SBML Editor's Report: Principles for Package Development

History

‣ SBML development 'outsourced' to packages in L3.

‣ Said long ago that community would vote on packages.

‣ Community: vote on **package proposals**

‣ Editors: approve **final specifications**

‣ Criteria for vote was simple:

> ‣ Need
>
> ‣ General approach

‣ Generally worked, but one package (dynamic models) was approved with only two paragraphs of speculative text.

New plan: Principles of SBML Development

New plan: Principles of SBML Development

1 Introduction

This document defines Version 1 of the Systems Biology Markup Language (SBML) Level 3 Core, an electronic model representation format for systems biology. SBML is oriented towards describing biological processes of the sort common in research on a number of topics, including metabolic pathways, cell signaling pathways, and many others. SBML is defined neutrally with respect to programming languages and software encoding; however, it is oriented primarily towards allowing models to be encoded using XML, the eXtensible Markup Language (Bray et al., 2004). This document contains many examples of SBML models written in XML. Formal schemas describing the syntax of SBML, as well as other materials and software are available from the SBML project web site, http://sbml.org/.

The SBML project is not an attempt to define a universal language for representing quantitative models. The rapidly evolving views of biological function, coupled with the vigorous rates at which new computational techniques and individual tools are being developed today, are incompatible with a one-size-fits-all idea of a universal language. A more realistic alternative is to acknowledge the diversity of approaches and methods being explored by different software tool developers, and seek a common intermediate format—a *lingua franca*—enabling communication of the most essential aspects of the models.

The definition of the model description language presented here does not specify *how* programs should communicate or read/write SBML. We assume that for a simulation program to communicate a model encoded in SBML, the program will have to translate its internal data structures to and from SBML, use a suitable transmission medium and protocol, etc., but these issues are outside the scope of this document.

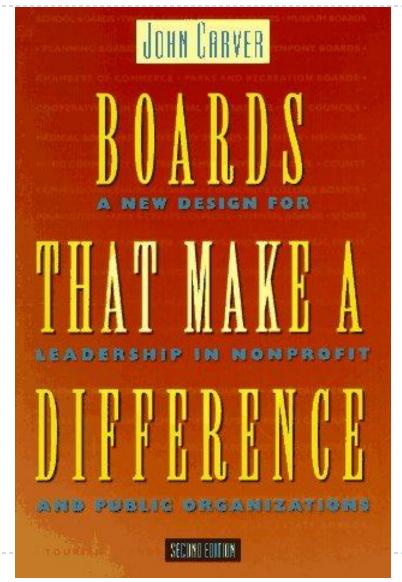### 1.1 Developments, discussions, and notifications of updates

SBML has been, and continues to be, developed in collaboration with an international community of re-

SBML Level 3 Specification Introduction

New plan: Principles of SBML Development

New plan: Principles of SBML Development

New plan: Principles of SBML Development

http://sbml.org/New_dev_process

Three principle types

- ## Architectural principles
  - ### Fundamental design decisions
- ## Community principles
  - ### How development should proceed
- ## Structural principles
  - ### Specific design decisions

Architectural principles

- Used by *community* to evaluate *proposals* when voting

- Example: *Orthogonality*

  - Are all concepts encoded only once?

  - Are existing concepts from core and other packages re-used?

Community Principles

- Used by *package working group* during development

- *Community involvement:*
  - People should be invited to be part of the package working group (pwg)
  - The pwg should keep the community at large informed of progress and decisions.

Structural Principles

▸ Used by *SBML editors* to evaluate *final specification*

Effective abstractions

- A package must provide ways to store data using the most useful and general abstractions possible, within
  - different mathematical frameworks
  - different software tools
  - different modeling paradigms

- Example: Don't define PDE equations; define diffusion constant, etc.

Implementability

- Can (and will) a developer implement the spec correctly, completely, and straightforwardly?

- Reason behind 'must have two independent implementations' rule.

Explicitness

▸ No default attributes/children

▸ If something is left undefined, this must mean something different from giving it a value.

▸ Exception: element attributes may inherit values from higher-level elements in the model.

Validity after reduction

▸ If you strip a package, the resulting core SBML must still be *syntactically* valid

▸ May or may not be mathematically meaningful

▸ (or meaningful in any sense of the word)

▸ Somewhat controversial

Validity after reduction implications

- No SIdRef in core elements may refer to package objects.
- No core MathML may refer to package SIds.
- Core MathML may not be extended by a package.

- Package elements may still extend the SId namespace
- Packages may define new <package:math> elements with extended MathML.

Semantic consistency

- No changing semantics of existing SBML Level 3 Core elements and attributes.

- **KineticLaw** must be extent/time
- **Species** must be substance

Rejected principle

- Apply 'validity after reduction' to packages
  - If you strip just one package, must remaining packages be valid? (no)
  - 'comp' rules about organization are allowed to have dangling references if a different package not understood.

Controversy

- Is 'validity after reduction' useful at all?
- Arguments:
  - If the math is different (required="true") why assume the model could possibly be helpful?
  - Awkward to follow in many situations
- Proposal: each SIdRef definition tells you whether it can point to a package SId
  - Math: no
  - Initial Assignment: yes
  - Annotations: yes

Controversy II

- Is 'semantic consistency' helpful?
- Argument:
  - 'spatial' actually does currently change the units of a KineticLaw, and notes that it did with a flag. Again, if 'required="true"' is on, why assume anything about the interpretation of the KineticLaw?
  - Otherwise makes for awkward design.
- Proposal: a flag to tell you if the semantics are different