Conditions

A condition is any kind of class expression: union, intersection, named, property restriction, etc. There are three kinds of conditions: Necessary, Sufficient, and Necessary and Sufficient.

Necessary Conditions

A property's multiplicity or type is declared in the context of an owning class or a property holder. When the minimum cardinality is at least one, these declarations are always *necessary* conditions for an instance to be a member of the owning class, or, in the case of a property holder, for an instance to be valid at all.

Necessary and Sufficient Conditions

Another kind of condition is known as both *necessary and sufficient*. A class with at least one necessary and sufficient condition is known as a *defined* class , which means the differentiating characteristics of the class that make it distinguishable from its parent and sibling classes are defined. Note that using a necessary and sufficient condition on a property with a minimum cardinality of zero is not meaningful.

In the modeling tool, a necessary and sufficient condition for a UML class are intersected together, and the UML class is equivalent to that intersection. The condition is available as a stereotype in the concept model and it can applied to a UML Generalization or a property restriction.

AVAILABLE FROM 18.0 SP12 **Aristotelian Definitions**

An Aristotelian definition is achieved when you want to model concept B as a kind of concept A. More specifically, this definition is achieved when the concept B is equivalent to all of the necessary and sufficient conditions of the concept A intersected with all the necessary and sufficient conditions of concept B.

Another explanation is:

- Concept A is equivalent to the intersection of Necessary & Sufficient conditions of A.
- Concept B is equivalent to the intersection of (intersection of Necessary & Sufficient conditions of A) **and** (the intersection of Necessary & Sufficient conditions of B).

> (i) **Importing and Exporting**
>
> The modeling tool supports importing and exporting Aristotelian definitions.

An Aristotelian definition can have only a single sufficient property constraint or generalization attached to it.

In a CCM model, given a class that has one or more sufficient property constraints and/or generalizations, when that class is exported the OWL will specify that the class is owl:equivalent to a owl:Class that is owl:intersectionOf all the restrictions and/or superclasses corresponding to those sufficient property constraints and/or generalizations, respectively. In the case that a CCM class has only a single sufficient property constraint of generalizations, during export CCM creates the equivalent intersection of that single sufficient property constraint or generalization. As it is an intersection containing only a single element that means that the class is equivalent to that single restriction or the superclass. Due to this, OWLAPI removes the intersection axiom to simplify the OWL.

Example

The diagram below shows that when an instance with the property "has contract with" satisfies specific multiplicity ("1..*") and type constraints (of type 'Steering Wheel Manufacturer" or "Windshield Manufacturer") for the property's values, the instance meets a necessary and sufficient condition to be a member of the class "Car Manufacturer". Therefore, an inferencing engine would classify this as an instance of the class "Car Manufacturer". As discussed above, an instance meeting any one of these necessary and sufficient conditions is enough to classify the instance regardless of conditions on the values of any other properties with the {sufficient} constraint owned by the class "Car Manufacturer". The conditions on the values of these properties become necessary conditions on an instance for it to be a valid member of class "Car Manufacturer." Also, an instance meeting any one of these necessary and sufficient conditions is enough to distinguish instances of the class "Car Manufacturer' from its parent class "Manufacturer."
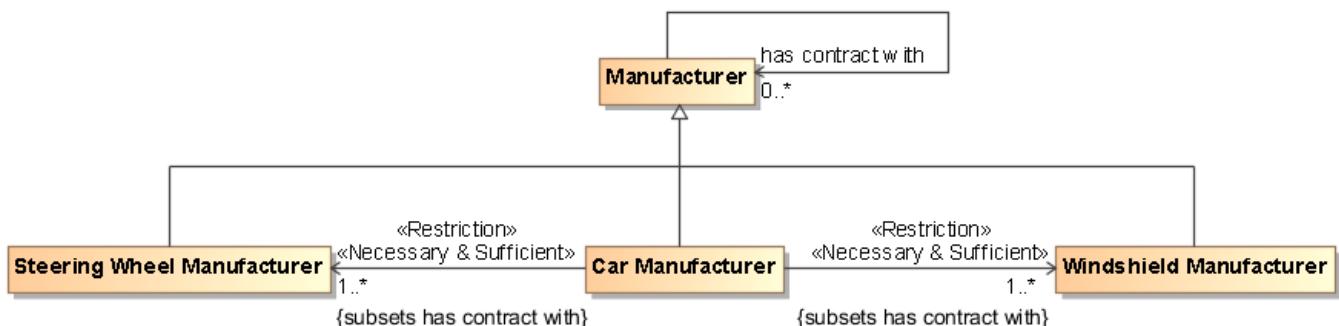
An example of necessary and sufficient condition.

Related Pages

- Concept Modeling Semantics