

Modular Modeling of cellular systems



Martin Ginkel, Jörg Stelling
Max-Planck-Institute for Dynamics of complex technical Systems
Magdeburg, Germany

1st June 2001

Overview

1. Motivation
2. In praise of Modularity
3. Modular Modeling in other Modeling Tools
4. Possible Realization in SBML2
5. Conclusion

Motivation

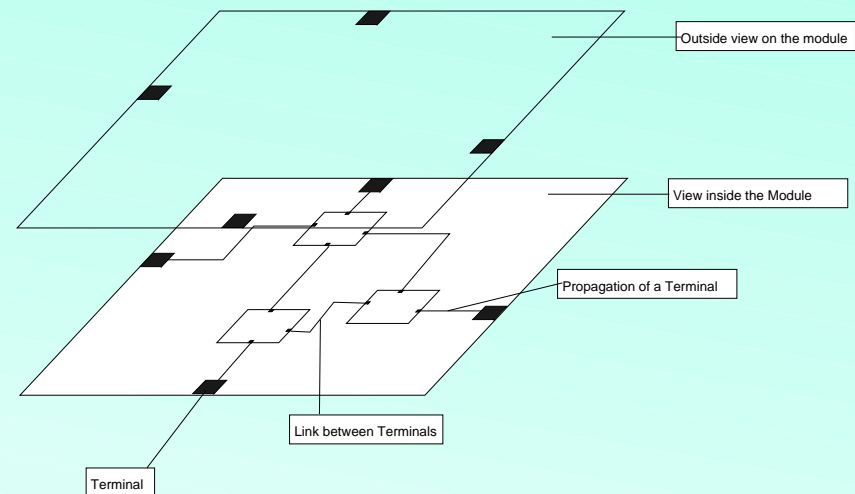
- Goal of Systems Biology:
Understanding the quantitative behaviour of complex cellular systems
- Models itself are complex networks with substances and reactions, couplings and feedbacks
- Working in teams and division of tasks is necessary

- *Are flat and complex reaction maps the model of choice?*

- What about
 - abstractions from the network level
 - representations for higher order structures
 - representation for functional units
 - reuse of (partial) models

In praise of Modularity

1. Modular models are divided in comprehensible modules
 - one can concentrate on his object of interest in the model
 - unnecessary detail is hidden from the user
 - Combining partial models of different modelers becomes easier
2. It is possible to separate interface and implementation of a module
 - Different implementations with the same interface \leadsto Exchangability
 - Example: Gene expression
 - PDE model for Polymerase movement
 - DAE model with time delay
 - transcription frequency = initiation frequency
 - Adjusting the detail level to the purpose of the model



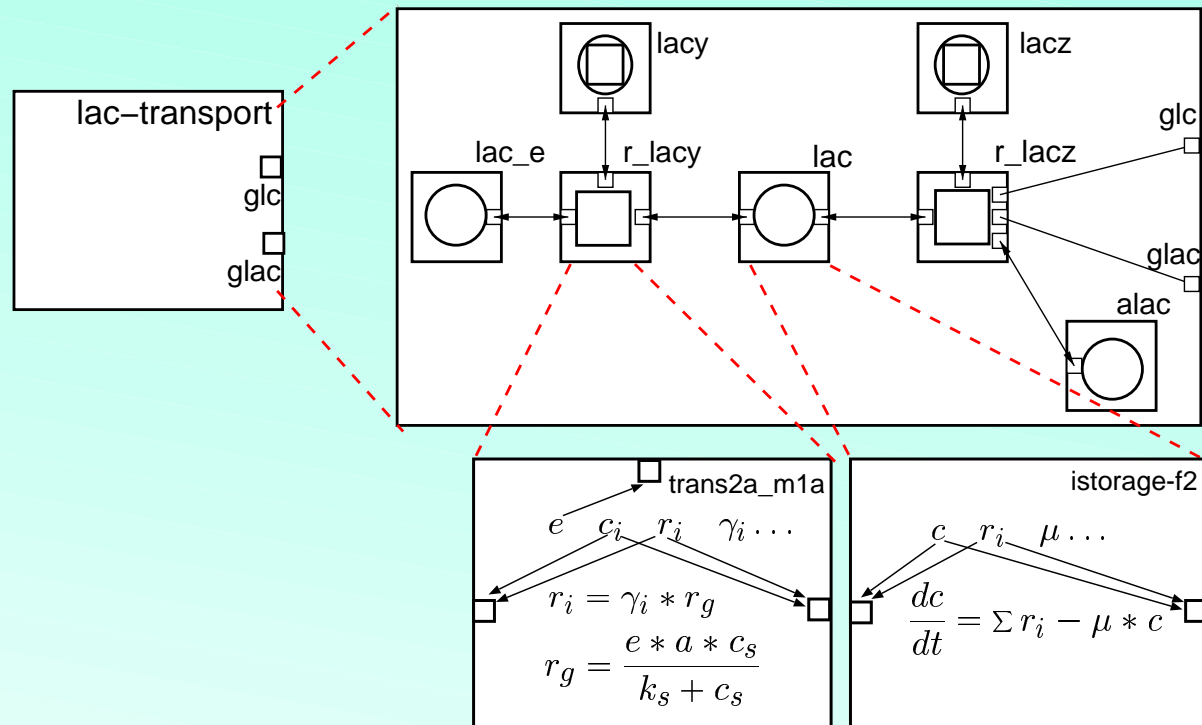
In praise of Modularity

3. Debugging is easier with modularity
 - For debugging one can isolate the module of interest from the rest of the system and study the I/O behaviour in a test frame
 - It is possible to compare the behaviour of different implementations of a module
4. Established standardized models in libraries improve understanding and development of larger models very much
 - Abstraction of the detailed level possible
 - Not lots of reactions but named modules like e. g. „gene-expression“ or „mapk-cascade“, which have to be connected and parametrized
 - reuse of modules becomes possible \leadsto modeling libraries

Modularity in other Modeling Tools

ProMoT: process modeling tool

- equation-based (DAE)
- originally process engineering
- encapsulated modules
object-oriented inheritance
- terminals as interfaces
exporting variables
- links connecting terminals \rightsquigarrow coupling equations

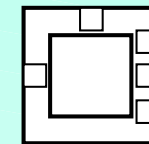
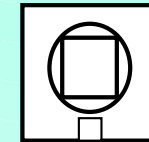
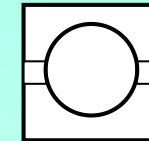


Basic Modules

Substance Storages: Intracellular substances, described by differential equations, 2 kinds:

Intermediates: Substrates, precursors (no genetic information)

Macromolecules: Proteins, DNA, RNA (genetic information)

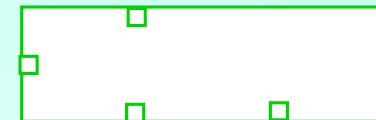


Substance Transformers: Reactions controlled by enzymes
2 complementing aspects:

Structure Flows of substances and respective stoichiometric parameters

Kinetic Reaction rate and controlling enzymes and ligands

Signal Transformers: Processes affecting concentrations of enzymes
~> control of substance transformers



Terminals: 2 basic kinds – substance flows and signals
(concentrations)

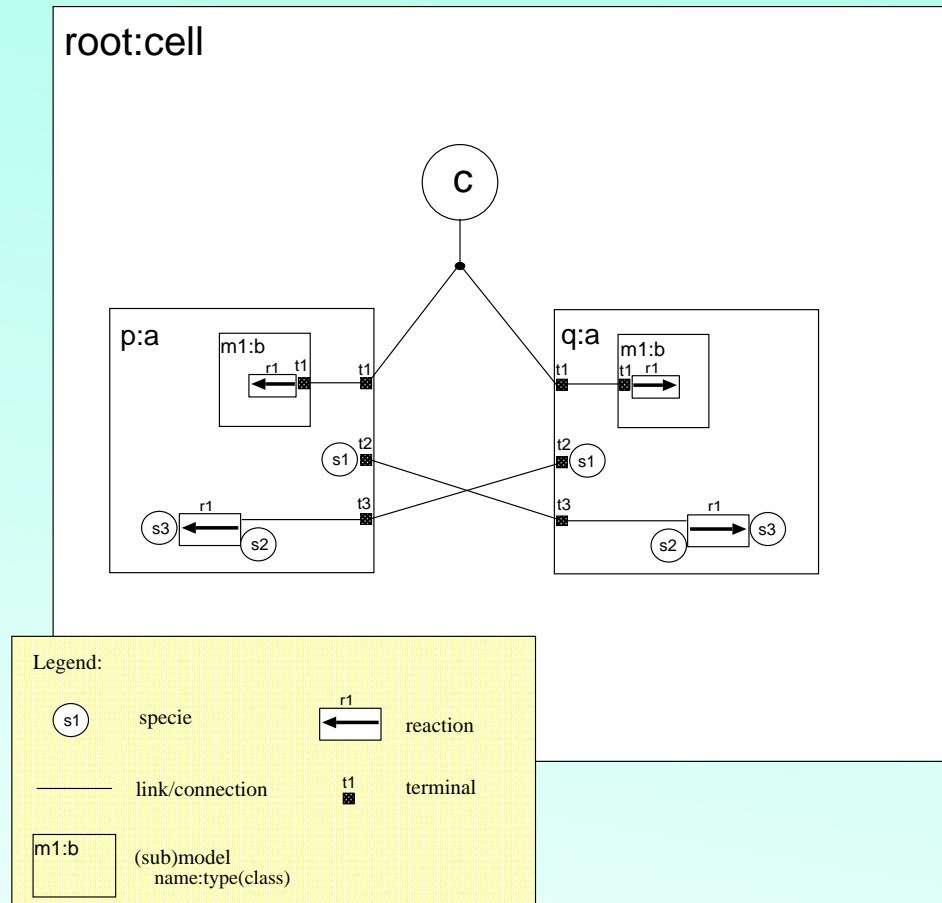
Modular Modeling in other Domains

- **gProms:** general process modeling and simulation environment
 - Purpose: chemical plant simulation
 - modular models and „streams“ as interfaces
 - binary stream connections
- **Modelica:** mechatronic modeling for automation (roboters and vehicles)
 - Purpose: modeling of mixed systems with electrics, mechanic, pneumatics etc.
 - object-oriented classes
 - interfaces: „connectors“
 - multiple connections of interfaces working with
 - * potential variables that are set equal
 - * flow variables that sum to 0
 - ~> very flexible connections

Possible Realization in SBML2

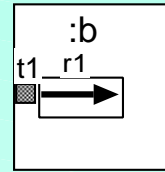
- Prerequisites:
 - Model structure in SBML1: Species and Reactions
 - Reactions hold references to Substances
 - Substances hold references to Compartments
 - Proposal of A. Finney: named (Sub-) Models and multiple Modelinstances
- Idea:
 - Models may have *terminals* representing inner species to the outside or outer species to the inside respectively
 - *links* establish connections between terminals of Modelinstances
 - Modelinstances may contain nearly all attribute info for species and reactions \rightsquigarrow Parametrization

Example Structure



Model in SBML

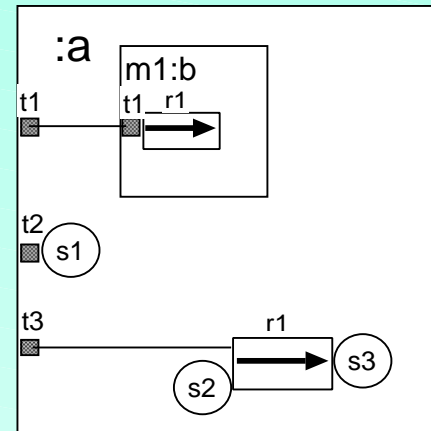
```
<sbml version="2">
  <model name="cell">
    <listOfModels>
      <model name="b">
        <listOfTerminals>
          <terminal name="t1">
        </listOfTerminals>
        <listOfReactions>
          <reaction name="r1">
            <listOfReactants>
              <specieReference terminal="t1">
            </listOfReactants>
          </reaction>
        </listOfReactions>
      </model>
    </listOfModels>
  </model>
</sbml>
```



- model of type “b” is defined
- contains terminal “t1” that references an external specie
- this acts as reactant in reaction “r1”

Model in SBML

```
<model name="a">
  ...
  <listOfModelInstances>
    <ModelInstance name="m1" model="b">
  </listOfModelInstances>
  <listOfSpecies>
    <specie name="s1">
    ...
  </listOfSpecies>
  <listOfTerminals>
    <terminal name="t1" terminal="m1.t1">
    <terminal name="t2" specie="s1">
    <terminal name="t3">
  </listOfTerminals>
  <listOfReactions>
    ...
  </listOfReactions>
</model>
```



- model "a" uses one instance of "b"
- terminal "b.t1" is propagated as terminal "t1" of "a"
- terminal "t2" exports specie "s1"
- "t3" imports an external specie

Model in SBML

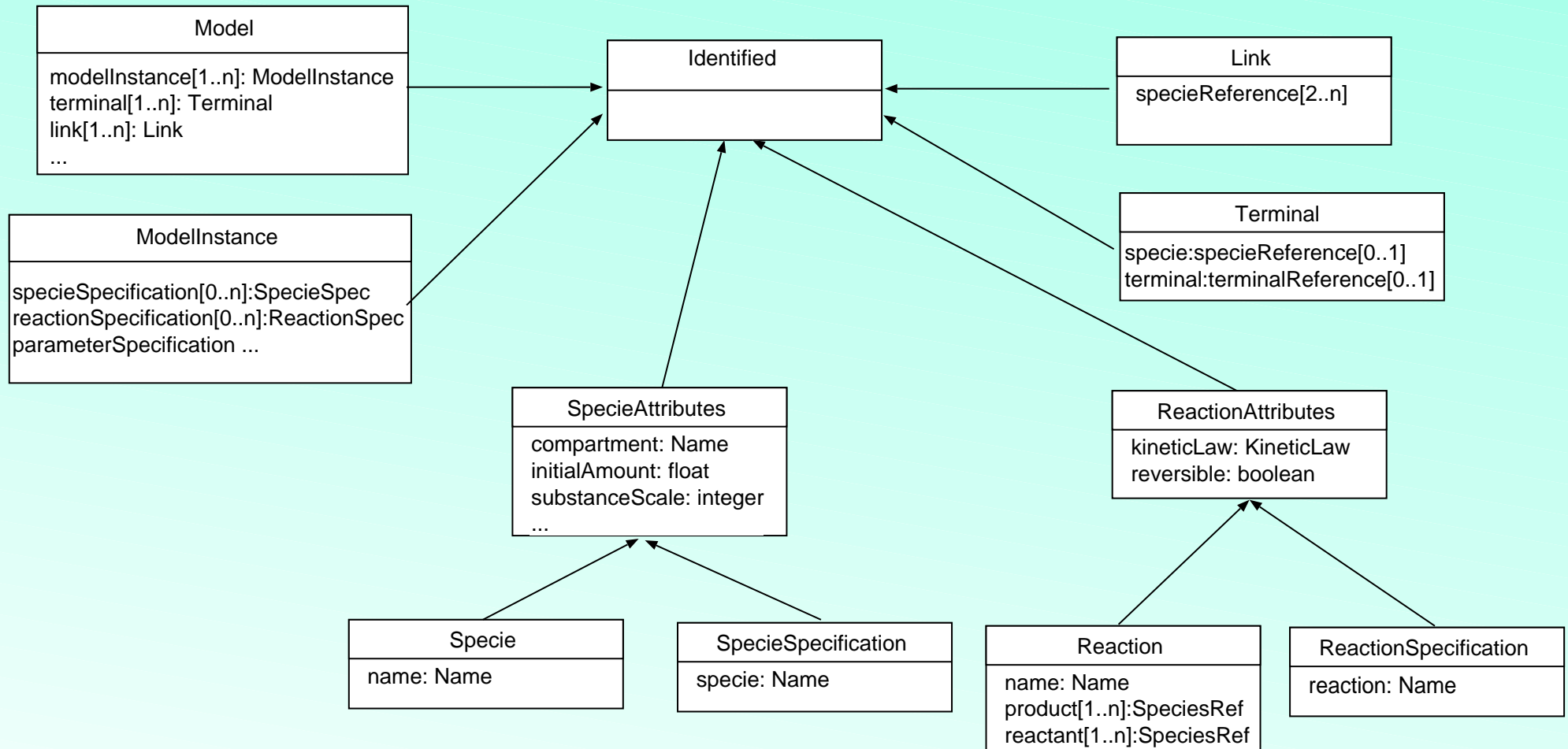
```
<listOfModelInstances>
  <modelInstance name="p" model="a">
    <listOfSpeciesSpecifications>
      <speciesSpecification specie="s1"
        initialAmount=0.1>
    </listOfSpeciesSpecifications>
  </modelInstance>
  ...
<listOfLinks>
  <link>
    <speciesReference terminal="p.t2">
    <speciesReference terminal="q.t3">
  </link>
  ...
  <link>
    <speciesReference specie="c">
    <speciesReference terminal="p.t1">
    <speciesReference terminal="q.t1">
  </link>
</listOfLinks>
```

- Whole model uses one specie "c" and 2 Instances of "a"
- first link connects a substance of "p" with a reaction in "q"
- second link connects 2 reactions in "p" and "q" with specie "c"

New Language Elements

- `listOfTerminals` and `terminal` for definition of models interfaces.
- `listOfReactants` and `listOfProducts` may contain `speciesReference` with attribute `terminal`
- Several submodels can then be connected with `listOfLinks` and `link`. `links` contain a list of terminals and species that should be connected. Links establish mathematical equations according to Kirchhoff's law: $c_1 = c_2 = c_3$ and $\vec{r}_1 + \vec{r}_2 + \vec{r}_3 = 0$. Precomputation
- `speciesSpecification` and `reactionSpecification` allow change of attributes but not adding of new parts to `modelInstances`

New Language Elements



Conclusions

- Modular modeling has several advantages for modelers
 - comprehensibility
 - maintainability
 - building of libraries
- Hierarchical models are possible in SBML with named models and modelinstances
- terminals define interfaces to the models
- modelinstances should be parametrized with xxx-specifications for known basic parts