XML format for RBA models

S. Fischer, V. Fromion, A. Goelzer

August 17, 2017

Contents

1 Introduction

In this document we present the XML structures used to define a RBA model. A complete RBAÂămodel is composed of the following files:

- metabolism.xml (definition of compartments, metabolic species and metabolic reactions).
- parameters.xml (definition of density constraints and user-defined functions).
- proteins.xml (definition of proteins).
- rnas.xml (definition of RNAs).
- dna.xml (definition of DNA).
- enzymes.xml (definition of enzymatic machineries catalyzing metabolic reactions).
- processes.xml (definition of cell processes necessary to growth and maintenance).

For every file, we present the nodes that composes the XML structure. For every node, we show a class diagram that shows the node's attributes and the children node that it may/must contain. We provide a brief description about the relevance of the node in the RBA model.

2 Conventions

2.1 Naming conventions

2.2 Boolean attributes

3 metabolism.xml

The metabolism file is strongly inspired by SBML. More precisely, it is a subpart of an SBML file.

3.1 RBAMetabolism

The outermost portion of the metabolism file is an instance of class **RBAMetabolism**, shown in Figure 1.

Currently, **RBAMetabolism** has no simple attributes. It includes exactly one instance of **ListOf** container classes. All **ListOf** classes do not have own attributes, they are merely used to organize a list of instances from another class. This organization was inspired by SBML.

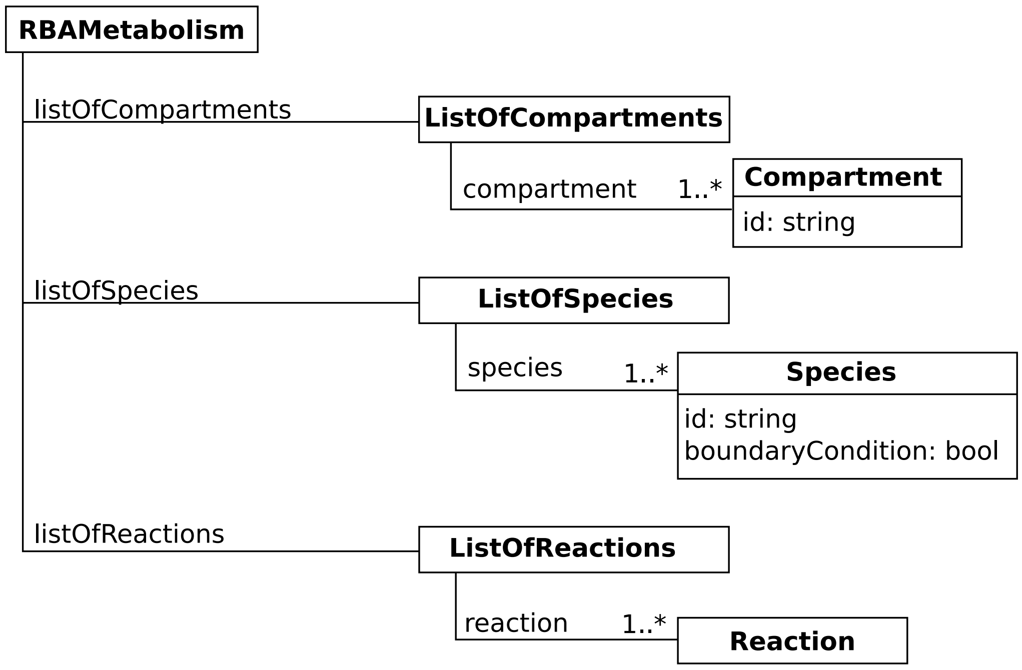

Figure 1: XML structure of metabolism document.

3.2 Compartment

The **Compartment** class is used to list existing cell compartments.

The *id* attribute The **id** attribute is a string defining the identifier of a compartment.

3.3 Species

The **Species** class is used to define *metabolic* species.

The *id* attribute The **id** attribute is a string defining the identifier of a metabolite.

The *boundaryCondition* attribute The **boundaryCondition** attribute is a boolean. If the attribute is set to true, the metabolite is considered to be at a constant concentration. In other words, it is not affected by reactions. This is typical for metabolites in the external medium.

3.4 Reaction

The **Reaction** class is used to define metabolic reactions (Fig. 2). Reactants and products are defined using a **ListOfSpeciesReferences**.

The *id* attribute The **id** attribute is a string defining the identifier of a reaction.

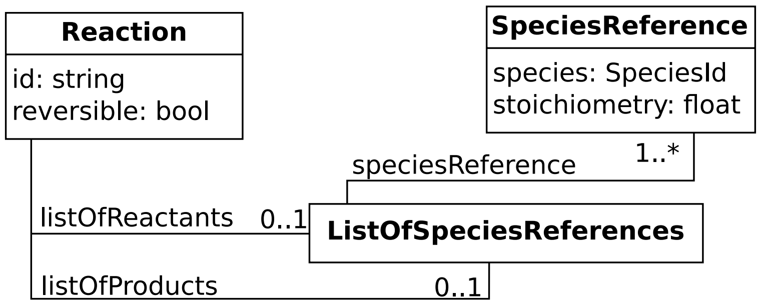

Figure 2: Class storing metabolic reactions.

The *reversible* attribute The **reversible** attribute is a boolean. If the attribute is set to true, the reaction can occur in both directions. If the attribute is set to false, only the forward reaction can occur.

3.5 SpeciesReference

The **SpeciesReference** class is used to refer to a metabolic **Species** and associate with it a stoichiometry (Fig. 2).

The *species* attribute The **species** attribute must match the identifier of a **Species**.

The *stoichiometry* attribute The **stoichiometry** is a positive real number. It repensents the stoichiometry of a **Species** in a given context (typically a **Reaction**).

4 parameters.xml

The parameter file contains density constraints for the RBA model, user-defined parameters and functions.

4.1 RBAParameters

The outermost portion of the parameter file is an instance of class **RBAParameters**, shown in Figure 3.

RBAParameters has no simple attributes. It includes exactly one instance of **ListOf** container classes. All **ListOf** classes do not have own attributes, they are merely used to organize a list of instances from another class.

4.2 TargetDensity

The **TargetDensity** class is used to define density constraints (Fig. 4). In a RBA model, a density constraint defines how many molecules a given compartment can contain. It inherits **TargetValue** for the constraint definition part.

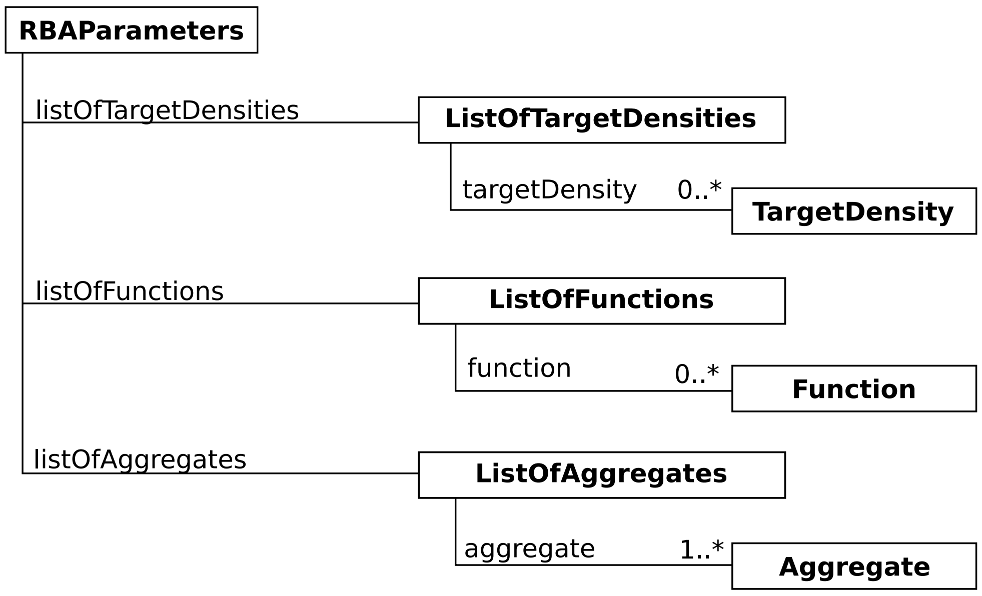

Figure 3: XML structure of parameter document.

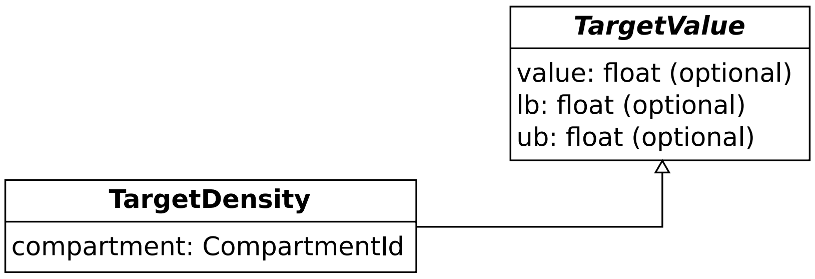

Figure 4: Class used to store density constraints.

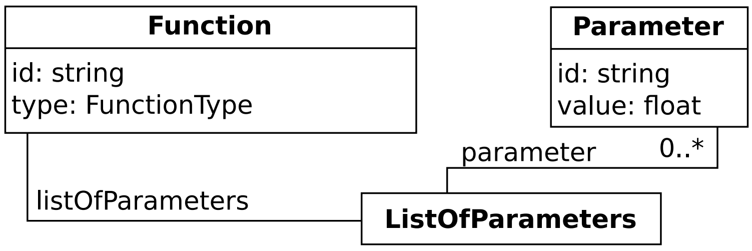

Figure 5: Class used to store user-defined functions.

The *compartment* attribute The **compartment** attribute must match the identifier of a **Compartment**.

4.3 TargetValue

The **TargetValue** class is used to define the sign of an additional RBA constraint and the value of its second member (Fig. 4). It is designed to be inherited. The child class usually holds information about the first member of the constraint (*e.g.* compartment for a density constraint, metabolite for a production constraint).

The *value*, *lb* and *ub* attributes Every attribute can be left undefined, set to a real number or contain the identifier of a **Function** or an **Aggregate**. In the latter case, its value will be growth-rate dependent.

If **value** is defined, the constraint is an equally constraint. **lb** and **ub** are ignored. If **value** is undefined, **lb** (resp. **ub**) defines a lower bound (resp. upper bound) inequality constraint. Note that **lb** and **ub** may both be defined, yielding two separate inequality constraints.

4.4 Function

The **Function** class is used for user-defined functions and parameters (Fig. 5). Implictly, all functions defined here are functions of *growth rate*. Every function holds a **ListOfParameters**, where **Parameter** are defined according to each type of function.

The *id* attribute The **id** attribute is a string defining the identifier of the function.

The *type* attribute The **type** attribute is a string that must match a known function type. Currently, the supported types are:

- **constant**. Constant function with parameter *CONSTANT*.
- **linear**. Linear function with parameters *LINEAR_ CONSTANT*, *LINEAR_ COEF*, *X_ MIN*, *X_ MAX*, *Y_ MIN*, *Y_ MAX*. The 4 last parameters are used to saturate the function. If the variable (growth rate) is outside of the [X_MIN, X_MAX]

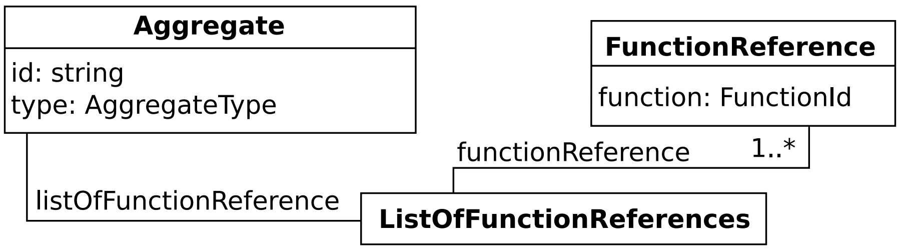

Figure 6: Class used to store user-defined aggregates.

range, it is set to the closest value in that range. The same applies for the return value and the [Y_MIN, Y_MAX] range.

- **exponential**. Exponential function with parameter *RATE*.
- **indicator**. Indicator function with parameters *X_MIN* and *X_MAX*. This function returns one if the variable (growth rate) is in the [X_MIN, X_MAX], zero otherwise.
- **michaelisMenten**. Michaelis Menten function with parameters *kmax*, *Km* and *Y_MIN* (optional). If Y_MIN is defined, any return value lower than Y_MIN will be set to Y_MIN.

4.5 Parameter

The **Parameter** class is used to store the values of function parameters (Fig. 5).

The *id* attribute The **id** attribute is a string that should match a valid parameter identifier. The list of valid parameters for each type of **Function** is listed above.

The *value* attribute The **value** attribute is a real number representing the value of the attribute.

4.6 Aggregate

The **Aggregate** class is used to assemble user-defined functions (Fig. 6). Every aggregate holds a **ListOfFunctionReferences**, where each **FunctionReference** refers to a previously defined function.

The *id* attribute The **id** attribute is a string defining the identifier of the aggregate.

The *type* attribute The **type** attribute is a string that must match a known aggregate type. Currently, the supported types are:

- **multiplication**. The result is the multiplication of the values returned by the function listed in the aggregate at current growth rate.

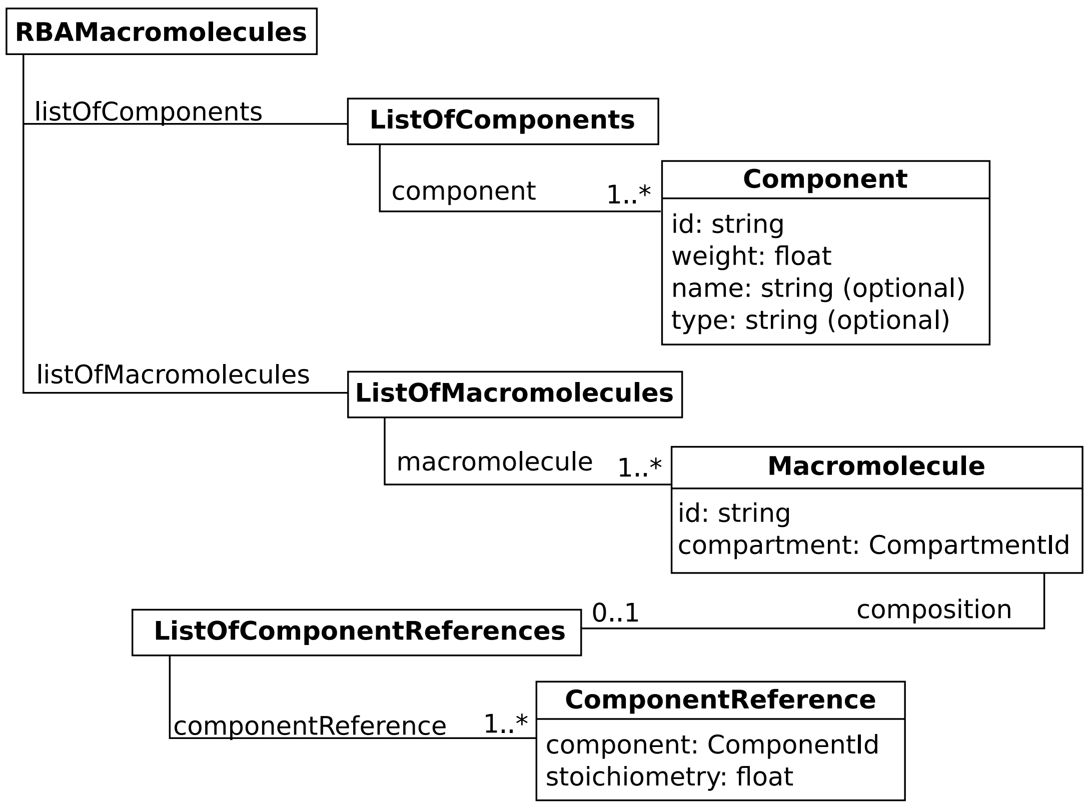

Figure 7: XML structure of macromolecule document.

4.7 FunctionReference

The **FunctionReference** class is used to refer to a user-defined **Function** (Fig. 6).

The *function* attribute The **function** attribute is a string that must match the identifier of a user-defined **Function**.

5 proteins.xml, rnas.xml and dna.xml

All these files use instances of the class **RBAMacromolecules**.

5.1 RBAMacromolecules

The outermost portion of the metabolism file is an instance of class **RBAMacromolecules**, shown in Figure 7.

RBAMacromolecules has no simple attributes. It contains exactly one instance of **ListOfComponents** and **ListOfMacromolecules**.

5.2 Component

The **Component** class is used to define the components of a **Macromolecule** (Fig. 7). For example, these are expected to be amino acids, vitamins and ions for proteins. Note that they use separate identifiers as metabolic **Species**. **ComponentMaps** define how they are assembled from metabolites.

The *id* attribute The **id** attribute is a string defining the identifier of a component.

The *weight* attribute The **weight** attribute is a real number defining the weight of a component. This information is essential for the density constraints. The weight of a macromolecule is defined as the sum of the weight of its components. Note that units used for the weight constraint must be consistent across files.

The *name* and *type* attributes The **name** and **type** attributes are string that provide additional information about the component. The name is a standard name of the component (*e.g.* full amino acid name). The type can be used to distinguish components if necessary (*e.g.* in amino acids, vitamins, ions).

5.3 Macromolecule

The **Macromolecule** class is used to define macromolecular species (Fig. 7). Its composition is given by a **ListOfComponentReferences**.

The *id* attribute The **id** attribute is a string defining the identifier of the macromolecule.

The *compartment* attribute The **compartment** attribute must match the identifier of a **Compartment**. It represents the compartment where the molecule is thought to be active.

5.4 ComponentReference

The **ComponentReference** class is used to refer to a **Component** and associate with it a stoichiometry (Fig. 7).

The *component* attribute The **component** attribute must match the identifier of a **Component** defined in the same **RBAMacromolecules** instance.

The *stoichiometry* attribute The **stoichiometry** is a positive real number. It repensents the stoichiometry of a **Component** in a given context (typically the composition of a **Macromolecule**).

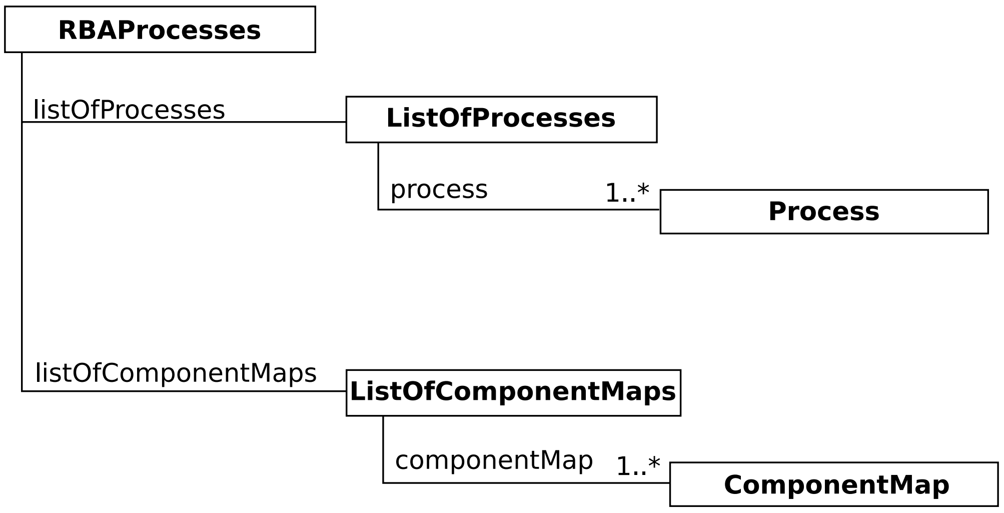

Figure 8: XML structure of process document.

6 processes.xml

The process file is used to define constraints related to complex cellular processes.

6.1 RBAProcesses

The outermost portion of the process file is an instance of class **RBAProcesses**, shown in Figure 8.

RBAMacromolecules has no simple attributes. It contains exactly one instance of **ListOfProcesses** and **ListOfComponentMaps**.

6.2 Process

The **Process** class is used to define cellular processes (Fig. 9). Processes cover a wide variety of non-metabolic reactions.

A **Process** revolves around 3 optional substructures. We encourage you to study existing examples to understand how the different substructures are used.

The **Machinery** is the molecular entity enabling the process (*e.g.* ribosome for translation.) Each machinery unit has a limited production/degradation capacity. Every target of a process has an intrinsic metabolite cost (metabolites needed to produce/degrade it and byproducts). However, if a machinery is defined, there is an additional metabolic cost to produce the machinery that will enable the production/degradation of the target. This is similar to the production of **Enzyme**s in order to catalyze metabolic **Reaction**s.

Operations define the sets of macromolecules that a process produces or degrades. A **Macromolecule** can only be produced if it was related to a **Process**. For example, a protein can only be produced if there is a translation process where proteins are explicitly listed as a production in **Operations**. **Operations** break down **Macromolecule**s in

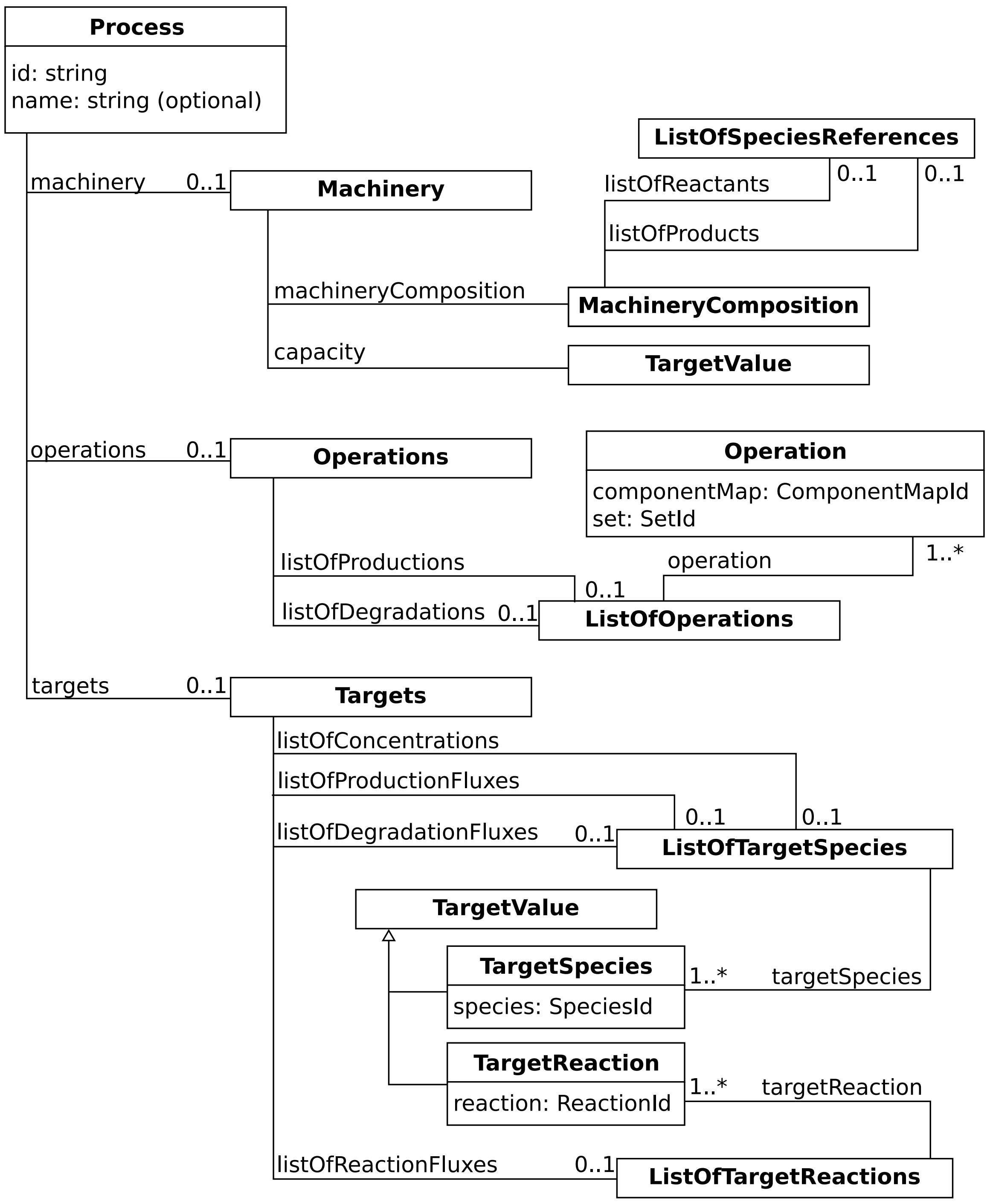

Figure 9: Class used to store processes.

metabolic **Species** and **Machinery** costs. **Operations** does not define how many molecules of a **Macromolecule** must be produced.

Targets define the fluxes that a process must maintain for the cell to work properly. Some fluxes, such as enzyme production, are already enclosed in the base model. They should not appear in **Targets**. On the other hand, fluxes such as production of housekeeping proteins need to be explicitly added to **Targets**.

The *id* attribute The **id** attribute is a string defining the identifier of a process.

The *name* attirbute The **name** attribute is a string that can be used to give the process a more human understandable name.

6.3 Machinery

The **Machinery** class defines the machinery used by a process (Fig. 9). **Machinery** has no simple attributes. If a **Machinery** is defined, it defines a *capacity constraint*. Every **Machinery** unit has a metabolic cost defined by a **MachineryComposition**. Every unit also has a capacity defined by a **TargetValue**. The capacity defines how many targets a **Machinery** can process in 1 unit of time. Total capacity (base capacity multiplied by number of **Machinery** units) must always exceed the number of targets produced.

6.4 MachineryComposition

The **MachineryComposition** class defines the assembly costs of a complex molecular machinery (Fig. 9). **MachineryComposition** has no simple attributes. It contains two **ListOfSpeciesReferences**. One is for reactants, the other for byproducts of the assembly reaction. Note that in this case, **SpeciesReference**s can refer to *both* metabolic **Species** and **Macromolecule**s. The assembly reaction should contain obvious components of the machinery, but also metabolic costs related to assembly (such as ATP/GTP costs) *unless* these costs are already covered by a process.

6.5 Operations

The **Operations** relates **Macromolecule**s production/degradation to some given **Process** (Fig. 9). **Operations** has no simple attributes. It may contain two **ListOfOperations**, one for production and one for degradation.

6.6 Operation

The **Operations** class defines how **Macromolecule**s are produced/degraded (Fig. 9). **Operation** is used to break down **Macromolecule**s into metabolites by linking them to a **ComponentMap**.

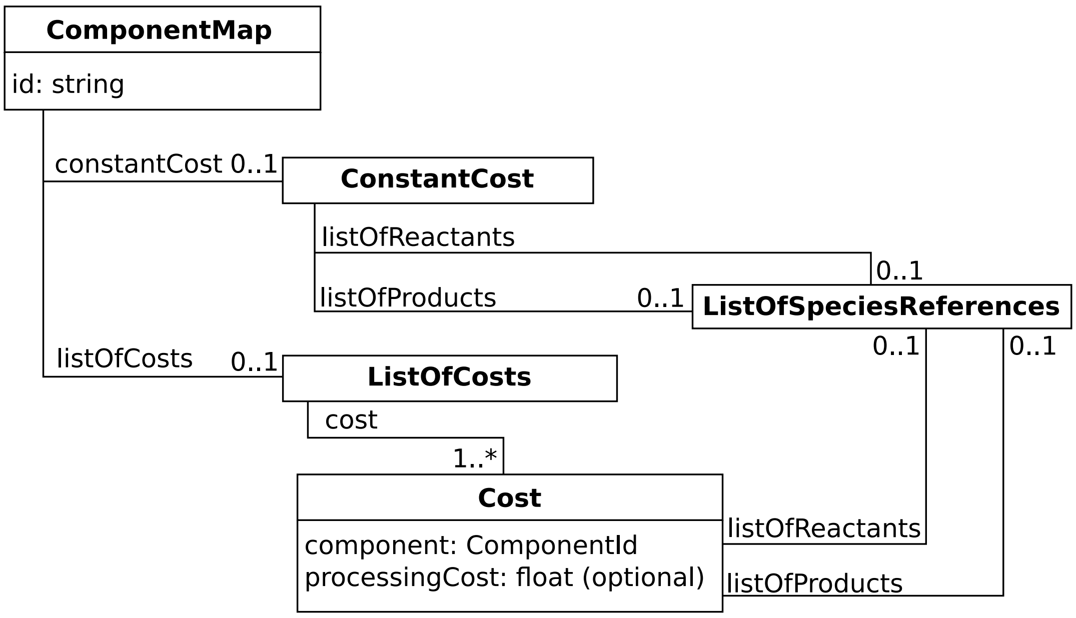

Figure 10: Class used to store production/degradation of macromolecules.

The *componentMap* attribute The **componentMap** attribute must match the identifier of a **ComponentMap**. This **ComponentMap** will be used to compute the metabolic costs.

The *set* attribute The **set** attribute must refer to a **Macromolecule** set. Currently, the only acceptable values are **protein**, **rna** and **dna**. The set contains the **Macromolecule**s that are being produced/degraded.

6.7 ComponentMap

The **ComponentMap** class is used to convert **Macromolecules** in metabolic and machinery costs (Fig.10).

There are two types of costs. The **ConstantCost** is a purely metabolic cost. It contains metabolites that are consumed/produced independent of the **Macromolecule**'s length (*e.g.* translation initiation). The **ListOfCosts** container details **Cost**s depending on the individual **Component**s of the **Macromolecule**. They cover metabolic costs used to assemble the **Component** onto the nascent **Macromolecule**. They also cover processing costs, *i.e.* how many **Machinery** units are needed to assemble the **Component**.

The *id* attribute The **id** attribute is a string defining the identifier of a component map.

6.8 ConstantCost

The **ConstantCost** class defines metabolites consumed and byproducts generated by an assembly process (Fig.10). It contains two **ListOfSpeciesReferences**, one for metabolites consumed and one for metabolites produced. Note that in this context, a **SpeciesReference** must refer to a metabolic **Species**.

6.9 Cost

The **Cost** class defines metabolites consumed and byproducts generated when assembling a specific **Component** (Fig.10). It contains two **ListOfSpeciesReferences**, one for metabolites consumed and one for metabolites produced. Note that in this context, a **SpeciesReference** must refer to a metabolic **Species**. Additionally, it defines a processing cost used in a **Machinery**'s capacity constraint.

The *component* attribute The **component** attribute is a string that must match the identifier of a **Component**.

The *processingCost* attribute The **processingCost** attribute is a real value that is used to compute how many **Machinery** units are needed to assemble the **Component**. For example, let the processing cost of an amino acid be 1. The capacity of the **Machinery** (the ribosome) is the number of amino acids it can assemble per unit of time. The processing cost allows to compute how many ribosomes are needed to produce the **Component** and, in the end, the **Macromolecule** (in this example the number of amino acids divided by the ribosome's capacity).

6.10 Targets

The **Targets** class defines what molecules a **Process** must produce for the cell to work properly (Fig. 9). **Targets** has no simple attributes.

It contains 3 **ListOfTargetSpecies**. These targets allow to define metabolic **Species** or **Macromolecule** fluxes. One is for production fluxes, another for degradation fluxes. The last list is for maintaining a target at a given concentration. The difference with a simple production flux is that keeping a target at a concentration depends on growth rate. More precisely, the flux needed to keep the concentration is the growth rate multiplied by the target concentration. Note that all fluxes must be positive. If the target is a **Macromolecule**, production/degradation can only occur if an **Operations** section of some **Process** defines how the **Macromolecule** is actually produced/degraded.

It contains a **ListOfTargetReactions**. It is also possible to define target fluxes as reaction fluxes. These targets add constraints on the flux of a specific metabolic **Reaction**. In this case, fluxes may be positive or negative.

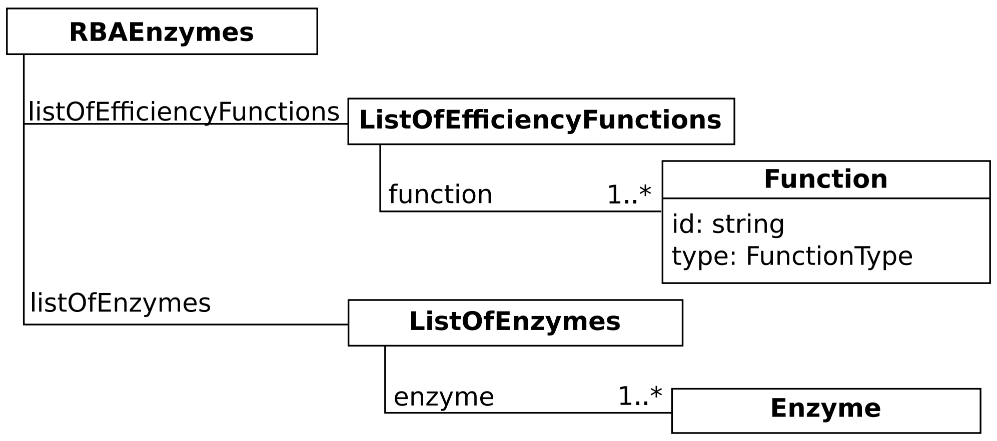

Figure 11: XML structure of enzyme document.

6.11 TargetSpecies

The **TargetSpecies** class defines constraints for a species flux (Fig. 9). It inherits **TargetValue** for the constraint definition part, allowing for equality or inequality constraints.

The *species* attribute The **species** attribute is a string that must match the identifier of a metabolic **Species** or a **Macromolecule**. Note that the **Macromolecule** must be broken down into metabolite costs through the **Operations** section of some **Process**. Otherwise no cost will be applied.

6.12 TargetReaction

The **TargetReaction** class defines constraints for a species flux (Fig. 9). It inherits **TargetValue** for the constraint definition part, allowing for equality or inequality constraints.

The *reaction* attribute The **reaction** attribute is a string that must match the identifier of a metabolic **Reaction**.

7 enzymes.xml

The enzyme file is used to define enzyme composition and efficiency.

7.1 RBAEnzymes

The outermost portion of the metabolism file is an instance of class **RBAenzymes**, shown in Figure 11.

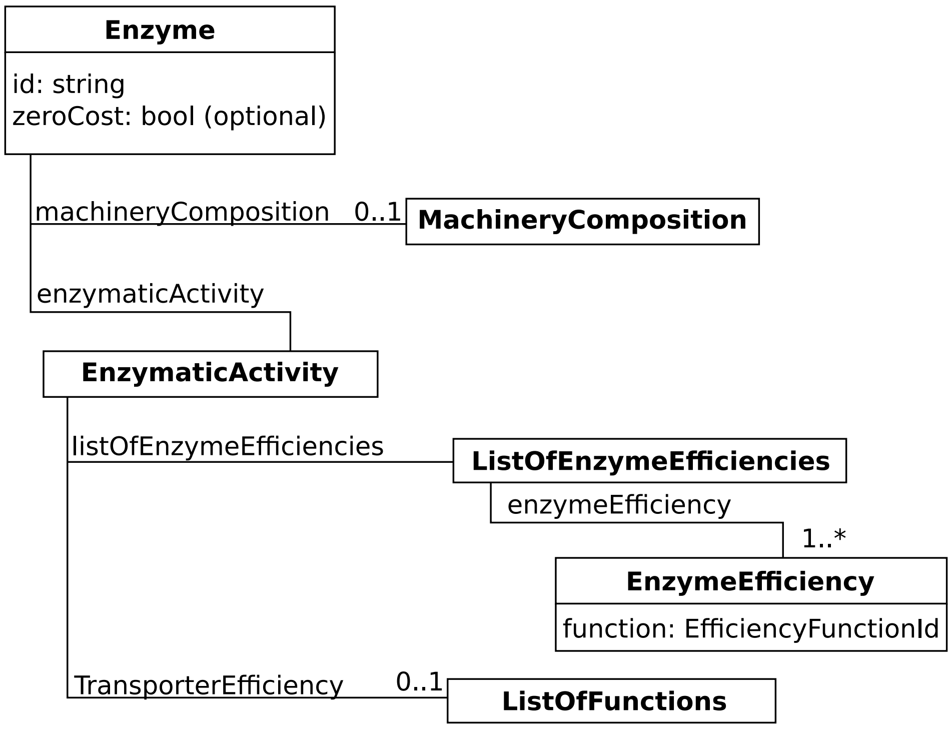

Figure 12: Class storing enzyme information.

RBAenzymes has no simple attributes. It contains exactly one instance of **ListOf-EfficiencyFunctions**.

It contains excatly one instance of **ListOfEnzymes**.

7.2 Enzyme

The **Enzyme** class is used to define enzymes (Fig. 12).

Its composition is given by a **ListOfComponentReferences**.

The *id* attribute The **id** attribute is a string defining the identifier of the macro-molecule.

The *compartment* attribute The **compartment** attribute must match the identifier of a **Compartment**. It represents the compartment where the molecule is thought to be active.

7.3 EnzymeEfficiency

The **EnzymeEfficiency** class is used to (Fig. 12).

The *function* attribute The **function** attribute must match the identifier of a **Component** defined in the same **RBAMacromolecules** instance.