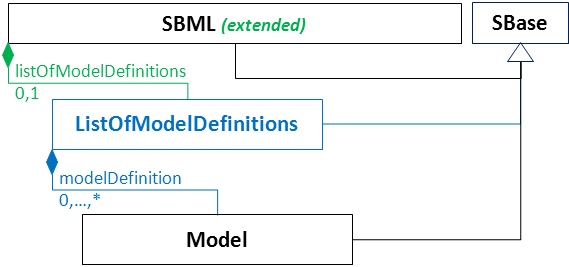
Proposed Syntax and Semantics

## Models and Submodels

[*A note about coloring: In all UML diagrams, classes that exist in SBML Level 3 core are pictured in black. If those elements are extended in this proposal, those extensions are displayed in green. Classes that are new to this proposal are shown in blue.*]

At its heart, Hierarchical Model Composition is about models that you can include in other models. When you define a model for inclusion, we call this a ModelDefinition, and when you actually include it, we call this a Submodel. A definition is like a platonic ideal: it may be a complete model in and of itself, but in this file, it exists only as a concept. A submodel is an instantiation or instance of the previously-defined model: it is the realization of that model inside another model. From the perspective of the model that contains it, it has come into being, and now exists as something that can be modified and adapted. If that containing model is the Model object of the SBML file, it has been fully instantiated. If that containing model is itself a ModelDefinition, it becomes part of that larger model, but has not been fully instantiated in the SBML file until that definition is itself instantiated in the Model.

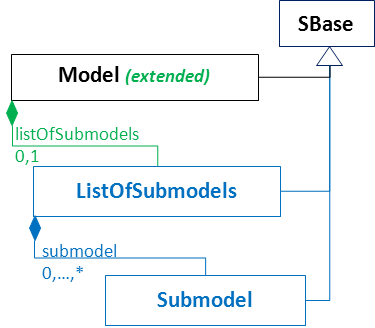
Previous proposals tended to call ModelDefinitions ‘submodels’. We avoid the term here because these models must be valid SBML Model objects in and of themselves, and may indeed have started life that way. (Also, we are using the term ‘submodel’ for the instance of the model inside the containing model.) Another possible term was ‘ModelTemplate’, which was close, but implied that the model was not complete, and needed to be ‘filled in’ by the containing model. While possible, this situation is not required, as for example in the case of model aggregation, when several complete working models are merged to form a larger whole.



**Figure 1**: The definition of the extended SBML class, and the new ListOfModelDefinitions class. We here extend the SBML class by having including a single ListOfModelDefinitions child, which may contain any number of Models. Note that this package extends Model as well, defined below. [Note: probably need the namespace thing here]

Because the model definitions are not ‘owned’ by any other model (as they can be instantiated anywhere), we have here pulled them out of the Model entirely, and instead make a new child of the SBML object itself: a new class, ListOfModelDefinitions, was created to hold them.

The Model class is extended in one respect: to contain a single ListOfSubmodels which, in turn, contain the Submodel objects which will instantiate model definitions inside the containing model.



**Figure 2**: Definition of the extension of the Model class and of the new ListOfSubmodels class. The Model class now may contain a single ListOfSubmodels, which in turn may contain any number of Submodel objects, defined below.

A Model that contains Submodels should be considered to itself contain those elements (such as species, reactions, compartments, rules, and events) contained in the submodel. This is the heart of this proposal: combining models together to make new models.

A Submodel object must say where the model is that it is an instantiation of, and may additionally contain information about how one must modify it to make sense in its new context. The model it instantiates may either be another model in this file, or it may be a model defined in a separate file. The only restriction is that it may not contain loops: it may not refer to its parent model, nor may it refer to a model which in turn instantiates its parent model, etc.

The modifications allowed are of two types: conversion factors, and deletions. If the math of the referenced model must be changed in this context, this is handled through conversion factors. Deletions are for when an element in the referenced model no longer makes sense in its new context: a duplicated degradation rate of a species, for example, or even an annotation of a rate rule that no longer applies.

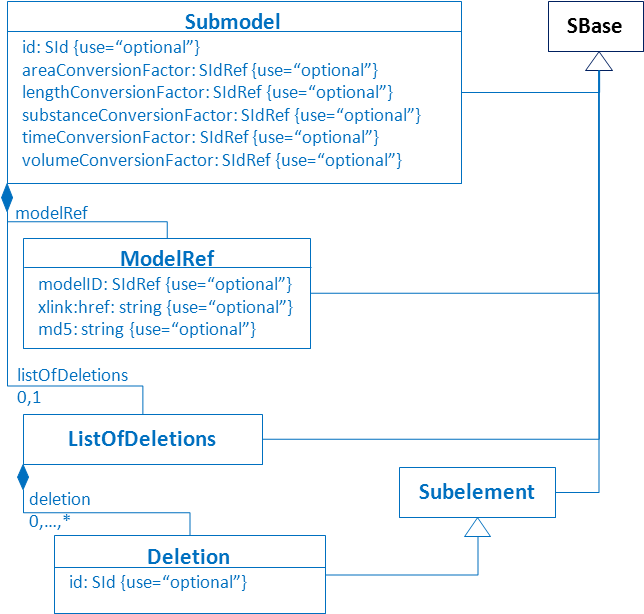


Figure 3: Definition of Submodel, ModelRef, ListOfDeletions, and Deletion. The Submodel object must contain a ModelRef, and may contain any of five different conversion factor attributes plus a ListOfDeletions object. The conversion factors, if present, must reference parameter objects from the parent model. A ModelRef must either contain an SidRef to a model or an xlink:href string attribute, or both, but may not contain neither. If an xlink:href string is used, the md5 string attribute may be additionally used to ensure the referenced file has not changed. The ListOfDeletions may contain any number of Deletions, which are of the class Subelement (defined below) with the addition of an optional ‘id’, so that they can be referenced if needed in the parent model.

The Submodel’s ID attribute is technically optional, but as this is the only way through which the parent model may refer to the submodel’s elements (to link and replace them), most working hierarchical models will have them. (The sole exception is models with pure aggregation, where several models are put together without linking their elements.)

The five possible conversion factors must be references to Parameter objects in the parent model which describe how to convert subelements whose units are inherited from the units of the base submodel. The intricacies of how this works out in practice are complex; for more detail, see the ‘Conversion Factors’ section.

If a ModelRef refers to another model defined in this file, only the ‘model’ attribute is needed. If used, the SidRef must be the ‘id’ of any other Model defined in this file, whether one in the ListOfModelDefinitions, or the ‘main model’ child of the SBML object. The latter is unlikely, as it would mean that the file is defining a modification of the model it presents to the world as the main model associated with this file, but it’s legal and possible—perhaps the main model defines a common scenario, and alternate initial conditions are defined in the ListOfModelDefinitions, for example.

If a ModelRef refers to a model defined in a separate file, the ‘xlink:href’ attribute is used. This is the W3C hyperlink standard, and provides a way to encode both relative and absolute links to other files. The xlink namespace URI is “<http://www.w3.org/1999/xlink>”. The linked-to document must be a SBML Level 3 Version 1 file, but may or may not contain elements of this package. Because the ‘id’ of a Model object of the target file is optional, the model in the referenced file might not be named—if no ‘model’ attribute is used, the main model of the referenced file is used (the ‘model’ object that is the child of the ‘SBML’ object). If the ‘model’ attribute is used along with the ‘xlink:href’ attribute, the model in the referenced file with the given ID is imported, whether this is the main model or a model listed in that file’s ListOfDefinedModules.

The imported model may in turn contain submodels imported from the same or from other external files. This chain should, of course, be followed, with the same caveat that no loops are allowed, whether internally or across files.

The list of deletions is for removing elements from the submodel that no longer have a role in the parent model it is now a part of, and that have no equivalent in the new model. We will discuss deletions and replacements in the next section, along with ports.