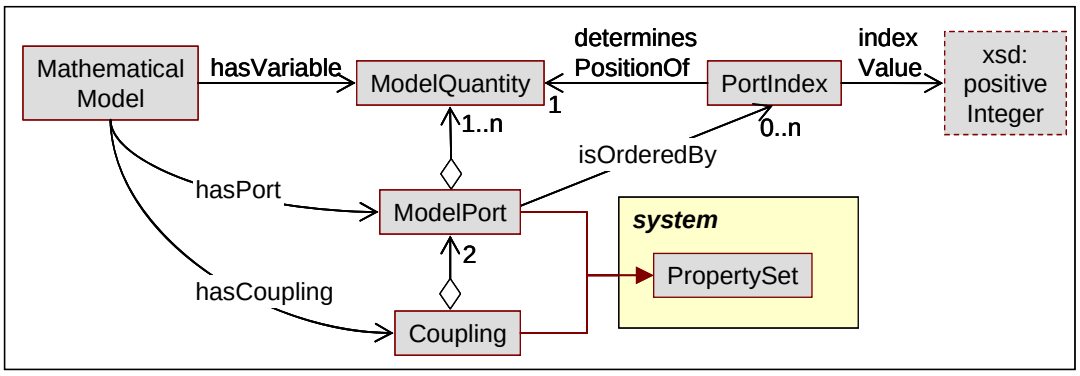

Fig. 5: Variables, ports, couplings

The order of the *model quantities* within a *model port* can be specified by a *port index*, as shown in Fig. 5. The *port index* is used to identify corresponding *model quantities* in a *coupling*: Two *model quantities* of different *model ports* are coupled if and only if their *port indices* have the same indexValues. The specification of a *port index* may be omitted if the correspondence between *model quantities* is evident from the context (e.g., if each of the coupled *model ports* comprises only a single *model quantity*, or if corresponding *model quantities* can be uniquely identified through their *physical dimension*.)

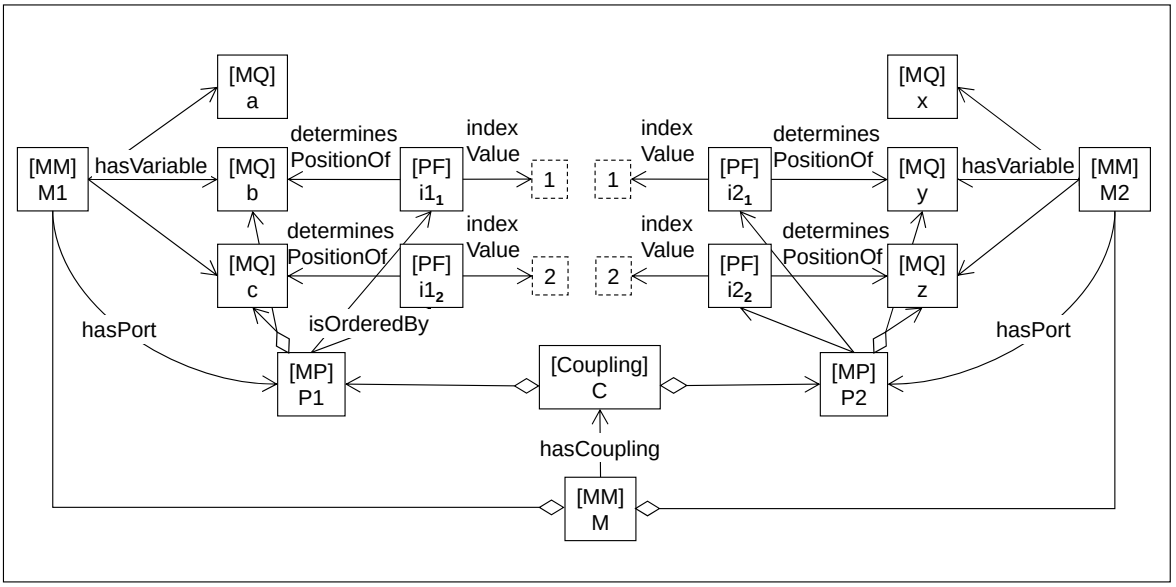

Fig. 6: Exemplary decomposition of model M into submodels M1 and M2

Fig. 6 shows exemplarily the definition of a *mathematical model* M, which consist of two *submodels*, M1 and M2. M1 has the *model quantities* a, b, and c, while M2 has the *model quantities* x, y, and z. Model M1 owns the *model port* P1, which comprises the quantities b and c. Similarly, the *model port* P2 of model

M2 comprises the quantities y and z. P1 and P2 are coupled via the *coupling* C, which is a property of the overall model M. The corresponding quantities of the *coupling* are identified via their *port indices*: b and y have the same indexValue and are thus linked by an equality constraint. The same holds true for quantities c and z.

Usage

The ontology module *mathematical_model* provides only the basic concepts for the description of mathematical models. For practical applications, further concepts may be required, which would typically be supplied by additional ontology modules located on the application-oriented layer. Some possible extensions of the *mathematical_model* module are discussed below.

One possible extension would be the introduction of concepts suitable for representing the model equations. Such an extension could be realized easily by reusing the concepts of the *mathematical_relation* module. However, such an extension is not required in practice, since specialized representation formats for mathematical equations are available, such as MathML (Ion & Miner, 1999), CapeML (von Wedel, 2002), or CellML (Lloyd et al., 2004).

Another possible extension would be the definition of different types (i.e., subclasses) of *model ports*. A particular *model port* type could, for example, prescribe the number of *model quantities* comprised in a *model port*, their types (i.e., *constant*, *parameter*, or *state variable*), their *physical dimensions*, etc. Moreover, a *model port* type could be further characterized through attributes (e.g., assigning a direction to a *model port*, thus turning it into either an *inlet port* or an *outlet port*). That way, standardized model interfaces can be defined – for instance, one may define a standard *energy port*, which contains a single scalar *model quantity* with the *physical dimension* of an energy flow and must furthermore be tagged as an *inlet* or *outlet port*. Such standardization facilitates checking the feasibility of a *coupling*: A *coupling* of two *mathematical models* will be feasible if their *model ports* (a) are of the same type (e.g., *energy port*) and (b) have matching attributes (e.g., an *inlet port* can only be coupled to an *outlet port*).

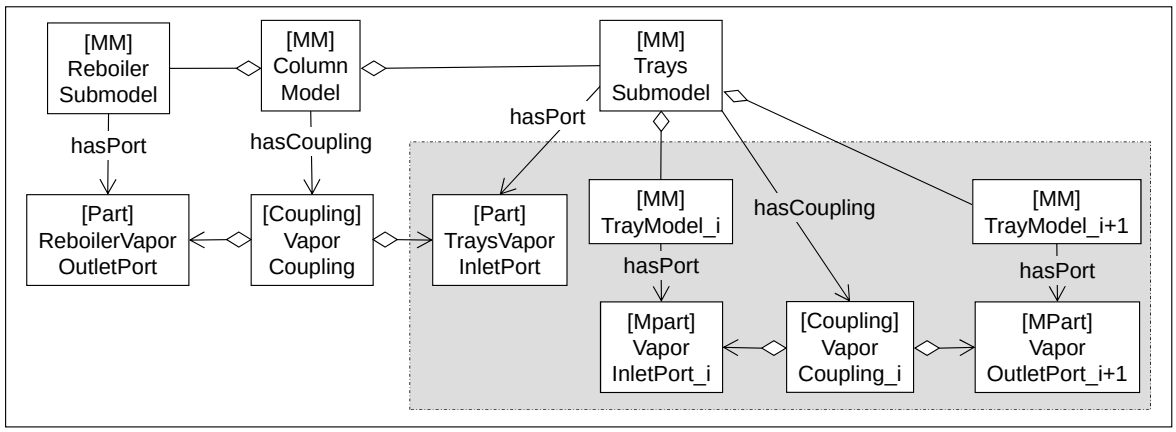

Fig. 7: Specification of the overall model

In practice, a *mathematical model* often consists of several interconnected *submodels* of same the type – for example, the model of a distillation column contains several models of distillation column trays. An application-oriented extension of *mathematical_model* could apply the loop design pattern introduced in the Meta Model (Morbach et al, 2007) to define such repetitive model structures. An example is given in the above and below figures: Fig. 7 specifies the overall structure of a Column Model. It consists of a Reboiler Submodel and a Trays Submodel, which are coupled via a Vapor Coupling (to simplify matters, the liquid phase is not considered in this example). The Trays Model is defined iteratively (see grey-shaded area in Fig. 7 and Fig. 8): It consists of several *submodels* of the same type, which are represented by the individual TrayModel_i. Each TrayModel_i has a VaporInletPort_i, which is coupled to the VaporOutletPort_i+1 of the next TrayModel_i+1. This statement is included in a ForLoop that counts from 1 to 20, that way defining a structure of 20 interconnected tray models. The vapor inlet port of the 20th tray model corresponds to the previously defined TraysVaporInletPort.

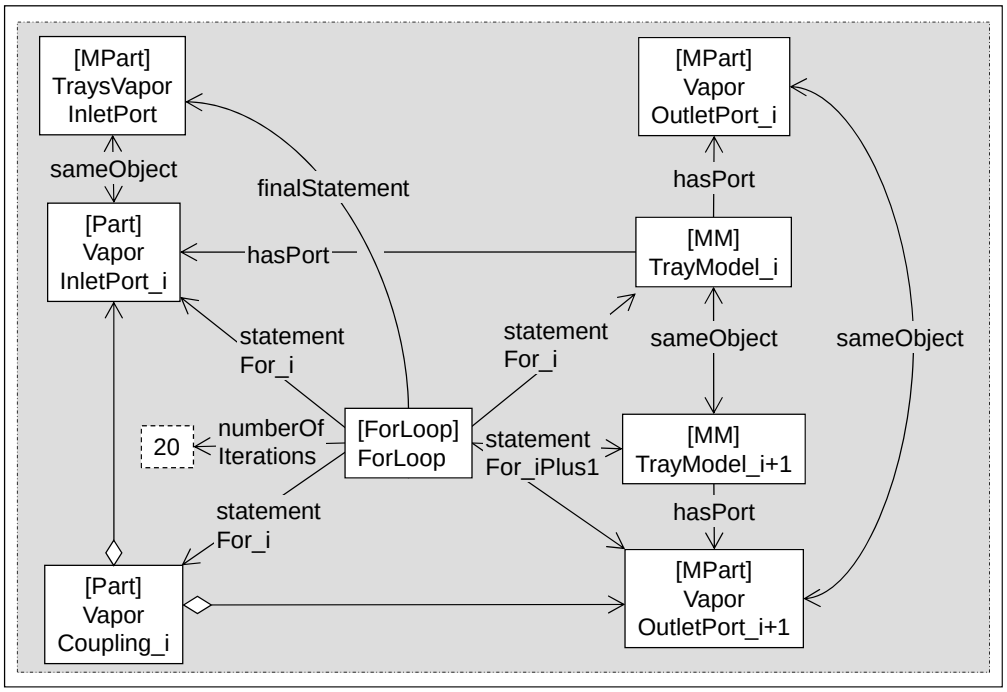

Fig. 8: Specification of the repetitive submodel structure

Concept Definitions

Class Descriptions

Constant

Description

A *constant* is a specified *model quantity*, the *model quantity specification* of which has a constant numericalValue in all simulation runs.

Relations

- *Constant* is a subclass of *model quantity*.
- A *constant* has exactly one *model quantity specification*, which has exactly one numericalValue.

Input variable

Description

Input variables represent time of spation dependent inputs, which have to be specified for dynamic and/or spatial distributed systems.

Relations

- *Input variable* is a subclass of *model quantity*.

Lower limit

Description

An *lower limit* is *model quantity specification* which defines an lower bound for the numericalValue of a *model quantity specification*.

Relations

- *Lower limit* is a subclass of *model quantity*.

Coupling

Description

A *coupling* connects two *model ports* of different *submodels*, thereby defining equality constraints between *model quantities* comprised in the two *model ports*.

Relations

- *Coupling* is a subclass of *property set*.
- A *coupling* comprises exactly two *model ports*.

Mathematical model

Description

A *mathematical model* is a *model* that uses mathematical language to describe the modeled *system* reflecting the phenomena capture in the system's *behavior.*

Relations

- *Mathematical model* is a subclass of *model*.
- A *mathematical model* may have a number of *model quantities*.
- A *mathematical model* may have a number of *model ports*.
- A *mathematical model* may have a number of *couplings*.
- A *mathematical model* may be decomposed into *submodels*.

Modeled object

Description

A *system* that is modeled by means of a *model* is denoted as a *modeled object*.

Definition

A *modeled object* is a *system* that is modeledBy a *model*.

Relations

- *Modeled object* is a subclass of *system*.

Model port

Description

A *model port* is a collection of *model quantities* that can participate in a connection with another *mathematical model*. Thus, a *model port* has the function to identify and to bundle the 'public' variables of a *mathematical model*. Optionally, a *model port* can be ordered by a *port index*.

Relations

- *Model port* is a subclass of *property set*.
- A *model port* comprises some *model quantities*.
- A *model port* comprises only *model quantities*.
- A *model port* can only be ordered by a *port index*.

Model quantity

Description

A *model quantity* represents a *physical quantity* involved in a *mathematical model*, the *value* of which can be either supplied by the modeler or computed from an evaluation of the *mathematical model*.

Definition

A *model quantity* is either a *state variable* or *parameter* or a *constant*.

Relations

- *Model quantity* is a subclass of *physical quantity*.
- The value of a *model quantity* is given by its *model quantity specification*.

Model quantity specification

Description

A *model quantity specification* specifies a *model quantity* in terms of its numerical value (or limits of its numeric value) and its unit of measurement.

Relations

- *Model quantity specification* is a subclass of *value*.
- A *model quantity specification* has exactly one *unit*.
- A *model quantity specification* either specifies the numericalValue of a *model quantity* or defines an upperLimit and a lowerLimit for its numerical value.

Parameter

Description

A *parameter* is a specified *model quantity* (i.e., an input variable), the *model quantity specification* of which may take different numericalValue in different simulation runs.

Relations

- *Parameter* is a subclass of *model quantity*.
- The values of a *parameter* are *model quantity specifications* which have exactly one numericalValue.

Port index

Description

A *port index* orders the *model quantities* comprised in a *model port* by assigning each of them an indexValue. In a *coupling*, *model quantities* with the same indexValue are coupled to each other.

Relations

- *Port index* is derived from the meta class *index*.
- A *port index* is index of exactly one *model port*.
- A *port index* determines the position of exactly one *model quantity*.
- A *port index* has exactly one indexValue.

Submodel

Description

A *mathematical model* can be decomposed into *submodels*.

Definition

A *submodel* is a direct subsystem of a *mathematical model*.

Relations

- *Submodel* is a subclass of *mathematical model*.
- A *submodel* is a direct subsystem of some *mathematical model*.
- A *submodel* can only be a direct subsystem of a *mathematical model*.

State variable

Description

A *state variable* constitute the fixed set of unknown variables which have to be computed by the model. Its *model quantity specification* either indicates the upperLimit and lowerLimit of the *model quantity* (before solving the model) or its numericalValue (after solving the model).

Relations

- *State variable* is a subclass of *model quantity*.

Upper limit

Description

An *upper limit* is *model quantity specification* which defines an upper bound for the numericalValue of a *model quantity specification*.

Relations

- *Upper limit* is a subclass of *model quantity*.

Relation Descriptions

correspondsToQuantity

Description

The relation denotes a one-to-one correspondence between a *model quantity* and a *physical quantity* of the *modeled object*.

Characteristics

- Derived from the meta relation inter-objectRelation
- Domain: *model quantity*
- Range: A *physical quantity* that is a property of some *modeled object*
- Functional

hasCoupling

Description

The relation indicates a *coupling* between two *submodels* of a *mathematical model*.

Characteristics

- Specialization of hasProperty
- Domain: *mathematical model*
- Range: *coupling*

hasModelPort

Description

The relation identifies the *model port* of a *mathematical model*.

Characteristics

- Specialization of hasProperty
- Domain: *mathematical model*
- Range: *model port*

hasModelVariable

Description

The relation indicates the *model quantities* of a *mathematical model*.

Characteristics

- Specialization of hasProperty
- Domain: *mathematical model*
- Range: *model quantity*

determinesPositionOf

Description

The one-to-one relation between a *port index* and the corresponding *model quantity*.

Characteristics

- Derived from the meta relation determinesPositionOf
- Domain: *Port index*
- Range: *model quantity*
- Functional

isIndexOf

Description

The relation isIndexOf points form a *port index* to the associated *model port*.

Characteristics

- Derived from the meta relation isIndexOfArray
- Domain: *Port index*
- Range: *Model port*
- Inverse: isOrderedBy
- Functional

isOrderedBy

Description

The relation isOrderedBy points from a *model port* to its sorting *port index*.

Characteristics

- Derived from the meta relation isOrderedBy
- Domain: *Model port*
- Range: *Port index*
- Inverse: isIndexOf
- Inverse functional

Attribute Descriptions

indexValue

Description

The attribute indexValue indicates the numerical value of a *port index*.

Characteristics

- Specialization of the meta relation index
- Domain: *Port index*
- Datatype: positiveInteger (built-in XML Schema Datatype)
- Functional

1.2. Equation System

The ontology module *equation_system* provides concepts for the description of the model equations that constitute a *mathematical model*. The model equations are not explicitly represented, only their *equation system characteristics* are specified. Moreover, the scope of *equation system characteristics* is confined to those characteristics that are of relevance for selecting an appropriate solver and/or solution strategy for the *mathematical model* (cf. Sec. "numerical solution strategy").

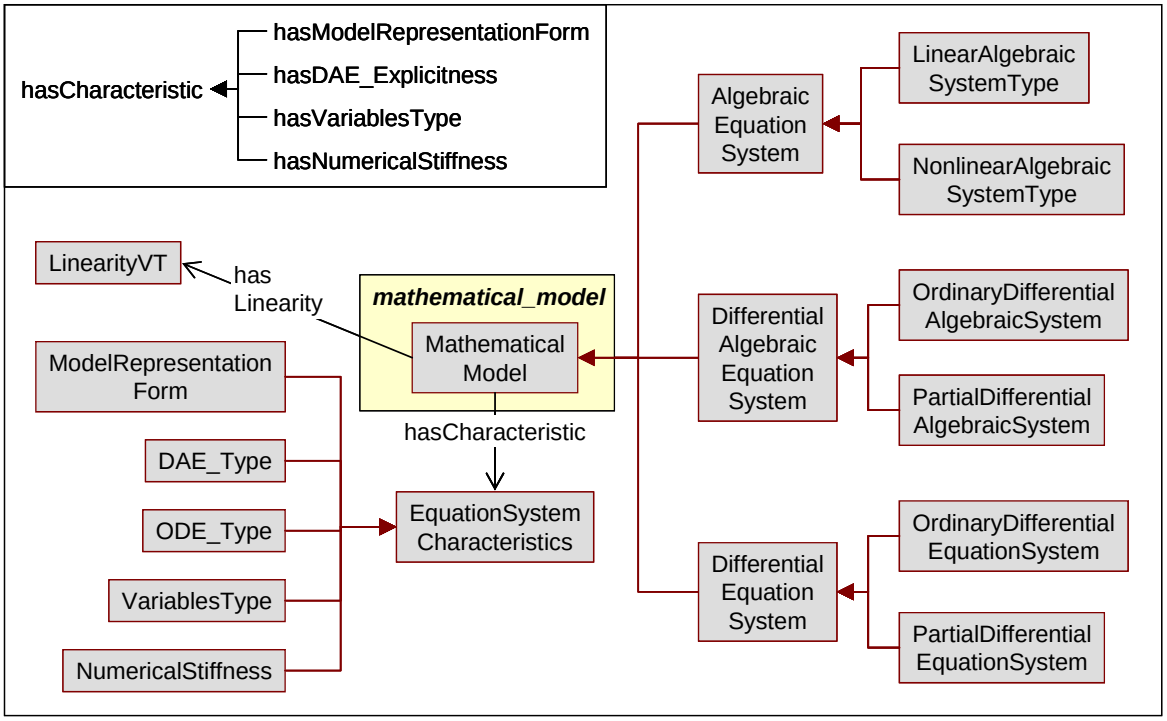

Fig. 9: Equation system characteristics

A *mathematical model* can be classified according to different criteria: equation system type, *variables type*, *model representation form*, etc. Following the recommendations for ontology normalization[1] given by Rector (2003), *equation systems* are classified along a single axis only, using the equation system type as a differentiating criterion (cf. Fig. 9). The other possible criteria are explicitly modeled as *equation system characteristics*, which are to be linked to a *mathematical model* via the relation hasCharacteristic (or one of its specializations, cf. upper left corner of Fig. 9). Note that some of the *equation system characteristics* can only be assigned to special types of *equation systems*; for instance, *DAE explicitness* only applies to *differential algebraic equation systems*.

While the meaning of most concepts displayed in Fig. 9 should be evident from their names, the concept of *model representation form* requires some explanation. A *mathematical model* may appear in two representation forms, which are termed open-form and closed-form:

An open-form model does not provide an explicit solution for its model equations; instead, a numerical solver needs to be applied to the model to obtain a solution. Hence, the open-form model must provide all the information required by the external numerical algorithm to solve the model. For example, a model representing a set of algebraic equations may provide equation residuals and derivatives to a Newton solver. Before an open-form model can be successfully solved, it has to be "squared", meaning that the number of its unknown variables must be the same as that of its equations. Among all the *model quantities* of an open-form model, those declared as *constants* or *parameters* have to be given

1 (1) A class should not have more than one primitive parent; (2) classification of primitive classes should be based on a single criterion (which may remain implicit); (3) other criteria are to be indicated explicitly. More details on this issue can be found in the informal specification of the Meta Model (Morbach et al., 2007).

values (i.e., they need to be assigned a *model quantity specification* with a definite numericalValue) before the model can be evaluated. The other variables are *state variables*. If there are still more *state variables* than equations in a model, it is necessary to assign values to some selected variables (i.e., turn them into *parameters* or *input variables*). Generally, one can freely choose the set of model quantities of an open-form model to be specified, as long as no numerical difficulties arise. The values of the remaining variables can be obtained by solving the model.

A closed-form model includes an underlying numerical algorithm, which solves its model equations. Thus, it does not require any external solver for obtaining the values of its unknown variables. The 'execution' of the closed-form model yields the values of the unknown variables (i.e., its outputs) based on the given values of the specified variables (i.e., its inputs). In this process, the algorithm of a closed-form model accepts only a fixed set of input variables, and consequently returns a fixed set of output variables; no choice for specifying additional variables is available, as in the case of open-form models. Reflected in *model quantity* types, *constants* and *parameters* constitute the fixed set of specified variables (i.e. input variables), while the *state variables* constitute the fixed set of unknown variables (i.e. output variables).

Usage

The ontology module *equation_system* provides the basic concepts for the identification of mathematical models from a mathematical point of view, e.g. whether it is an ODE or DAE etc.. This identification was primarily of concern in the COGents project (cf. Sect. **Fehler! Verweisquelle konnte nicht gefunden werden.**) where this module was applied to specify the type of mathematical model to search for in various libraries.

Typically mathematical models may be classified either by means of content, e.g. a mathematical model for a polyethene reactor, or simply by mathematical features as it is done here. As an example consider a process engineer who searches for a particular mathematical model which is supposed to be applied for the calculations of a reactor. Depending on the software, e.g. he might have only a solver for ODEs available, a classification with respect to the characteristics, e.g. ODE type, is extremely helpful to identify the suitable mathematical model.

Concept Descriptions

Individual concepts of the module *equation_system* are defined below. For an extensive description of the introduced individuals, we refer to Morbach et al (2008j).

Class Descriptions

Algebraic equation system

Description

An *algebraic equation system* is a *mathematical model* which solely consists of algebraic equations.

Definition

An *algebraic equation system* is either a *linear algebraic system* or a *nonlinear algebraic system*.

Relations

- *Algebraic equation system* is a subclass of *mathematical model*.
- An *algebraic equation system* has a characteristic of type *linearity* (i.e., it is either linear or nonlinear).

DAE explicitness

Description

Characterizes the explicitness of a *differential algebraic equation system*.

Definition

The class *DAE explicitness* is an exhaustive enumeration of the individuals fully_implicit and semi-explicit.

Relations

- *DAE explicitness* is a subclass of *equation system characteristics*.

Differential algebraic equation system

Description

A *differential algebraic equation system* (DAE system) is a *mathematical model* that comprises both algebraic and differential equations.

Definition

A *differential algebraic equation system* is either an *ordinary differential algebraic system* or a *partial differential algebraic system*.

Relations

- *Differential algebraic equation system* is a subclass of *mathematical model*.
- A *differential algebraic equation system* has a characteristic of type *linearity* (i.e., it is either linear or nonlinear).
- A *differential algebraic equation system* has a characteristic of type *explicitness* (i.e., it is either (fully_implicit or semi-explicit).
- An *differential algebraic system* has exactly one differentialIndex.

Differential equation system

Description

A *differential equation system* is a *mathematical model* that solely consists of differential equations.

Definition

A *differential equation system* is either an *ordinary differential equation system* or a *partial differential equation system*.

Relations

- *Differential equation system* is a subclass of *mathematical model*.
- A *differential equation system* is of a certain differentialOrder.

Equation system characteristics

Description

The *equation system characteristics* characterize the model equations of a *mathematical model*.

Relations

- *Equation system characteristics* is a subclass of *fixed value set*.

Linear algebraic system type

Description

A *linear algebraic system* type is an *algebraic system* which contains only linear equations.

Definition

A *linear algebraic system* is an *algebraic system* that is characterized as linear.

Relations

- *Linear algebraic system* is a subclass of *algebraic equation system*.

Linearity VT

Description

Linearity VT characterizes the linearity of a mathematical model.

Definition

Linearity is an exhaustive enumeration of the individuals linear and nonlinear.

Relations

- *Linearity* is a subclass of *equation system characteristics*.

Mathematical model (continued)

Relations

- A *mathematical model* is of a certain *variables type*.
- A *mathematical model* has a particular *model representation form*.

Model representation form

Description

A *mathematical model* may appear in two forms, as indicated by the *model representation form*:

- An open-form model is solved by an external algorithm. One can freely choose the inputs and outputs of the open-form model.
- A closed-form model includes an underlying numerical algorithm that solves the model equations. The algorithm accepts only a fixed set of input variables, and consequently returns only a fixed set of output variables.

Definition

The class *model representation form* is an exhaustive enumeration of the individuals open-form and closed-form.

Relations

- *Model representation form* is a subclass of *equation system characteristics*.

Nonlinear algebraic system type

Definition

A *nonlinear algebraic system* is an *algebraic equation system* that is characterized as nonlinear.

Relations

- *Nonlinear Algebraic system* is a subclass of *algebraic equation system*.

Numerical stiffness

Description

In mathematics, stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit ones (Hairer and Wanner 1996). The class *numerical stiffness* is an exhaustive enumeration of the individuals stiff and nonstiff.

Definition

The class *numerical stiffness* is an exhaustive enumeration of the individuals stiff and nonstiff.

Relations

- *Numerical stiffness* is a subclass of *equation system characteristics*.

ODE_type

Description

Characterizes the explicitness of an *ordinary differential equation system*, which can be given in implicit_formulation or explicit_formulation.

Definition

ODE_Explicitness is an exhaustive enumeration of the individuals implicit_formulation and explicit_formulation.

Relations

- *ODE_Explicitness* is a subclass of *equation system characteristics*.

Ordinary differential algebraic system

Description

An *ordinary differential algebraic system* comprises algebraic equations as well as ordinary differential equations, but no partial differential equations.

Relations

- *Ordinary differential algebraic system* is a subclass of *differential algebraic equation system*.

Ordinary differential equation system

Description

An *ordinary differential equation system* (ODE system) is a *differential equation system* which solely consists of ordinary differential equations.

Relations

- *Ordinary differential equation system* is a subclass of *differential equation system*.
- An *ordinary differential equation system* has a characteristic of type *numerical stiffness* (i.e., it is either stiff or nonstiff).

Partial differential algebraic system

Description

A *partial differential algebraic system* is a *differential algebraic equation system* which comprises both partial differential equations and algebraic equations.

Relations

- *Partial differential algebraic system* is a subclass of *differential algebraic equation system*.

Partial differential equation system

Description

A *partial differential equation system* (PDE system) is a *differential equation system* which consists of partial differential equations.

Relations

- *Partial differential equation system* is a subclass of *differential equation system*.

Variables type

Description

A *variables type* indicates whether the *model quantities* of a *mathematical model* are all continuous, all discrete, or partly continuous and partly discrete:

Definition

The class *variables type* is an exhaustive enumeration of the individuals continuous_model, discrete_model, and mixed_model.

Relations

- *Variables type* is a subclass of *equation system characteristics*.

Relation Descriptions

hasDAE_Explicitness

Description

Indicates an *equation system characteristic* of type *DAE explicitness*.

Characteristics

- Specialization of hasCharacteristic
- Domain: *Differential algebraic equation system*
- Range: *DAE explicitness*

hasModelRepresentationForm

Description

Indicates an *equation system characteristic* of type *model representation form*.

Characteristics

- Specialization of hasCharacteristic
- Domain: *mathematical model*
- Range: *Model representation form*

hasNumericalStiffness

Description

Indicates an *equation system characteristic* of type *numerical stiffness*.

Characteristics

- Specialization of hasCharacteristic
- Domain: *Ordinary differential equation system*
- Range: *Numerical stiffness*

hasODE_Type

Description

Indicates an *equation system characteristic* of type *ODE explicitness*.

Characteristics

- Specialization of hasCharacteristic
- Domain: *Ordinary differential equation system*
- Range: *ODE explicitness*

hasVariablesType

Description

Indicates an *equation system characteristic* of type *variables type*.

Characteristics

- Specialization of hasCharacteristic
- Domain: *Mathematical model*
- Range: *Variables type*

Attribute Descriptions

differentialIndex

Description

The attribute represents the differential index of a *differential algebraic equation system*, as defined by Gear & Petzold (1984).

Characteristics

- Domain: *Differential algebraic equation system*
- Datatype: positiveInteger (built-in XML Schema Datatype)

differentialOrder

Description

The attribute differentialOrder denotes the order of a differential equation, which is defined as the order of the highest derivative of a *model quantity* appearing in the differential equation.

Characteristics

- Domain: *Differential equation system*
- Datatype: positiveInteger (built-in XML Schema Datatype)

Individual Descriptions

closed-form

Description

A closed-form model includes an underlying numerical algorithm that solves the model equations. The algorithm accepts only a fixed set of input variables, and consequently returns only a fixed set of output variables.

Characteristics

- Instance of *model representation form*
- Different from open-form

continous_model

Description

A continuous_model denotes a *mathematical model* in which all the *model quantities* are continuous.

Characteristics

- Instance of *variables type*
- Different from discrete_model and mixed_model

discrete_model

Description

A discrete_model denotes a *mathematical model* in which all the *model quantities* are discrete. An example of a discrete_model is an integer model.

Characteristics

- Instance of *variables type*
- Different from continous_model and mixed_model

explicit_formulation

Description

In an explicit_formulation, the *ordinary differential algebraic system* is explicitly solved for the highest-order derivative, i.e., **Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.**.

Characteristics

- Instance of *ODE explicitness*
- Different from implict_formulation

fully_implicit

Description

A *differential algebraic equation system* is fully_implicit if it has the form **Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.**.

Characteristics

- Instance of *DAE explicitness*
- Different from semi-explicit

implict_formulation

Description

In an implicit_formulation, the *ordinary differential algebraic system* is not solved for the highest-order derivative, i.e., **Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.**.

Characteristics

- Instance of *ODE explicitness*
- Different from explicit_formulation

linear

Description

Characterizes a linear equation system.

Characteristics

- Instance of *linearity*
- Different from nonlinear

mixed_model

Description

A mixed_modeldenotes a *mathematical model* in which some of the model quantities are discrete while the others are continuous. An example of a mixed_model is a mixed integer model.

Characteristics

- Instance of *variables type*
- Different from continous_model and discrete_model

nonlinear

Description

Characterizes a nonlinear equation system.

Characteristics

- Instance of *linearity*
- Different from linear

nonstiff

Description

Characterizes a nonstiff differential equation.

Characteristics

- Instance of *numerical stiffness*
- Different from stiff

open-form

Description

An open-form model is solved by an external algorithm. One can freely choose the inputs and outputs of the open-form model.

Characteristics

- Instance of *model representation form*
- Different from closed-form

semi-explicit

Description

A *differential algebraic equation system* is semi-explicit if it has the form $\dot{x} = f(x, y), 0 = g(x, y)$.

Characteristics

- Instance of *DAE explicitness*
- Different from fully_implicit

stiff

Description

Characterizes a stiff differential equation.

Characteristics

- Instance of *numerical stiffness*
- Different from nonstiff

1.3. Numerical Solution Strategy

In this ontology module, strategies for solving *mathematical models* are defined. At present, it is confined to numerical solution strategies only. A classification of numerical solution techniques is given, and the ability of a strategy to solve a particular type of *mathematical model* is explicitly specified. The major concepts are shown in Fig. 10: A *model solution strategy* solves a *mathematical model*; the subclasses of *model solution strategy* represent different types of algorithms, which are specifically designed to solve a certain type of *mathematical model* with certain *equation system characteristics*. To this end, a *model solution strategy* may apply some other, specialized *model solution strategy*. So far, only numerical solution strategies have been considered in OntoCAPE, but symbolic/analytical solution methods could be added in an analogous manner.

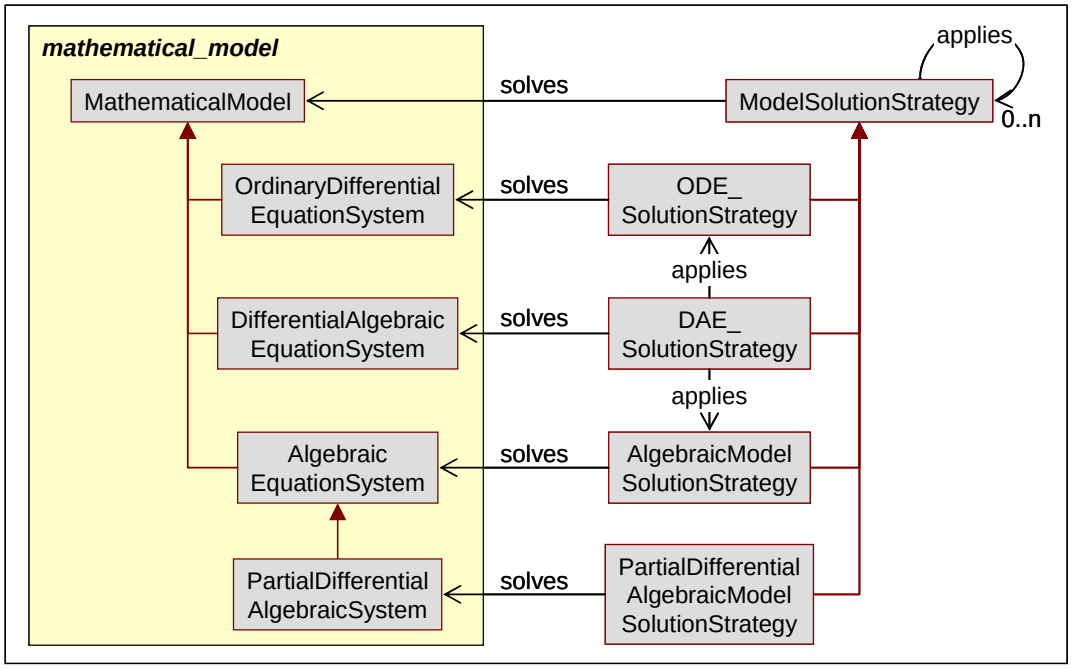

Fig. 10: Numerical solution strategy.

Fig. 11 shows the refinement of class *algebraic model solution* strategy. An exemplary *linear algebraic model solution strategy* is Gauss-elimination, an example of a *nonlinear algebraic model solution strategy* is Newton's method.

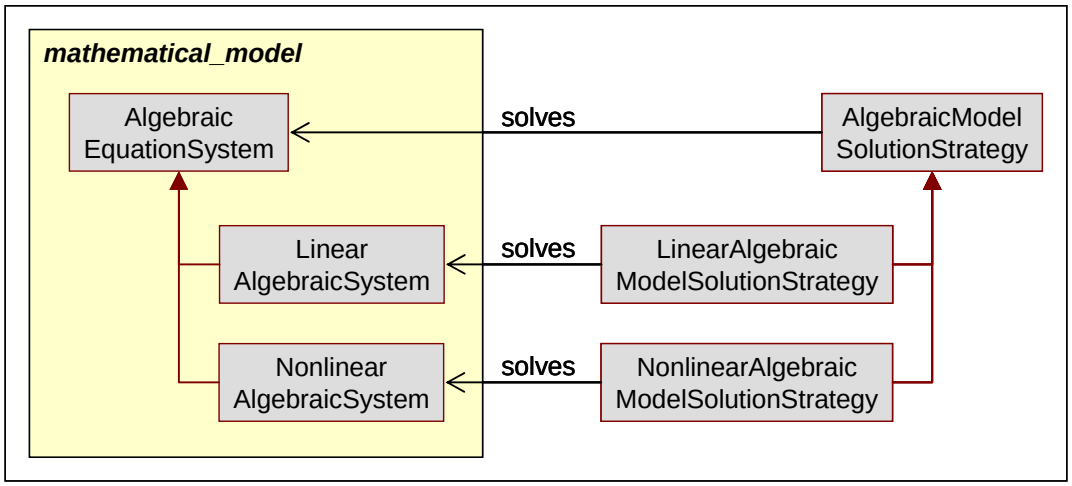

Fig. 11: Types of *algebraic model solution strategies*

An *ODE solution strategy* can be further characterized by indicating if the algorithm is one-step_method (e.g., the classical Runge-Kutta methods) or a multi-step_method (e.g., the Adams-Bashforth methods). Moreover, it can be specified whether the algorithm is a *solution strategy for explicit ODEs* or *implicit ODEs* (cf. Fig. 12).

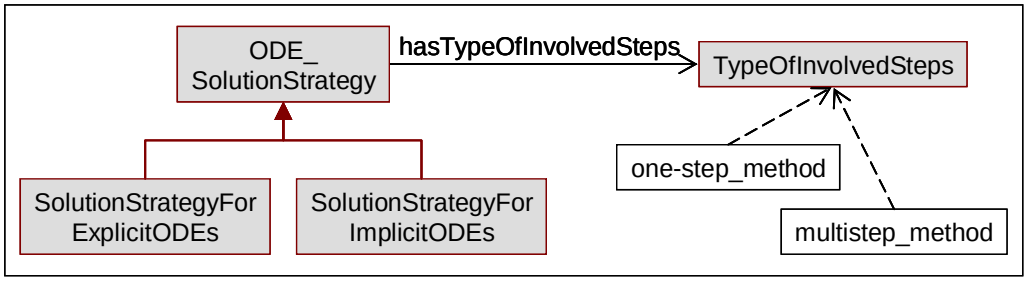

Fig. 12: Further specification of *ODE solution strategy*

Concept Descriptions

Individual concepts of the module *numerical_solution_method* are defined below.

Class Descriptions

Algebraic model solution strategy

Description

An *algebraic model solution strategy* is a *model solution strategy* for solving *algebraic equation systems*.

Relations

- *Algebraic model solution strategy* is a subclass of *model solution strategy*.
- An *algebraic model solution* solves only *algebraic equation systems*.

DAE solution strategy

Description

A *DAE solution strategy* is a *model solution strategy* for solving *differential algebraic equation systems*. Examples are implicit Runge-Kutta, BDF, etc.

Relations

- *DAE solution strategy* is a subclass of *model solution strategy*.
- A *DAE solution strategy* applies an *algebraic model solution strategy* (for initialization, solving corrector equation, etc.).
- A *DAE solution strategy* applies an *ODE solution strategy* (for solving differential equations).
- A *DAE solution strategy* solves only *differential algebraic equation system*.
- A *DAE solution strategy* can only solve *differential algebraic equation systems* up to a certain differentialIndex. This restriction is specified through the attribute handlesDifferentialIndexUpTo.

Linear algebraic model solution strategy

Description

A *linear algebraic model solution strategy* is a *model solution strategy* for solving *linear algebraic systems.* An example is Gauss-elimination.

Relations

- *Linear algebraic model solution strategy* is a subclass of *algebraic model solution strategy*.
- A *linear algebraic model solution strategy* solves only *linear algebraic systems*.

Model solution strategy

Description

A *model solution strategy* is a (typically numerical) algorithm that can be used to solve *mathematical models*.

Relations

- *Model solution strategy* is a subclass of *system*.
- A *model solution strategy* solves some *mathematical models*.
- A *model solution strategy* can only solve some *mathematical models*.
- A *model solution strategy* may apply some other *model solution strategy*.

Nonlinear algebraic model solution strategy

Description

A *nonlinear algebraic model solution strategy* is a *model solution strategy* for solving *nonlinear algebraic systems*. An example is Newton's method.

Relations

- *Nonlinear algebraic model solution strategy* is a subclass of *algebraic model solution strategy*.
- A *non-linear algebraic model solution strategy* solves only *nonlinear algebraic systems*.

ODE solution strategy

Description

An *ODE solution strategy* is a *model solution strategy* for solving *ordinary differential equation systems*.

Relations

- *ODE solution strategy* is a subclass of *model solution strategy*.
- An *ODE solution strategy* has exactly one *type of involved steps*.
- An *ODE solution strategy* solves only *ordinary differential equation systems*.

Partial differential algebraic model solution strategy

Description

A *partial differential algebraic model solution strategy* is a *model solution strategy* for solving *partial differential algebraic systems*.

Relations

- *Partial differential algebraic model solution strategy* is a subclass of *model solution strategy*.
- A *partial differential algebraic model solution strategy* solves only *partial differential algebraic systems*.

Solution strategy for explicit ODEs

Description

A *solution strategy for explicit ODEs* is used to solve *ordinary differential equation systems* that are given in an explicit_formulation. Examples are explicit Euler, explicit Runge-Kutta, etc.

Relations

- *Solution strategy for explicit ODEs* is a subclass of *ODE solution strategy*.
- A *solution strategy for explicit ODEs* solves only *ordinary differential equation systems* that have an *ODE explicitness* of type explicit_formulation.

Solution strategy for implicit ODEs

Description

A *solution strategy for implicit ODEs* is used to solve *ordinary differential equation systems* that are given in an implicit_formulation. Examples are implicit Euler, implicit Runge-Kutta, etc.

Relations

- *Solution strategy for implicit ODEs* is a subclass of *ODE solution strategy*.
- A *solution strategy for implicit ODEs* solves only *ordinary differential equation systems* that have an *ODE explicitness* of type implicit_formulation.

Type of involved steps

Description

A *type of involved step* denotes whether an *ODE solution strategy* is a one-step_method or a multi-step_method.

- A one-step_method characterizes an *ODE solution strategy* that uses information of one integration step. Examples are various Runge-Kutta methods.
- A multi-step_method characterizes an *ODE solution strategy* that uses information of multiple integration steps. Examples are Adams, BDF, etc.

Definition

Exhaustive enumeration of the individuals one-step_method and multi-step method.

Relations

- *Type of involved step* is a subclass of *fixed value set*.

Relation Descriptions

applies

Description

A *model solution strategy* may apply some other, specialized *model solution strategy* (e.g., for initialization, solving corrector equation, solution of a subproblem, etc.).

Characteristics

- Specialization of containsDirectly
- Domain: *model solution strategy*
- Range: *model solution strategy*

hasTypeOfInvolvedSteps

Description

Indicates the *type of involved steps* of an *ODE solution strategy*.

Characteristics

- Specialization of hasCharacteristics
- Domain: *ODE solution strategy*
- Range: *type of involved steps*
- Functional

solves

Description

The relation indicates the type of *mathematical model*, for the solution of which a particular *model solution strategy* is designated.

Characteristics

- Specialization of isDirectlyRelatedTo
- Domain: *model solution strategy*
- Range: *mathematical model*

Attribute Descriptions

handlesDifferentialIndexUpTo

Description

A *DAE solution strategy* can only solve *differential algebraic equation systems* up to a certain differentialIndex. This restriction is specified through the attribute handlesDifferentialIndexUpTo.

Characteristics

- Domain: *DAE solution strategy*
- Datatype: positiveInteger (built-in XML Schema Datatype)

Individual Descriptions

multi-step_method

Description

A multi-step_method characterizes an *ODE solution strategy* that uses information of multiple integration steps. Examples are Adams, BDF, etc.

Characteristics

- Instance of *type of involved step*
- Different from one-step_method

one-step_method

Description

A one-step_method characterizes an *ODE solution strategy* that uses information of one integration step. Examples are various Runge-Kutta methods.

Characteristics

- Instance of *type of involved step*
- Different from mulit-step_method

1.4. Cost Model

The ontology module *cost_model* establishes some *cost models* for predicting the (investment) *costs* of chemical plants. A *cost model* is a special type an *economic performance model*, which models the *economic performance* of a *chemical process system*.

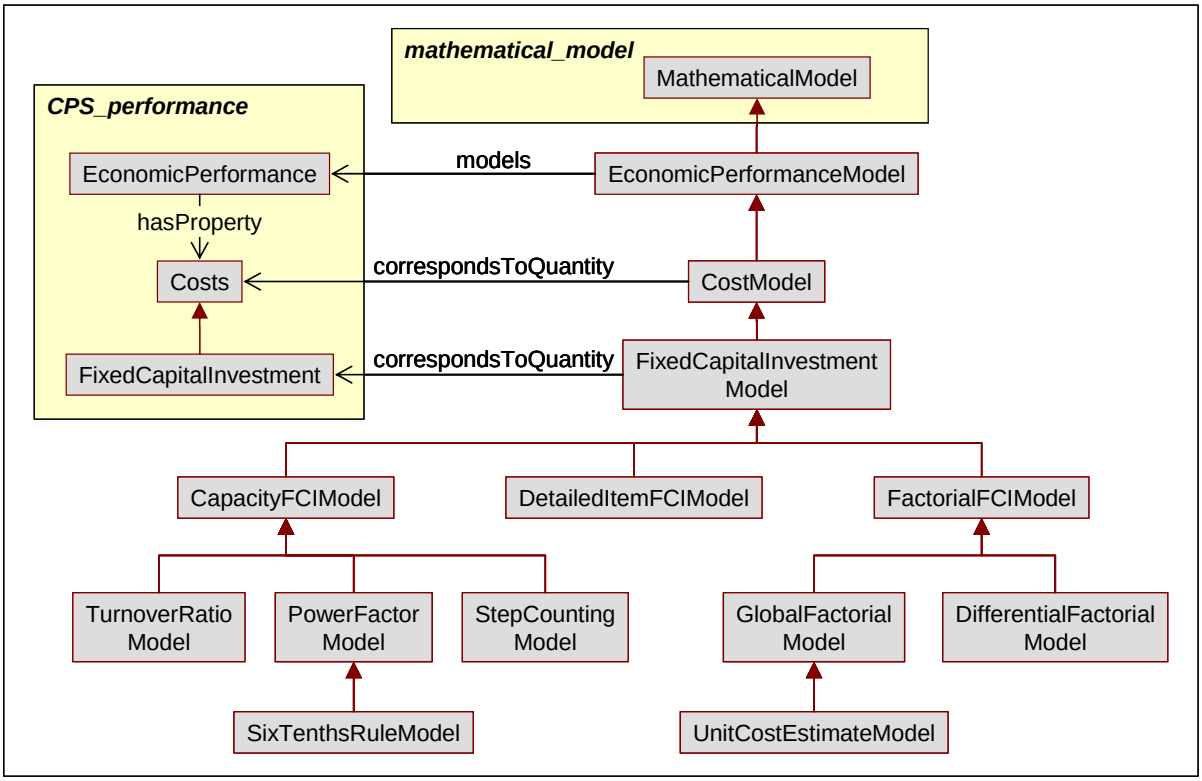

Fig. 13: Models for estimating the fixed capital investment

At present, the module merely holds a number of models for the estimation of the *fixed capital investment*; in the future, further types of *cost models* are to be added, and the existing ones are to be specified in detail. Fig. 13 gives an overview on the *cost models* defined so far. For an explanation of the individual classes, we refer to the concept definitions below.

Concept Descriptions

Individual concepts of the module *cost_model* are defined below.

Class Descriptions

Capacity FCI model

Description

Capacity FCI models are based on *fixed capital investments* of past design projects that are similar to the current *chemical process system*. Besides, some relating factors (e.g., the turn-over ratio), exponential power ratios or more complex relations are given.

Relations

- *Capacity FCI model* is a subclass of *fixed capital investment model*.

Cost model

Description

A *mathematical model* to estimate the investment costs of a *chemical process system*.

Definition

A *cost model* is an *economic performance model* that has a *model quantity* which corresponds to the quantity of *costs*.

Relations

- *Cost model* subclass of *economic performance model*.
- A *cost model* has a *model quantity* that corresponds to the quantity of *costs*.

Detailed-item FCI model

Description

A *detailed-item FCI model* requires careful determination of all individual direct and indirect cost items. For such models, extensive data and large amounts of engineering time are necessary. Therefore, this type of estimate is almost exclusively prepared by contractors bidding on complete and all-inclusive work from finished drawings and specifications.

Relations

- *Detailed-item FCI model* is a subclass of *fixed capital investment model*.

Differential factorial model

Description

Within *differential factorial models*, different factors are used for estimating the costs of the *fixed capital investment*. Examples are modular estimate models, where individual modules consisting of a group of similar items are considered separately, and their costs are then summarized (Guthrie, 1969).

Relations

- *Differential factorial model* is a subclass of *factorial FCI model*.

Economic performance model

Description

An *economic performance model* models the *economic performance* of a *chemical process system*.

Definition

An *economic performance model* is a *mathematical model* that models some *economic performance*.

Relations

- *Economic performance model* is a subclass of *mathematical model*.
- An *economic performance model* models only *economic performance*.

Factorial FCI model

Description

Factorial FCI models rely on the fact that the percentages of the different costs within the *fixed capital investment* are similar for different *chemical process systems*. Based on one or several known costs (for example the *equipment costs*), the *fixed capital investment* is estimated using some factors that are derived from cost records, published data, and experience.

Relations

- *Factorial FCI model* is a subclass of *fixed capital investment model*.

Fixed capital investment model

Description

Fixed capital investment models (FCI models) are *mathematical models* that are used to estimate the *fixed capital investment* of a chemical process system.

Definition

A *fixed capital investment model* is a *cost model* which has a *model quantity* that corresponds to the quantity of *fixed capital investment*.

Relations

- *Fixed capital investment model* is a subclass of *cost model*.
- A *fixed capital investment model* has *model quantity* that corresponds to the quantity of *fixed capital investment*.

Global factorial model

Description

A *global factorial model* estimates the *fixed capital investment* by multiplying the basic equipment cost by some factor. This factor depends, among other things, on the type of chemical process involved, required materials of construction, and the location of the *chemical process system realization*. Examples for global factors are the ones proposed by (Lang, 1947). This model can be extended to calculate the *total capital investment*.

Relations

- *Global factorial model* is a subclass of *factorial FCI model*.

Power factor model

Description

The *power factor model* relates the *fixed capital investment* of a new *chemical process system* to the one of similar, previously constructed systems by an exponential power ratio (cf. Peters & Timmerhaus, 1991).

Relations

- *Power factor model* is a subclass of *capacity FCI model*.

Six-tenths rule model

Description

The *six-tenths rule model* is a *power factor model* with x=0.6.

Relations

- *Six-tenths rule model* is a subclass of *power factor model*.

Step counting model

Description

Step counting models are based on the assumption that the *fixed capital investment* can be estimated from the number of *process steps* (depending on the specific approach, *composite process steps* or *unit operations* and *reactions* are used), multiplied with the costs per *process step* and some correcting factors. The costs of the *process steps* are estimated from their capacity and some other factors (Vogt, 1996).

Relations

- *Step counting model* is a subclass of *capacity FCI model*.

Turnover ratio model

Description

The *turnover ratio model* is a fast evaluation method for order-of-magnitude estimates. The turnover ratio is defined as the ratio of gross annual sales to *fixed capital investment*. Values of turnover ratios for different types of chemical processes are for example given by Schembra (1991) and Vogt (1996).

Relations

- *Turnover ratio model* is a subclass of *capacity FCI model*.

Unit-cost estimate model

Description

Unit-cost estimate models are based on detailed estimates of the main *purchase costs for system realization* (either obtained from quotations or from cost records and published data).

Relations

- *Unit-cost estimate model* is a subclass of *differential factorial model*.

1.5. Process model

As an extension to *mathematical_model*, the ontology module *process_model* enables the definition of specialized *mathematical models* for the domain of chemical engineering. Such models, which model either *process units* or *materials* or subsystems of these, are called *process models* (cf. Fig. 14). The *modeling principle* based on which a *process model* is developed may also be indicated.

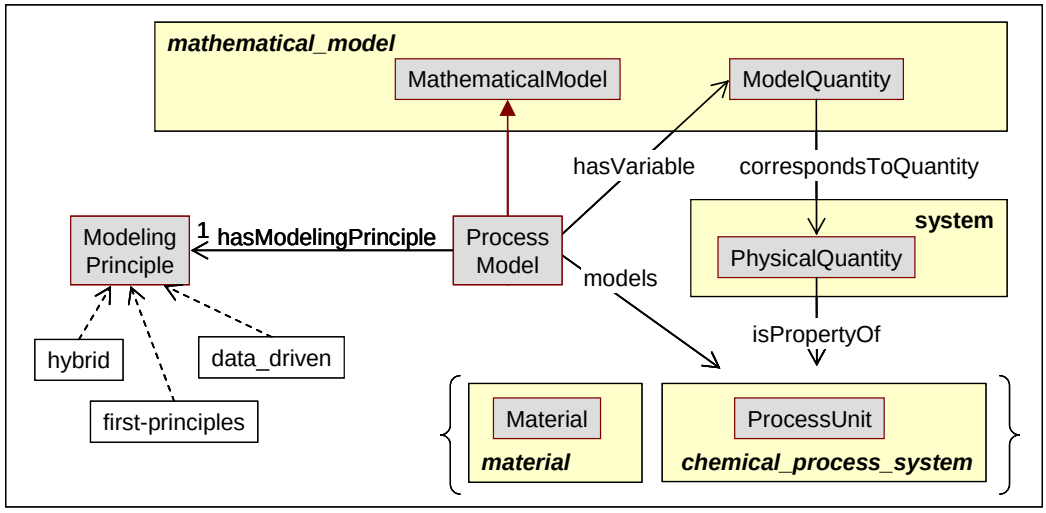

Fig. 14: Overview on *process_model*

A *process model* may contain other *process models*, particularly established *laws* and *property models* (cf. Fig. 15). Neither *laws* nor *property models* are self-contained models, but form part of an overall *process model*, where they represent mathematical correlation between designated *model quantities*.

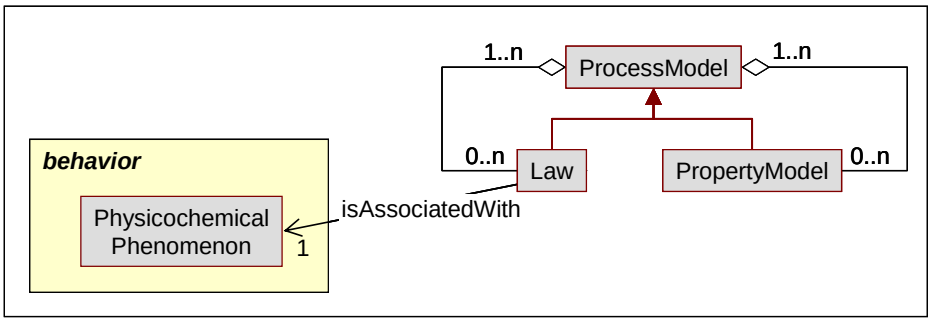

Fig. 15: Laws and property models

A *law* constitutes the mathematical representation of a scientific law, such as the law of energy conservation. Each *law* can be associated with a *physicochemical phenomenon.* The former gives a quantitative, the latter a qualitative description of a certain physical behavior. The correspondence between a *law* and a *physicochemical phenomenon* can be stated via the relation isAssociatedWith, as indicated in Fig. 15. Moreover, the *model quantities* of the *law* correspond to the *physical quantities* that are influenced by the *physicochemical phenomenon*, as exemplarily shown in Fig. 16.

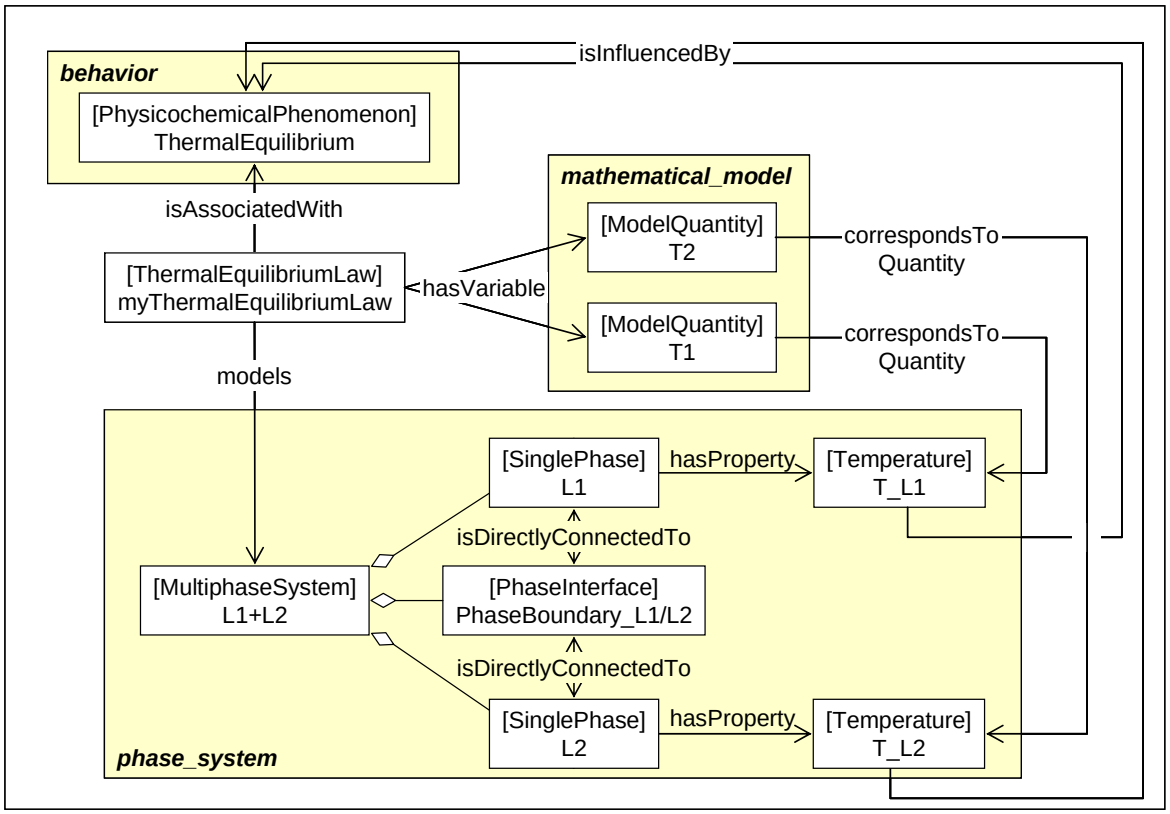

Fig. 16: Exemplary law modeling thermal equilibrium

A *property model* represents a mathematical correlation for the computation of one designated *model quantity*, which corresponds to one specific *physical quantity*. An example is given in Fig. 17: An *activity coefficient model* constitutes a correlation for the computation of *activity coefficients*; consequently, an *activity coefficient model* has one *model quantity* which corresponds to an *activity coefficient*.

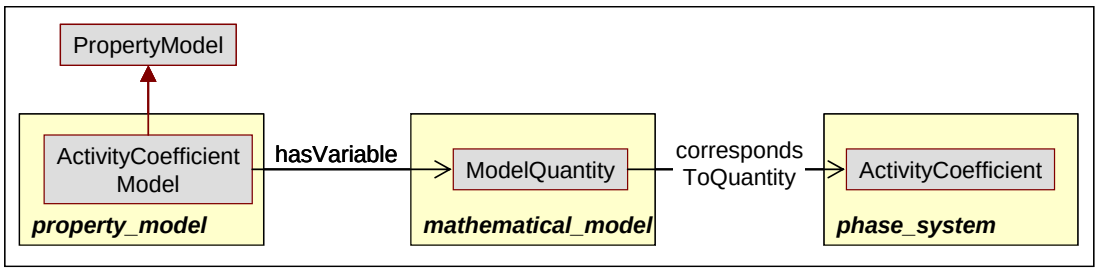

Fig. 17: Exemplary property model

Concept Descriptions

Individual concepts of the module *process_model* are defined below. For an description of the instances of modeling principle, we refer to Morbach et al (2008j).

Class Descriptions

Law

Description

A *law* constitutes the mathematical representation of a scientific law. It usually forms part of an overall *process model*.

Relations

- *Law* is a subclass of *process model*.
- A *law* is associated with exactly one *physicochemical phenomenon*.
- A *law* can only be a direct subsystem of a *process model*.

Modeling principle

Description

A *modeling principle* represents the principle following which the *process model* is developed.

- Following the data_driven *modeling principle*, a *process model* is derived from the *values* of the *properties* of a *modeled object*. Examples of this type of models are neural network models.
- Following the first-principles *modeling principle*, the *process model* is based on established physical laws and mechanisms.
- A hybrid *modeling principle* applies both the first-principles and the data_driven approach.

Definition

Modeling principle is defined by an exhaustive enumeration of the individuals data_driven, first-principles, and hybrid.

Relations

- *Modeling principle* is a subclass of *fixed value set*.

Process Model

Description

A *process model* is a *mathematical model* that models a *process unit* or *material* (or *subsystems* of these).

Relations

- *Process model* is a subclass of *mathematical model*.
- A *process model* may have only *process models* as direct subsystems.
- A *process model* is built following a certain *modeling principle*.
- A *process model* models a *process unit* or a *material*, or a subsystem of these.
- Some *model quantities* of a *process model* correspond to some *physical quantities* of a *process unit* or a *material* or a subsystem of these.

Property model

Description

A *property model* forms part of an overall *process model*. It represents a mathematical correlation for the computation of one designated *model quantity*, which corresponds to one specific *physical quantity*. Examples are vapor pressure correlations or activity coefficient models.

Relations

- A *property model* is a subclass of a *process model*.
- A *property model* is a direct subsystem of a *process model*.
- A *property model* can only be a direct subsystem of a *process model*.

Relation Descriptions

hasModelingPrinciple

Description

Indicates the *modeling principle* on which a *process model* is based.

Characteristics

- Specialization of hasCharacteristic
- Domain: *process model*
- Range: *modeling principle*

isAssociatedWith

Description

The relation denotes a correspondence between a *law* and a *physicochemical phenomenon*. The former gives a quantitative, the latter a qualitative description of a certain physical behavior.

Characteristics

- Specialization of the meta relation inter-objectRelation
- Domain: *law*
- Range: *physicochemical phenomenon*

Individual Descriptions

data_driven

Description

Following the data_driven *modeling principle*, a *process model* is derived from the *values* of the *properties* of a *modeled object*. Examples of this type of models are neural network models.

Characteristics

- Instance of *modeling principle*
- Different from first-principles and hybrid

first-principles

Description

Following the first-principles *modeling principle*, the *process model* is based on established physical laws and mechanisms.

Characteristics

- Instance of *modeling principle*
- Different from data_driven and hybrid

hybrid

Description

A hybrid *modeling principle* applies both the first-principles and the data_driven approach.

Characteristics

- Instance of *modeling principle*
- Different from first-principles and hybrid

1.6. Laws

The ontology module *laws*, located on the Application-Oriented Layer of OntoCAPE, introduces a hierarchical collection of *laws* that are frequently used in process modeling. The law hierarchy shown was originally presented by Marquardt (1995). A selection of the taxonomy related to physicochemical laws is given in Fig. 18: High-level classification of *laws*Fig. 18 - Fig. 22. The high-level concepts include *balance laws*, *constitutive laws*, and *constraints* as shown in Fig. 18.

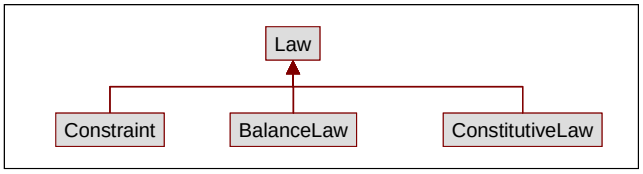

Fig. 18: High-level classification of *laws*

Balance laws generally represent the change of an extensive quantity in *process models*. This typically include balances for total mass and mass of species in a mixture (*mass balance law*), for momentum (*momentum balance law*), for total or any other kind of energy (*energy balance law*), and for the particle number in case of a particulate system (*population balance laws*) as depicted in Fig. 19.

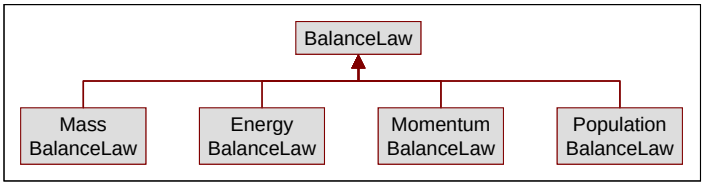

Fig. 19: Specialization of *balance laws*

However, balance equations do not suffice to describe the behavior related to a *process model*. Thus, *constitutive laws* have to be added in order to determine the *process model* completely. Three types of *constitutive laws* may be distinguished (cf. Fig. 20), including *generalized flux laws*, *phenomenological coefficient law*, and *thermodynamic state function law*. *Generalized flux laws* describe the contribution to any kind of *balance law*. These laws typically constitute of phenomenological coefficient and a driving force determined by some thermodynamic state function which are modeled by *phenomenological coefficient laws* and *thermodynamic state function laws*.

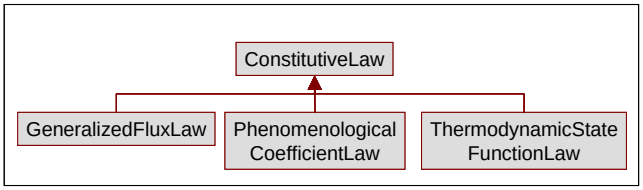

Fig. 20: Specialization of *constitutive law*

In Fig. 21 the specializations of *generalized flux law* are presented including some further specialization associated to transport and exchange phenomena. These specific *laws* have to be considered before a concrete *process model* can be generated.

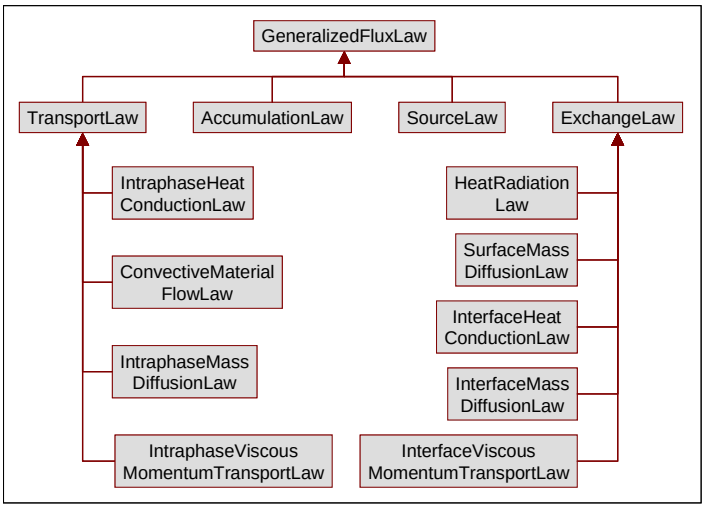

Fig. 21: Specialization of *generalized flux laws*

Finally, *constraints* describe all kinds of (algebraic) relations between *process quantities* which – literally or by assumption – have to hold at any time. Typical examples are *volume constraints* or *equilibrium constraints*, which specialize the class constraint in Fig. 22.

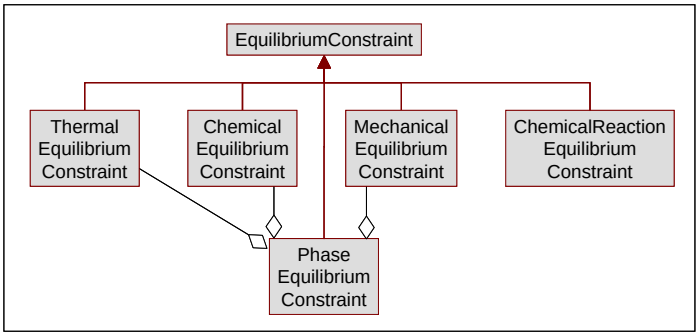

Fig. 22: Specialization of *equilibrium constraints*

In Fig. 22 *equilibrium constraints* are specialized into *thermal equilibrium*, *chemical equilibrium*, and *mechanical equilibrium* on the one hand, which refer to equal temperature, pressure or chemical potential in adjacent phases. *Phase equilibrium* and *chemical reaction equilibrium* are considered on the other hand. It shows that *phase equilibrium* is just an aggregation of thermal, chemical, and mechanical equilibrium. *Chemical reaction equilibrium* refers to a network of chemical reactions residing in a phase where all forward reaction rates equal the backward reaction rates. The constraint *phase equilibrium* can also be formulated in various alternative ways which are fully equivalent[2].

Currently, only the hierarchy of *laws* and the associated *physicochemical phenomenon* are modeled. In future extensions of this ontology module, one may add further definitions and constraints in order to specify a *law*'s *model quantities* and their corresponding *physical quantities*.

Concept Descriptions

Adsorption equilibrium law

Description

A *law* that is associated with the phenomenon of equilibrium_adsorption.

[2] For example, a number of alternative formulations exist for chemical equilibrium, such as the equality of chemical potentials between two phases and the equality of fugacities between two phases. The equality of fugacities can further be written in different forms depending on what property models are to be used in conjunction with the law for the chemical equilibrium.

Relations

- *Adsorption equilibrium law* is a subclass of *equilibrium law*.
- An *adsorption equilibrium law* is associated with the phenomenon of equilibrium_adsorption.

Adsorption kinetics law

Description

A *law* that is associated with the phenomenon of non-equilibrium_adsorption.

Relations

- *Adsorption kinetics law* is a subclass of *non-equilibrium law*.
- An *adsorption kinetics law* is associated with the phenomenon of non-equilibrium_adsorption.

Agglomeration law

Description

A *law* that is associated with the phenomenon of agglomeration (of particles).

Relations

- *Agglomeration law* is a subclass of *particle kinetics law*.
- An *agglomeration law* is associated with the phenomenon of agglomeration.

Breakage law

Description

A *law* that is associated with the phenomenon of breakage (of particles).

Relations

- *Breakage law* is a subclass of *particle kinetics law*.
- A *breakage law* is associated with the phenomenon of breakage.

Chemical kinetics law

Description

A *law* that is associated with some kinetic phenomenon, such as non-equilibrium reaction or adsorption.

Relations

- *Chemical kinetics law* is a subclass of *non equilibrium law*.

Chemical reaction equilibrium law

Description

A *law* that is associated with the phenomenon of chemical_reaction_equilibrium.

Relations

- *Chemical reactions equilibrium law* is a subclass of *equilibrium law*.
- A *chemical reactions equilibrium law* is associated with the phenomenon of chemical_reaction_ equilibrium.

Balance law

Description

A law that is associated with an *accumulation* phenomenon.

Relations

- *Conversation law* is a subclass of *law*.
- A *conversation law* is associated with an *accumulation* phenomenon.
- A *conversation law* can only be associated with an *accumulation* phenomenon.

Constitutive law

Description

Constitutive laws have to be added to balance laws in order to determine the behavior *process model* completely.

Relations

- *Constitutive law* is a subclass of *law.*

Constraint

Description

Constraints describe all kinds of (algebraic) relations between *process quantities* which – literally or by assumption – have to hold at any time.

Relations

- *Constraint* is a subclass of *law.*

Convective material flow law

Description

A *law* that is associated with the phenomenon of convective_material_flow.

Relations

- *Convective material flow law* is a subclass of *inter-phases transport phenomenon law*.
- A *convective material flow law* is associated with the phenomenon of convective_material_flow.

Energy balance law

Description

A *law* that is associated with the phenomenon of energy_accumulation.

Relations

- *Energy conservation law* is a subclass of *conversation law*.
- An *energy conservation law* is associated with the phenomenon of energy_accumulation.

Equilibrium constraint

Description

A *law* that is associated with a *physical equilibrium phenomenon*.

Relations

- *Equilibrium constraint* is a subclass of *law*.
- An *equilibrium constraint* is associated with a *phase equilibrium phenomenon*.
- An *equilibrium constraint* can only be associated with a *phase equilibrium phenomenon*.

Generalized flux law

Description

Generalized flux laws describe the contribution to any kind of *balance law*.

Relations

- *Generalized flux law* is a subclass of *constitutive law*.

Growth law

Description

A *law* that is associated with the phenomenon of growth (of particles).

Relations

- *Growth law* is a subclass of *particle kinetics law*.
- A *growth law* can only be associated with the phenomenon of growth.

Heat radiation law

Description

A *law* that is associated with the phenomenon of heat_radiation.

Relations

- *Heat radiation law* is a subclass of *inter-phases transport phenomenon law*.
- A *heat radiation law* can only be associated with the phenomenon of heat_radiation.

Inter-phases transport phenomenon law

Description

A *law* that is associated with a cross-phase transport phenomenon (i.e., describing the exchange of mass, momentum, or energy between phases and/or across a phase interface).

Relations

- *Inter-phases transport phenomenon law* is a subclass of *transport phenomenon law*.
- An *inter-phases transport phenomenon law* is associated with one of the following phenomena: *interface molecular transport phenomenon* or heat_radiation or convective_material_flow or surface_mass_diffusion.

Interface heat conduction law

Description

A *law* that is associated with the phenomenon of interface_heat_conduction.

Relations

- *Interface heat conduction law* is a subclass of *inter-phases transport phenomenon law*.
- An *interface heat conduction law* is associated with the phenomenon of interface_heat_conduction.

Interface mass diffusion law

Description

A *law* that is associated with the phenomenon of interface_mass_diffusion.

Relations

- *Interface mass diffusion law* is a subclass of *inter-phases transport phenomenon law*.
- An *interface mass diffusion law* is associated with the phenomenon of interface_mass_diffusion.

Interface viscous momentum transport law

Description

A *law* that is associated with the phenomenon of phase_interface_viscous_momentum_transport.

Relations

- *Interface viscous momentum transport law* is a subclass of *inter-phases transport phenomenon law*.
- An *inteface viscous momentum transport law* is associated with the phenomenon of phase_interface_ viscous_momentum_transport.

Intraphase heat conduction law

Description

A *law* that models the phenomenon of heat_conduction within a single phase.

Relations

- *Intraphase heat conduction law* is a subclass of *intraphase transport phenomenon law*.
- An *intraphase heat conduction law* is associated with the phenomenon of heat_conduction.

Intraphase mass diffusion law

Description

A *law* that is associated with the phenomenon of mass_diffusion within a single phase.

Relations

- *Intraphase mass diffusion law* is a subclass of *intra phase transport phenomenon law*.
- An *intraphase mass diffusion law* is associated with the phenomenon of mass_diffusion.

Intraphase viscous momentum transport law

Description

A *law* that is associated with the phenomenon of viscous_momentum_transport within a single phase.

Relations

- *Intraphase viscous momentum transport law* is a subclass of *intraphase transport phenomenon law*.
- An *intraphase viscous momentum transport law* is associated with the phenomenon of viscous_ momentum_transport.

Intraphase transport phenomenon law

Description

A *law* that models some type of molecular transport within a *single phase*.

Relations

- *Intraphase transport phenomenon law* is a subclass of *transport phenomenon law.*
- An *intraphase transport phenomenon law* is associated with a *molecular transport phenomenon*.
- An *intraphase transport phenomenon law* can only be associated with a *molecular transport phenomenon*.

Mechanical equilibrium law

Description

A *law* that is associated with the phenomenon of mechanical_equilibrium.

Relations

- *Mechanical equilibrium law* is a subclass of *equilibrium law*.
- A *mechanical equilibrium law* is associated with the phenomenon of mechanical_equilibrium.

Mass balance law

Description

A *law* that is associated with the phenomenon of mass_accumulation within a *material amount*.

Relations

- *Mass conservation law* is a subclass of *conservation law*.
- A *mass conservation law* is associated with the phenomenon of mass_accumulation.

Momentum balance law

Description

A *law* that is associated with the phenomenon of momentum_accumulation within a *material amount*.

Relations

- *Momentum conservation law* is a subclass of *conservation law*.
- A *momentum conservation law* is associated with the phenomenon of momentum_accumulation.

Nucleation law

Description

A *law* that is associated with the phenomenon of nucleation (of particles).

Relations

- *Nucleation law* is a subclass of *particle kinetics law*.
- A *nucleation law* is associated with the phenomenon of nucleation.
- A *nucleation law* can only be associated with the phenomenon of *nucleation*.

Particle kinetics law

Description

A *law* that is associated with a *particle phenomenon*.

Relations

- *Particle kinetics law* is a subclass of *non-equilibrium law*.
- A *particle kinetics law* is associated with some *particle phenomenon*.
- A *particle kinetics law* can only be associated with a *particle phenomenon*.

Phase equilibrium law

Description

A *law* that is associated with the phenomenon of phase_equilibrium.

Relations

- *Phase equilibrium law* is a subclass of *equilibrium law*.
- A *phase equilibrium law* is associated with the phenomenon of phase_equilibrium.

Phenomenological coefficient law

Description

This laws typically constitute of phenomenological coefficient which are modeled by *phenomenological coefficient laws*.

Relations

- *Phenomenological coefficient law* is a subclass of *constitutive law*.

Population conservation law

Description

A *law* that is associated with the phenomenon of particle_population_accumulation.

Relations

- *Population conservation law* is a subclass of *conservation law*.
- A *population conservation law* is associated with the phenomenon of particle_population_accumulation.

Reaction kinetics law

Description

A *law* that is associated with some of reaction kinetics phenomenon.

Relations

- *Reaction kinetics law* is a subclass of *chemical kinetics law*.
- A *reaction kinetics law* is associated with the phenomenon of non-equilibrium_chemical_reaction or non-equilibrium_surface_reaction.

Surface mass diffusion law

Description

A *law* that is associated with the phenomenon of surface_mass_diffusion.

Relations

- *Surface mass diffusion law* is a subclass of *inter-phases transport phenomenon law*.
- A *surface mass diffusion law* is associated with the phenomenon of surface_mass_diffusion.

Thermodynamic state function law

Description

This *law* typically constitutes of a driving force determined by some thermodynamic state function.

Relations

- *Thermodynamic state function law* is a subclass of *constitutive law*.

Thermal equilibrium law

Description

A *law* that is associated with the phenomenon of thermal_equilibrium.

Relations

- *Thermal equilibrium law* is a subclass of *equilibrium law*.
- A *thermal equilibrium law* is associated with the phenomenon of thermal_equilibrium.

Transport phenomenon law

Description

A *law* that is associated with a phenomenon of inter- or intra-phase transport.

Relations

- *Transport phenomenon law* is a subclass of *non-equilibrium law*.

1.7. Property models

The ontology module *property_models*, which is located on the Application-Oriented Layer of OntoCAPE, provides a hierarchically ordered collection of frequently used *property models*. As indicated in Fig. 23, a *property model* might be one of the following:

- A *chemical kinetics property model*, which specifies how to calculate the rate coefficient of a homogenous or heterogeneous reaction.
- A *phase interface transport property model*, which provides a correlation for computing certain *phase interface transport properties*.
- A *thermodynamic property model*, which indicates the correlation between certain *intensive thermodynamics state variables* and *intra-phase transport properties*

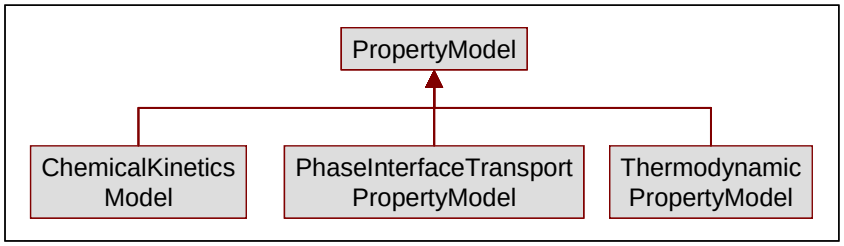

Fig. 23: High-level classification of *property models*

The classification of these specialized *property models* is given in Fig. 24 to Fig. 27:

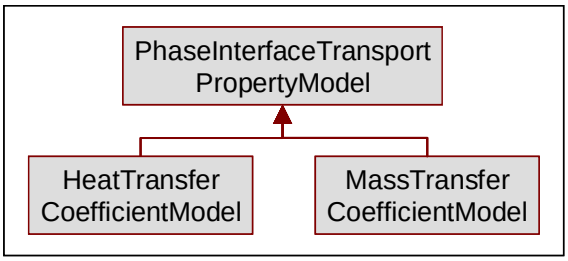

Fig. 24: Some *phase interface transport property models*

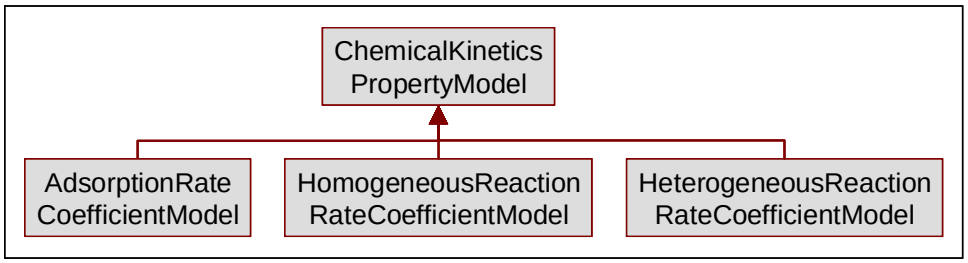

Fig. 25: Some *chemical kinetics models*

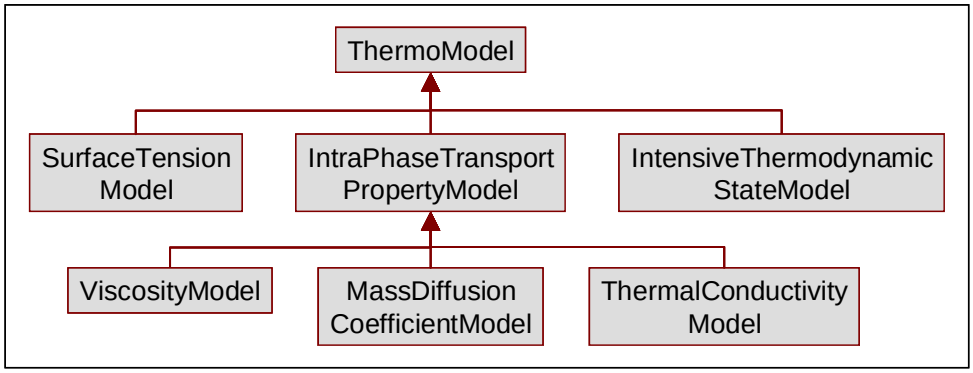

Fig. 26: Some *thermodynamic property models*

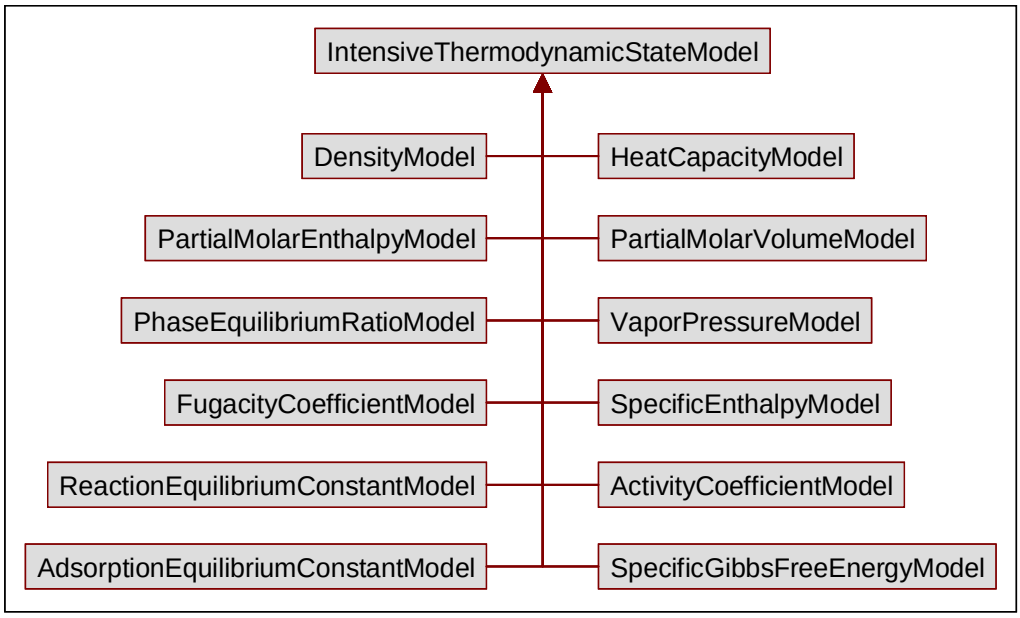

Fig. 27: Some *intensive thermodynamic state models*

Exemplarily, the definition of the class *density model* is shown in Fig. 28: A *density model* has some *model quantities*, one of which corresponds to a *physical quantity* of type *density*. The other *property models* are defined analogously.

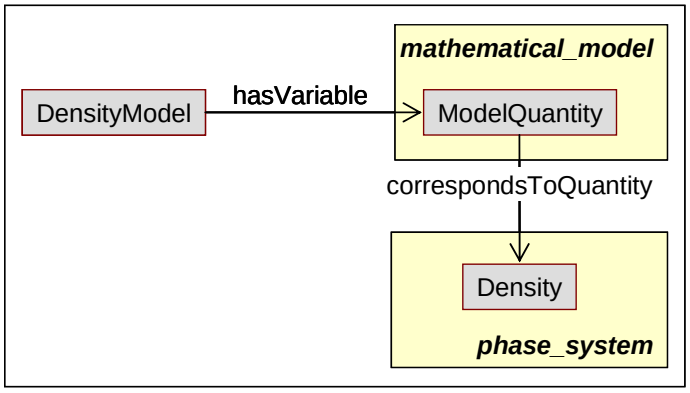

Fig. 28: Definition of the class *density model*

Concept Descriptions

Activity coefficient model

Description

A *property model* that provides a correlation for an *activity coefficient*.

Relations

- *Activity coefficient model* is a subclass of *intensive thermodynamic state model*.
- An *activity coefficient model* has a *model quantity* which corresponds to an *activity coefficient*.

Adsorption equilibrium constant model

Description

A *property model* that provides a correlation for an adsorption equilibrium constant.

Relations

- *Adsorption Equilibrium Constant model* is a subclass of *intensive thermodynamic state model*.

Adsorption rate coefficient model

Description

A *property model* that provides a correlation for an adsorption rate coefficient.

Relations

- *Adsorption rate coefficient model* is a subclass of *chemical kinetics property model*.

Chemical kinetics property model

Description

A *property model* that provides a correlation for a chemical kinetics coefficient.

Relations

- *Chemical kinetics property model* is a subclass of *property model*.

Density model

Description

A *property model* that provides a correlation for the computation of *density*.

Relations

- *Density model* is a subclass of *intensive thermodynamic state model*.
- A *density model* has a *model quantity* which corresponds to a *density.*

Fugacity coefficient model

Description

A *property model* that provides a correlation a *fugacity coefficient*.

Relations

- *Fugacity coefficient model* is a subclass of *intensive thermodynamic state model*.
- A *fugacity coefficient model* has a *model quantity* which corresponds to a *fugacity coefficient.*

Specific Gibbs free energy model

Description

A *property model* that provides a correlation for the computation of the *specific Gibbs free energy*.

Relations

- *Specific Gibbs free energy model* is a subclass of *intensive thermodynamic state model*.
- A *specific Gibbs free energy model* has a *model quantity* which corresponds to a *specific Gibbs free energy*.

Heat capacity model

Description

A *property model* that provides a correlation for the computation of the heat capacity.

Relations

- *Heat capacity model* is a subclass of *intensive thermodynamic state model*.

Heat transfer coefficient model

Description

A *property model* that provides a correlation for a heat transfer coefficient.

Relations

- *Heat transfer coefficient model* is a subclass of *phase interface transport property model*.

Heterogeneous reaction rate coefficient model

Description

A *property model* that provides a correlation for a *surface reaction rate coefficient*.

Relations

- *Heterogeneous reaction rate coefficient model* is a subclass of *chemical kinetics property model*.

Homogeneous reaction rate coefficient model

Description

A *property model* that provides a correlation for a chemical *reaction rate coefficient*.

Relations

- *Homogeneous reaction rate coefficient model* is a subclass of *chemical kinetics property model*.

Intensive thermodynamic State Model

Description

A *property model* that provides a correlation for an *intensive thermodynamic state variable*.

Relations

- *Intensive thermodynamic state model* is a subclass of *thermo model*.
- An *intensive thermodynamic state model* has a *model quantity* which corresponds to an *thermodynamic state variable*.

Intra-phase transport property model

Description

A *property model* that provides a correlation for an *intra-phase transport coefficient*.

Relations

- *Intra phase transport property model* is a subclass of *thermo model*.
- An *intra phase transport property model* has a *model quantity* which corresponds to a *intra-phase transport coefficient*.

Mass diffusion coefficient model

Description

A *property model* that provides a correlation for an intra-phase mass *diffusion coefficient*.

Relations

- *Mass diffusion coefficient model* is a subclass of *intra phase transport property model*.
- A *mass diffusion coefficient model* has a *model quantity* which corresponds to a *diffusion coefficient*.

Mass transfer coefficient model

Description

A *property model* that provides a correlation for a phase interface mass transfer coefficient.

Relations

- *Mass transfer coefficient model* is a subclass of *phase interface transport property model*.

Partial molar enthalpy model

Description

A *property model* that provides a correlation for the computation of the *partial molar enthalpy*.

Relations

- *Partial molar enthalpy model* is a subclass of *intensive thermodynamic state model*.
- A *partial molar enthalpy model* has a *model quantity* which corresponds to a *partial molar enthalpy*.

Partial molar volume model

Description

A *property model* that provides a correlation for the computation of the *partial molar volume*.

Relations

- *Partial molar volume model* is a subclass of *intensive thermodynamic state model*.
- A *partial molar volume model* has a *model quantity* which corresponds to a *partial molar volume*.

Phase equilibrium ratio model

Description

A *property model* that provides a correlation for a *phase equilibrium ratio*.

Relations

- *Phase equilibrium ratio model* is a subclass of *intensive thermodynamic state model*.
- A *phase equilibrium ratio model* has a *model quantity* which corresponds to a *phase equilibrium ratio*.

Phase interface transport property model

Description

A *property model* that provides a correlation for a phase interface transport coefficient.

Relations

- *Phase interface transport property model* is a subclass of *property model*.

Reaction equilibrium constant model

Description

A *property model* that provides a correlation for a reaction equilibrium constant.

Relations

- *Reaction equilibrium constant model* is a subclass of *intensive thermodynamic state model*.

Specific enthalpy model

Description

A *property model* that provides a correlation for the computation of the *specific enthalpy*.

Relations

- *Specific enthalpy model* is a subclass of *intensive thermodynamic state model*.
- A *specific enthalpy model* has a *model quantity* which corresponds to a *specific enthalpy*.

Surface tension model

Description

A *property model* that provides a correlation for the computation of the *surface tension*.

Relations

- *Surface tension model* is a subclass of *intensive thermodynamic state model*.
- A *surface tension model* has a *model quantity* which corresponds to the *surface tension*.

Thermal conductivity model

Description

A *property model* that provides a correlation for the computation of the *thermal conductivity*.

Relations

- *Thermal conductivity model* is a subclass of *intra phase transport property model*.
- A *thermal conductivity model* has a *model quantity* which corresponds to a *thermal conductivity*.

Thermodynamic property model

Description

A *property model* that provides a correlation for *phase system properties* not relevant to rates.

Relations

- *Thermodynamic property model* is a subclass of *property model*.
- A *thermodynamic property model* has a *model quantity* which corresponds to a *phase system property*.

Vapour pressure model

Description

A *property model* that provides a correlation for the computation of the vapor pressure.

Relations

- *Vapour pressure model* is a subclass of *intensive thermodynamic state model*.

Viscosity model

Description

A *property model* that provides a correlation for the computation of the *dynamic viscosity*.

Relations

- *Viscosity model* is a subclass of *intra phase transport property model*.
- A *viscosity model* has a *model quantity* which corresponds to a *dynamic viscosity.*

1.8. Process unit models

The ontology module *process_unit_models*, located on the application-oriented level of OntoCAPE, provides a collection of *mathematical models* that model the behavioral aspect of *process units*.

Please note that this module is introduced not for the purpose of providing a full account on this topic, but rather for suggesting a principle for defining various types of *process unit models* and illustrating the principle by means of only a few examples.

These exemplary *property unit models* are classified according to the modeled *process units* (Morbach et al. 2008): a *chemical reactor model* models a *chemical reactor behavior*, a *flash unit model* models a *flash unit behavior*, etc. (cf. Fig. 29).

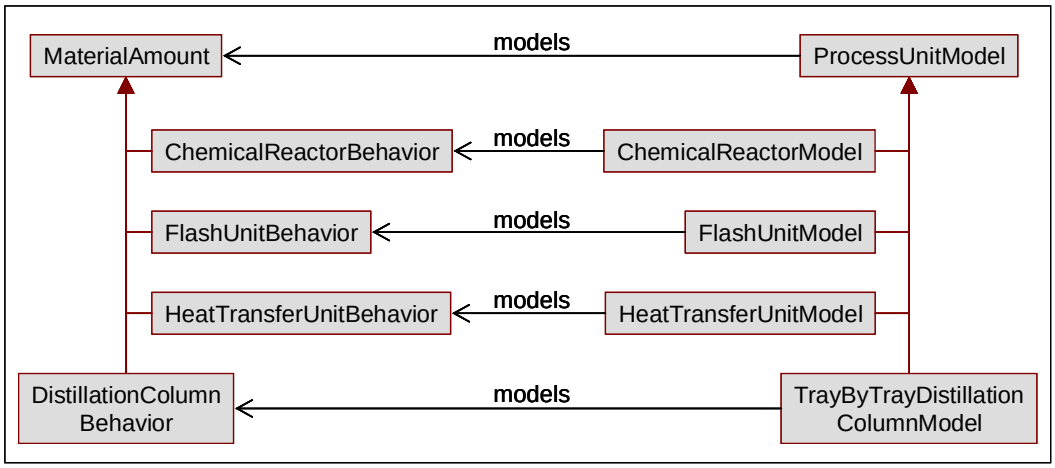

Fig. 29: High-level classification of *process unit models*

Beyond this high-level classification, the ontology module comprises some special types of *process unit models*. Fig. 30 exemplarily shows the definition of a *CSTR model*: A *CSTR model* is a *chemical reactor model* that models a *chemical reactor behavior* with the *physicochemical phenomenon* of phenomenon ideal_mixing. Furthermore, the *CSTR model* is a first-principles model and incorporates the following *laws*: *energy conservation law*, *mass conservation law*, and *reaction kinetics law*.

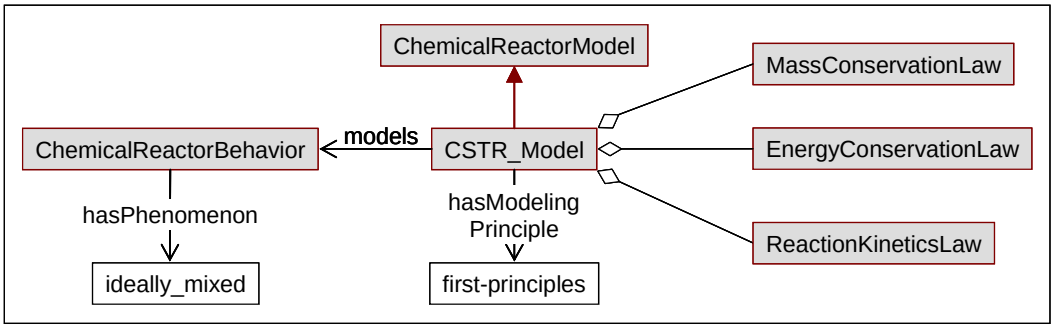

Fig. 30: Definition of the class *CSTR model*

Currently, the ontology module provides only a few of such specialized *process unit models*. In the future, it is to be extended to offer a substantial library of *process unit models*. As for now, the module merely provides the framework for establishing such a library.

References

Gear CW, Petzold L (1984) ODE methods for the solution of differential/algebraic systems. *Trans. Society Computer Simulation* **1**:27–31.

Guthrie K (1969) Data and techniques for preliminary capital cost estimation. *Chem. Eng. (New York)* **24** (3):114-142.

Hairer E, Wanner G (1996) Solving Ordinary Differential Equations II -- Stiff and Differential-Algebraic Problems, Springer, Berlin.

Ion P, Miner R, eds. (1999) *Mathematical Markup Language (MathML) 1.01 Specification*. W3C Recommendation, revision of 7 July 1999. Online available at http://www.w3.org/TR/REC-MathML/. Accessed April 2007.

Jensen AK (1998) *Generation of problem specific simulation models within an integrated computer aided system*. PhD Thesis, Department of Chemical Engineering, Technical University of Denmark.

Lang HJ (1947) Engineering approach to preliminary cost estimates. *Chem. Eng. (New York)*:130-133.

Lloyd CM, Halstead MDB, Nielsen PF (2004) CellML: its future, present and past. Progress in Biophysics and Molecular Biology **85** (2-3):433-450.

Martinson WS, Barton PI (2000) A differentiation index for partial differential-algebraic equations. SIAM *J. Sci. Comput.* **21**:2295–2315.

Morbach J, Yang A, Marquardt W (2008j) *OntoCAPE 2.0 – Mathematical Models*. Technical Report (LPT-2008-28), Lehrstuhl für Prozesstechnik, RWTH Aachen University.

Peters MS, Timmerhaus KD (1991) *Plant Design and Economics for Chemical Engineers*, McGraw-Hill, New York.

Schembra M (1991) Daten und Methoden zur Vorkalkulation des Anlagekapitalbedarfs von Chemieanlagen. PhD thesis, Technische Universität Berlin.

Vogt M (1996) Neuere Methoden der Investitionsrechnung in der Chemischen Industrie. Diploma thesis, Technische Universität Berlin.

von Wedel L (2002) *CapeML – A Model Exchange Language for Chemical Process Modeling*. Technical Report (LPT-2002-16), Lehrstuhl für Prozesstechnik, RWTH Aachen University.

Appendix A Documentation Format

Classes

Classes are characterized by the following categories:

Description: A lexical description of the class, for example “A chemical reactor is an apparatus for holding substances that are undergoing a chemical reaction.” The description explains the meaning of the class to the user.

Definition: Unlike a description, a definition can be transcribed into a formal ontology language, where it establishes the set of necessary and sufficient conditions from which the membership of an ontological concept (class or individual) to the class can be inferred. Classes for which such a definition can not be indicated are called primitive classes.

Relations: The following characteristics are indicated, if existent:

- *Specialization*. A list of parent classes from which the current class is derived via specialization.
- *Disjointness*. A list of classes which are disjoint with the present class. Disjointness between classes means that an instance of the first class cannot simultaneously be an instance of the second class.
- *Restrictions*. Restrictions of binary relations (or attributes) specify the existence of a relation (or attribute) as well as its cardinality and value range with respect to the current class.

Usage: Some recommendations for the use of the class may be given if such advice is required.

Relations

Binary relations are characterized by the following categories:

Description: Similar to that of classes mentioned above.

Characteristics: The following characteristics are listed, if existent:

- *Specialization*. A listing of the relations from which the relation is derived via specialization.
- *Domain*. The domain of the relation.
- *Range*. The value range of the relation.
- *Inverse*. The inverse of a relation.
- Further characteristics, such as if the relation is *transitive*, *symmetric*, or (*inverse*) *functional*.

Usage: As above.

Attributes

Attributes are characterized by the following categories:

Description: As above.

Characteristics: The following characteristics are listed, if existent:

- *Specialization*. A listing of the attributes from which the attribute is derived via specialization.
- *Domain*. The domain of the attribute.
- *Range* or *datatype*. The value range of the attribute, which is usually indicated by referring to a built-in XML Schema Datatype (Biron et al., 2004).
- Further characteristics, such as if the attribute is *functional*.

Usage: As above.

Individuals

Predefined individuals are characterized by the following categories:

Description: As above.

Characteristics: The following characteristics are indicated, if existent:

- *Instance of*. The classes from which the individual is instantiated.
- *Different from*. A list of individual which are explicitly declared to be different from the present individual.
- *Relations*. Instances of binary relations the individual is involved in.
- *Attributes*. Attribute values of the individual.

Usage: As above.

Notation Conventions

Classes and relations of the Meta Model are named according to the CamelCase[3] naming convention: UpperCamelCase notation is used to denote identifiers of classes, while relation identifiers are represented in lowerCamelCase notation. No particular naming convention is followed for identifiers of individuals (i.e., instances of classes).

In this document, class identifiers are highlighted by *italicized sans-serif font*; for better readability, the UpperCamelCase notation is not applied in the text, but the individual words that constitute the class identifiers are written separately and in lowercase (e.g., *class identifier*). If relations are explicitly referred to in the text, they are written in lowerCamelCase notation and are additionally highlighted by sans-serif font. Individuals are accentuated by bold sans-serif font. Partial models are denoted **bold serif font**, *italicized serif font* refers to ontology modules.

In figures, a graphical notation in the style of UML class diagrams is used; the basic elements are depicted in Fig. 31. Grey shaded boxes represent *classes*, white boxes represent *individuals*. *Attributes* are denoted by grey shaded boxes with dashed boundary lines, *attribute values* by white boxes with dashed boundary lines. *Specialization* is depicted through a solid line with a solid arrowhead that points from the subclass to the superclass. A dashed line with an open arrowhead denotes *instantiation*. *Binary relations* are depicted though solid lines. Three basic relation types are distinguished: a line with one open arrowhead represents a *unidirectional* relation; a line with two open arrowheads represents a *symmetric* relation; a line without any arrowheads represents a bidirectional relation[4]. Finally, graphic elements for two special types of relation are introduced: an *aggregation* relation is depicted through a line with a white diamond-shaped arrowhead pointing towards the aggregate class. Similarly, a black diamond-shaped arrowhead indicates a *composition* relation.

[3] CamelCase is the practice of writing compound words joined without spaces; each word is capitalized within the compound. While the UpperCamelCase notation also capitalizes the initial letter of the compound, the lowerCamelCase notation leaves the first letter in lowercase.

[4] In OWL, a bidirectional relation is modeled through a unidirectional relation and its inverse.

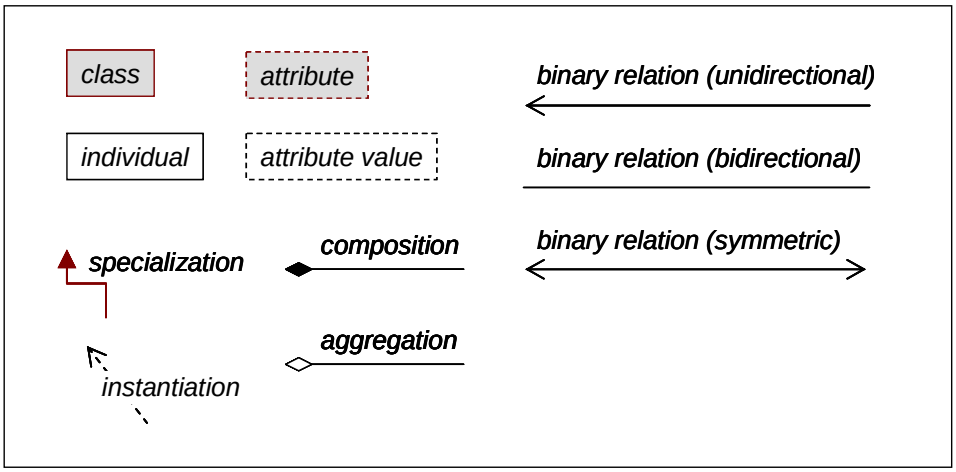

Fig. 31: Basic elements of graphical notation

Index of Concepts