LibSBGN
Current Status and Future Plans

Tobias Czauderna, Augustin Luna,
Martijn van Iersel

HARMONY 2011, NYC

Background Context & Motivation

WHY LIBSBGN?

Many tools support SBGN

- Arcadia
- Athena
- BiNoM
- BioModels Database
- BioPAX
- BioUML
- ByoDyn
- CellDesigner
- Dunnart
- Edinburgh Pathway Editor
- JWS Online
- Mayday
- Netbuilder (Apostrophe)
- PANTHER
- PathwayLab
- Reactome
- Vanted
- VISIBIOweb
- … 19 tools (and still counting)

Cf. http://sbgn.org/SBGN_Software

The problem with SBGN tools

- **No interchange of maps**
  - But useful features (e.g. validation, layout) may be scattered across tools!
- **No reuse** of **code**
  - The same set of core features (e.g. conversion) is duplicated…

Solution? LibSBGN

- facilitate development of SBGN compliant tools
- increase interoperability
- **Exchange format** for all SBGN maps: **SBGN-ML**
  - XML Schema based
  - express semantics, relationships and geometry
- **Software library** to interact with SBGN maps: **LibSBGN**
  - Java and C++
  - key features: conversion, validation and layout

Development Methods & Infrastructure

HOW IT’S DONE

Community project

- Mirit Aladjem (MIM)
- Frank Bergmann (SBML Layout)
- Michael Blinov (BioNetGen)
- Sarah Boyd (Dunnart)
- Tobias Czauderna (VANTED)
- Emek Demir (Pathway Commons)
- Ugur Dogrusoz (Patika)
- Akira Funahashi (CellDesigner)
- Hiroaki Kitano (CellDesigner)
- Nicolas Le Novère (BioModels Database)
- Augustin Luna (MIM)
- Yukiko Matsuoka (CellDesigner)
- Huaiyu Mi (PANTHER Pathway)
- Stuart Moodie (EPE)
- Falk Schreiber (VANTED)
- Anatoly Sorokin (EPE)
- Martijn van Iersel (PathVisio)
- Alice Villéger (Arcadia)

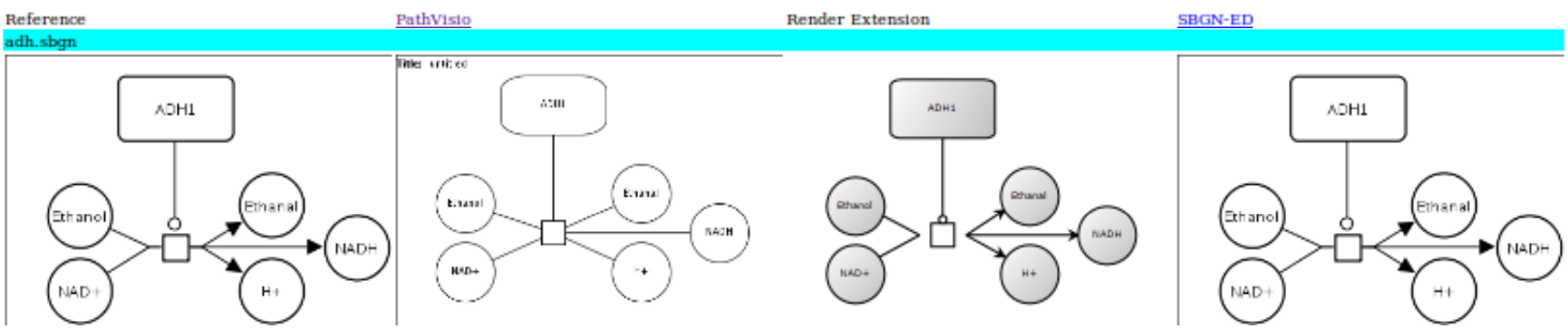
Online workspace

- **Mailing list**: sbgn-libsbgn@lists.sourceforge.net
- **Monthly online meetings**
  - first on Skype, now on **EVO**: http://evo.caltech.edu
  - **minutes** and **announcement** on mailing list
  - **scheduled** on Doodle: http://www.doodle.com/
  - **agenda** on wiki

- SourceForge project: http://libsbgn.sourceforge.net
  - **Wiki**: documentation, road map, "how to", useful links, …
  - **Tracker**: "to do" list (bugs and missing features)
  - **SVN** repository: test suite, specs, XSD
- "Quality control" tools
  - **Automatic XSD validation** against examples on test server http://azraelbigcat.dyndns.org/reports/libsbgn/
  - **Rendering comparison** pipeline http://libsbgn.sourceforge.net/rendering_comparison

Development infrastructure

- **Test suite**: test cases (so far):
  - 22 for PD
  - 17 for ER
  - 0 for AF
  - SBGN diagram (PNG)
  - corresponding SBGN-ML file
- Rendering comparison pipeline



Current Status

WHERE WE ARE

SBGN-ML Roadmap

- **Milestone 1 released (Jan. 2011)**
  - Implement semantics of SBGN PD L1v1.1
  - Only high-level graphics specification
  - Basic validation using XML Schema
- **Milestone 2**
  - Implement semantics for all 3 languages: SBGN PD, ER and AF
  - Extra validation using Schematron
- **Milestone 3**
  - Third-party extensibility
  - Complete graphical specification
- **Milestone 4**
  - Annotations, Linking, MIRIAM compatibility

SBGN-ML Roadmap

- **Milestone 1 released (Jan. 2011)**
  - Implement semantics of SBGN PD L1v1.1
  - Only high-level graphics specification
  - Basic validation using XML Schema
- **Milestone 2**
  - Implement semantics for all 3 languages: SBGN PD, ER and AF
  - Extra validation using Schematron
- **Milestone 3**
  - Third-party extensibility
  - Complete graphical specification
- **Milestone 4**
  - Annotations, Linking, MIRIAM compatibility

Brief SBGN-ML overview

- Main requirements
  - **Easy to draw** (explicit coordinates)
  - **Easy to interpret** (network and semantics)

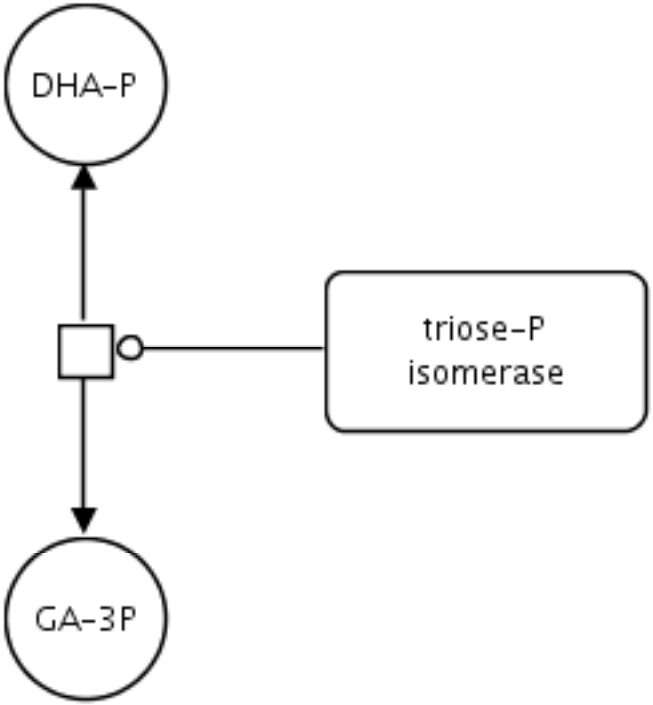
  } Somewhat redundant, but that's a choice

- Only two top-level elements: **Glyph** and **Arc**
  - "class" attribute determines semantic ➔ rendering

- Glyph geometry: **bounding box** only
- Glyph children:
  - label
  - other glyphs (e.g. state variable, unit of information)
  - ports where arcs can connect

- Arcs **refer to glyph** or glyph ports (network connectivity)
- Arcs contain an **optional route** (list of lines and Bezier curves)

Example



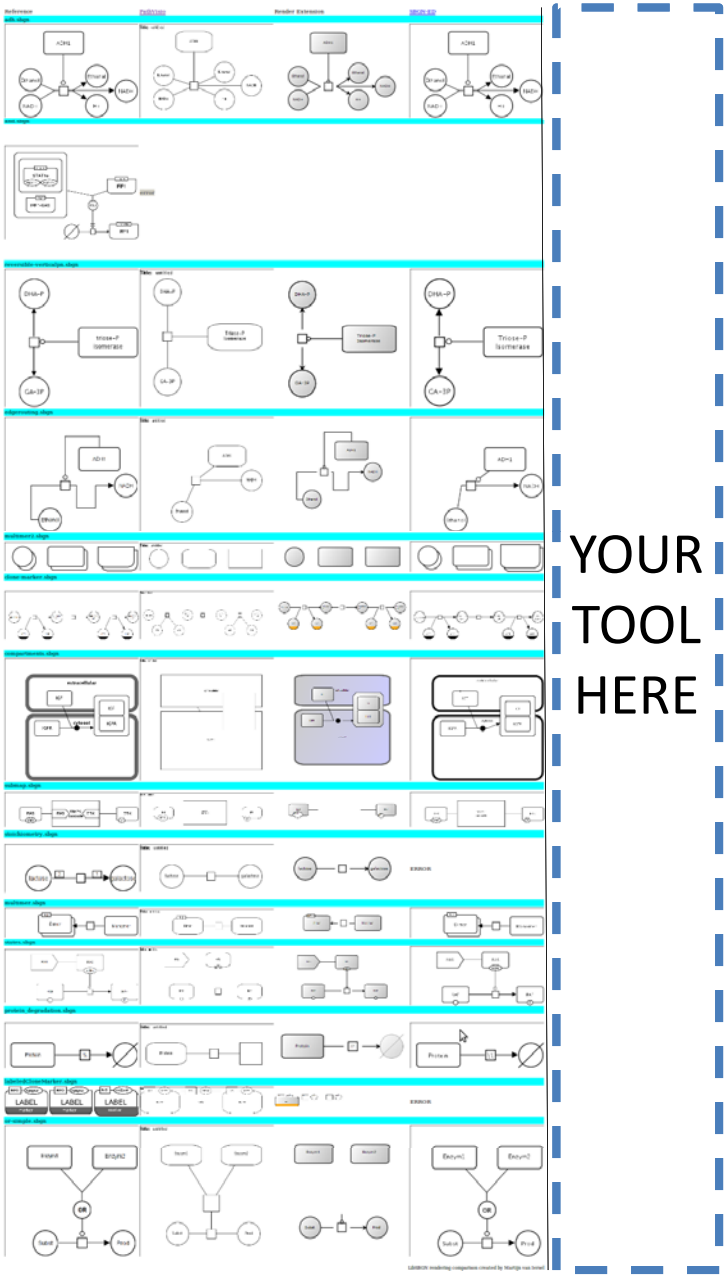

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sbgn xmlns="http://sbgn.org/libsbgn/pd/0.1">

    <glyph class="simple chemical" id="glyph1">
        <label text="DHA-P"/>
        <bbox x="30" y="20" w="60" h="60"/>
    </glyph>
    <glyph class="simple chemical" id="glyph2">
        <label text="GA-3P" />
        <bbox x="30" y="220" w="60" h="60"/>
    </glyph>
    <glyph class="macromolecule" id="glyph3">
        <label text="Triose-P&#xA;Isomerase" /> <!-- contains line break -->
        <bbox x="150" y="120" w="120" h="60"/>
    </glyph>

    <glyph class="process" orientation="vertical" id="pn1">
        <bbox x="50" y="140" w="20" h="20"/>
        <port x="60" y="130" id="pn1.1"/>
        <port x="60" y="170" id="pn1.2"/>
    </glyph>

    <arc class="production">
        <source x="60" y="130" ref="pn1.1" />
        <target x="60" y="80" ref="glyph1" />
    </arc>

    <arc class="production">
        <source x="60" y="170" ref="pn1.2" />
        <target x="60" y="220" ref="glyph2" />
    </arc>

    <arc class="catalysis">
        <source x="150" y="150" ref="glyph3" />
        <target x="70" y="150" ref="pn1" />
    </arc>

</sbgn>
```

Software support

- LibSBGN prototype: **SBGN-ML parser**
  - automatic Java binding with JAXB

- 3 compatible tools
  - PathVisio (Martijn van Iersel)
  - SBGN-ED (Tobias Czauderna)
  - SBML Layout (Frank Bergmann)

➔ featured in the **rendering comparison gallery**

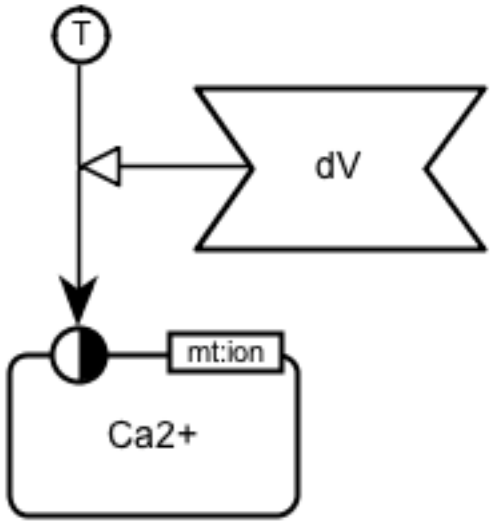


Future Plans

WHAT NEXT?

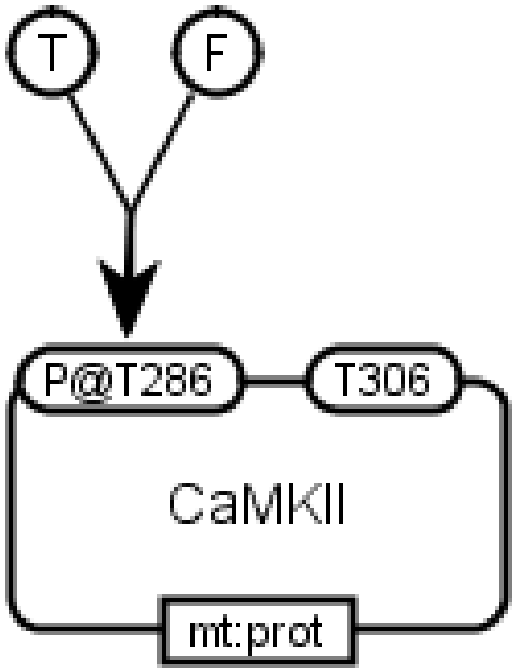
Changes for SBGN-ER

- Target of an influence arc
  - Modeled as a port on the target arc.



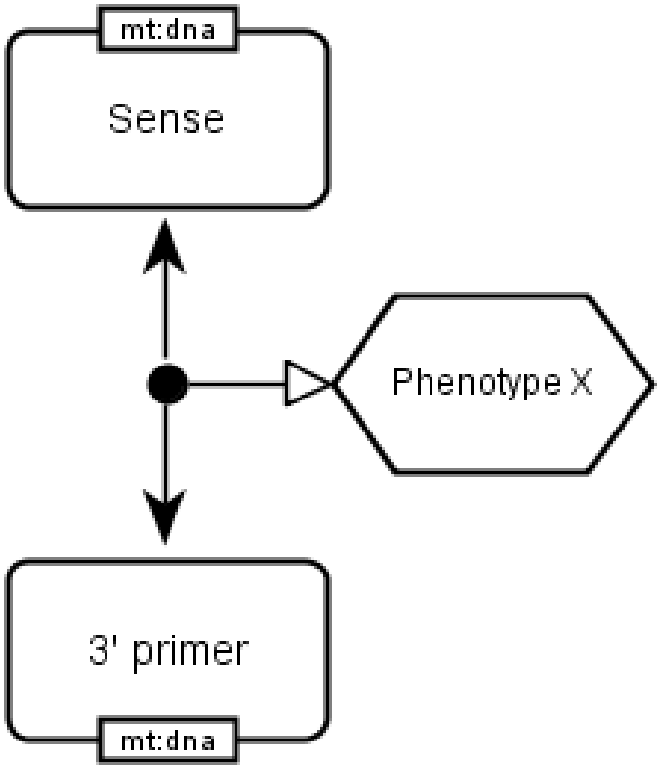
Changes for SBGN-ER

- Variable assignment with multiple values
  - Modeled as an invisible glyph



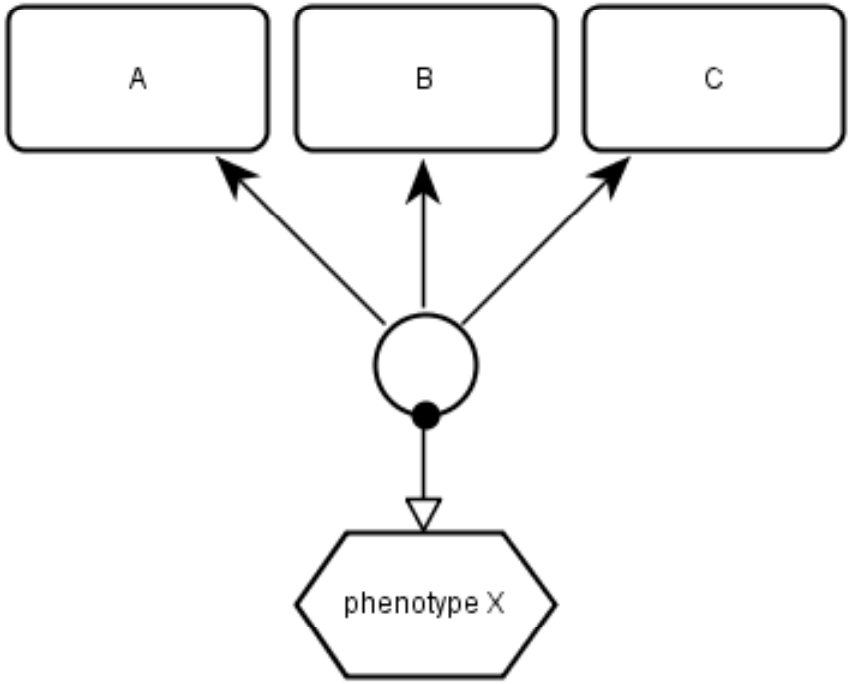
Changes for SBGN-ER

- Outcome
  - Modeled as a sub-glyph of an arc
  - Multiple outcomes (sub-glyphs) allowed per arc



Changes for SBGN-ER

- Binary interaction
  - Modeled as an arc
- Ternary interaction
  - Modeled as three or more arcs plus a glyph
  - Grouped using a special XML element



Changes for SBGN-ER

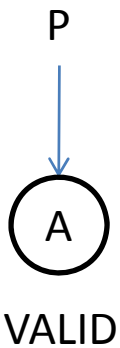
- All three languages will be validated using a **single** XML Schema
- Extra validation rules using schematron

Schematron Overview

- Detects **patterns** in XML.
- Steps in validation
  1. Schematron rulesets converted into **XSL stylesheet**
  2. XML transformed against it -> **validation report**.
- Features
  - Based on **XPath** expressions
  - Uses **assertions** (for errors) and **reports** (for confirmation).
  - rules can be grouped in **phases**.
  - reporting of **diagnostics**.
  - Standardized report language (**SVRL**)

Schematron Rule (for MIM)

```
<!– CovalentModification (SBGN Assignment) -->
<iso:pattern name="check-cvm" id="check-cvm">
<iso:rule
context="mimVis:InteractionGlyph[mimVis:Point[@arrowHead=
'CovalentModification'] and not(mimVis:Point[starts-
with(@arrowHead,'Branching')])]">
<iso:let name="vis-id" value="@visId"/>
<iso:assert test="(
       mimVis:Point[1][@arrowHead='CovalentModification'
] and
       mimVis:Point[last()][@arrowHead='Line']
               ) or (
       mimVis:Point[last()][@arrowHead='CovalentModifica
tion'] and
       mimVis:Point[1][@arrowHead='Line']
       )"                     diagnostics="vis-id"
               >Non-branched interactions possessing a
'CovalentModification' arrowhead should be terminated
with a 'Line' arrowhead.
</iso:assert></iso:rule></iso:pattern>
```



Schematron Report

```
<svrl:fired-rule
context="mimVis:InteractionGlyph[mimVis:Point[@arrowHea
d='CovalentModification'] and not(mimVis:Point[starts-
with(@arrowHead,'Branching')])]"/>
<svrl:failed-assert test="(
mimVis:Point[1][@arrowHead='CovalentModification'] and
mimVis:Point[last()][@arrowHead='Line'] ) or (
mimVis:Point[last()][@arrowHead='CovalentModification']
and mimVis:Point[1][@arrowHead='Line'] )">
<svrl:text>Non-branched interactions possessing a
'CovalentModification' arrowhead should be terminated
with a 'Line' arrowhead.</svrl:text>
<svrl:diagnostic-reference diagnostic="vis-
id">id96b9ed7c</svrl:diagnostic-reference>
</svrl:failed-assert>
```

Cons/Proposals

1. Use Schematron
   - Schematron acts on the file and produces a file, libraries need to export to a file to validate and then need to parse that file to extract errors
   - Schematron may not be capable of dealing with all errors
     - BioPAX validation is not necessarily on patterns in the file
2. Use Schematron (with custom parser to act in memory)
   - Shifts the programming work to a parser
3. Use Java, C++, Groovy, etc
   - Will need to be re-written for each language binding
   - Needs additional format for rule exchange
4. Use a webservice
   - May not always be available

THANK YOU

- To everyone involved so far: **GOOD JOB TEAM**
- To **all developers supporting SBGN** (or planning to):

  feel free to join the club!

  – Use the **library** and support the **schema**
  – Take part in online discussions
  – **Contribute content** to the SourceForge project

http://libsbgn.sourceforge.net