SBFC

The Systems Biology Format Converter Framework

Outline

- Introduction
- Implementation Details
- Applications

Introduction

What is SBFC?

Context

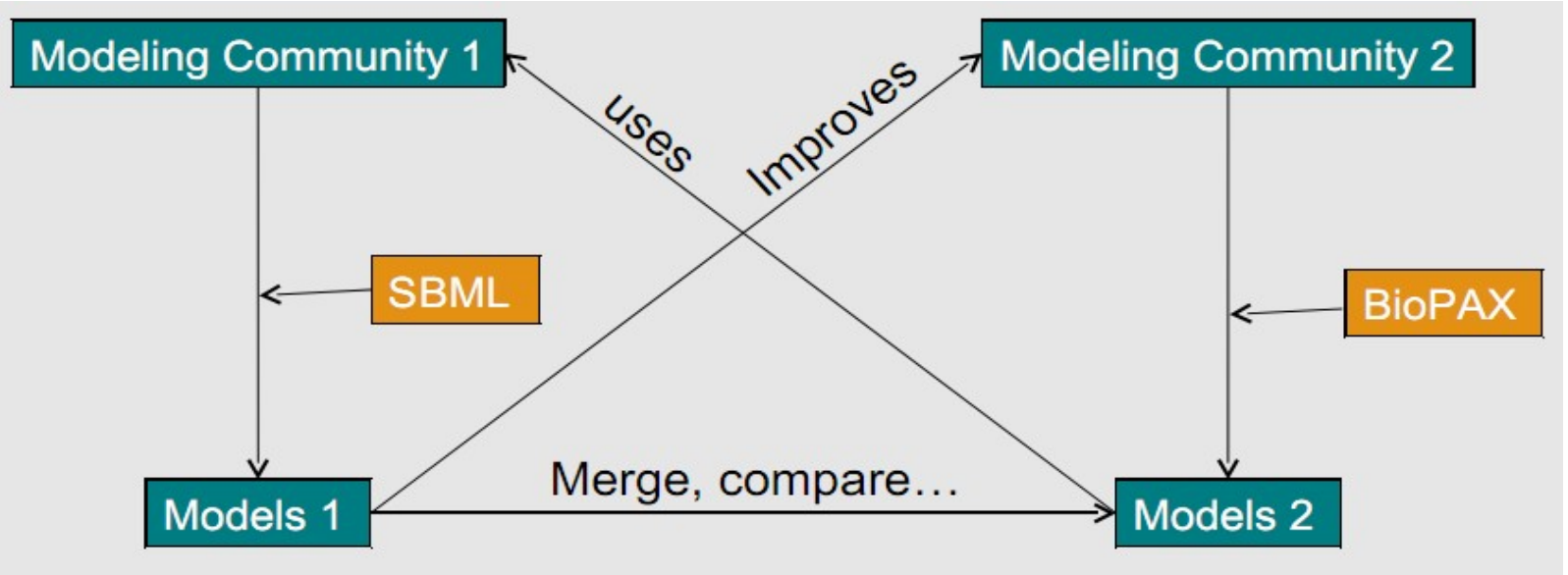
- Computational Modeling in Biology
  - Community with different goals
    - Descriptive models
    - Mathematical models
  - Different Formats
    - SBML, BioPAX, CellML, …
    - Octave, R, Matlab, Mathematica, …
    - SBGN, GPML,...

Context

- Problem of interoperability
- Need for conversion between formats

Context

- Problem of interoperability
- Need for conversion between formats



Problem

- Lots of different formats
- Existing conversion tools by different groups
- Separate programs in different programming languages
- Often integrated in existing tools – not easy to reuse

Goal

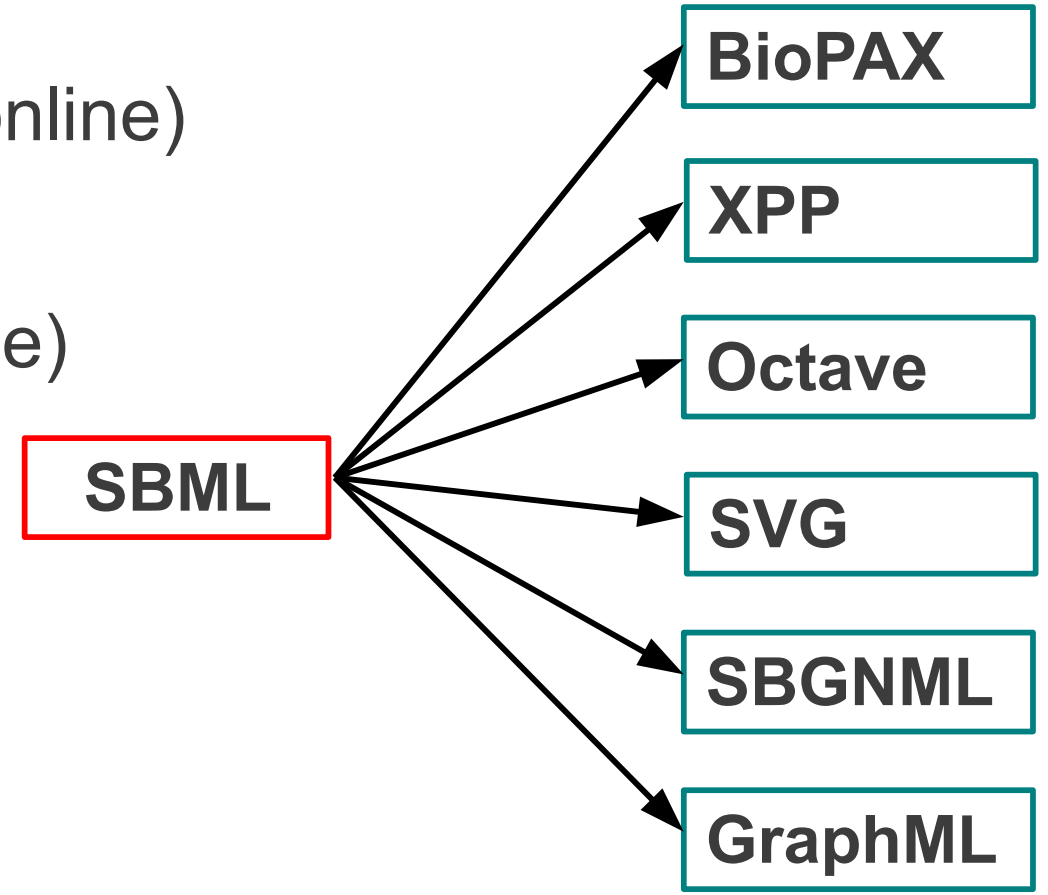
- Generic Framework in Java
  - potentially translate any format into another
  - add new converters easily
  - easy to use locally (command line tool)
  - easy to integrate into existing applications

Goal

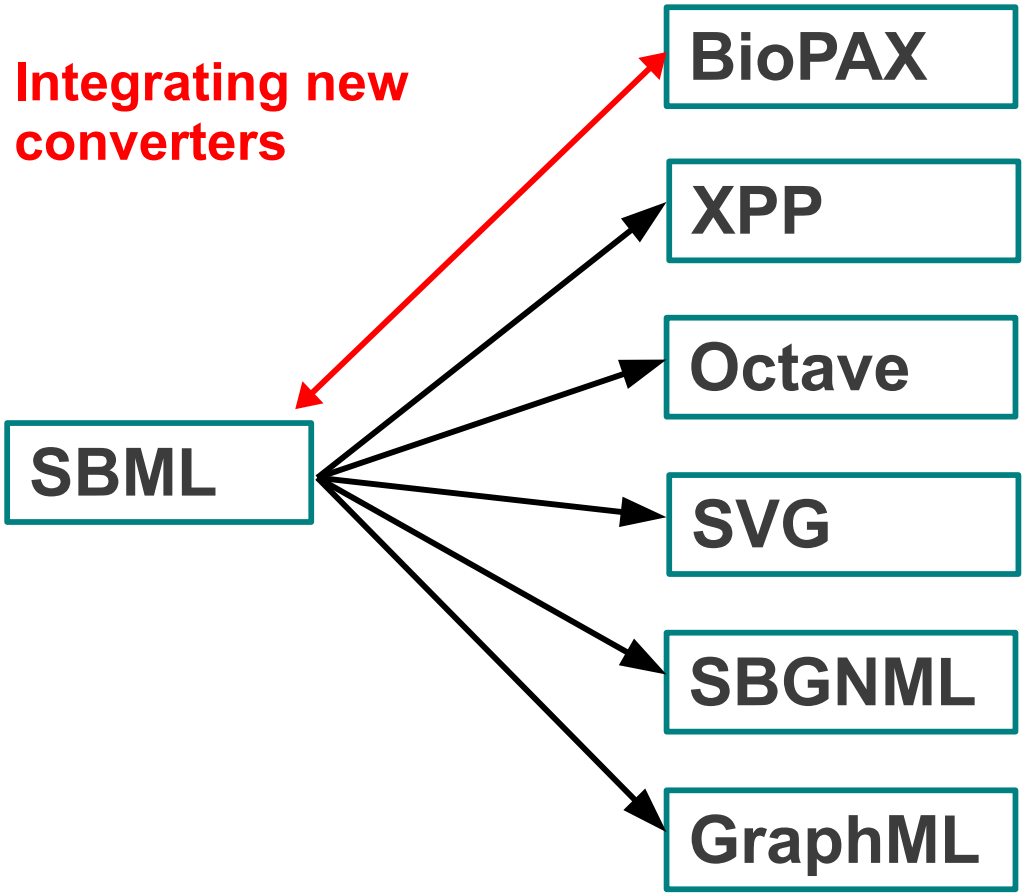
- Web Application
  - model upload (file, URL, copy/paste)
  - Prototype using EBI resources
- Web Service
  - use converters from within applications

Status Update

- Existing converters:
  - SBML 2 BioPAX2/3 (online)
  - SBML 2 XPP (online)
  - SBML 2 Octave (online)
  - SBML 2 SVG
  - SBML 2 SBGNML
  - SBML 2 GraphML

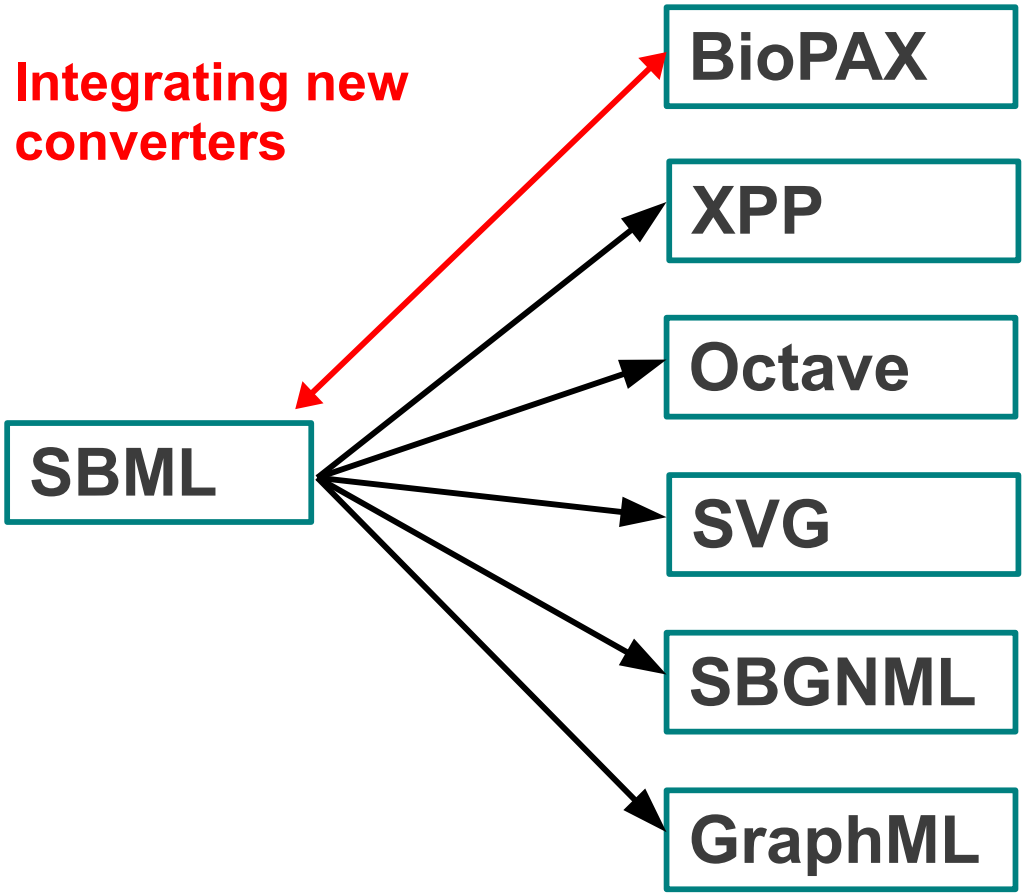


If we go further...

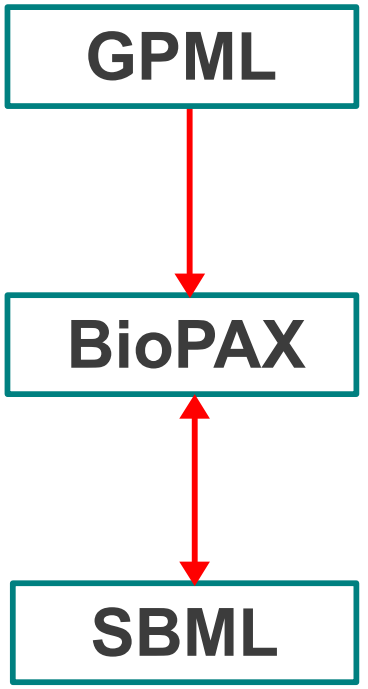


If we go further...

Integrating new converters



Converter Pipeline



Implementation Details

How does it work?

SBFC API

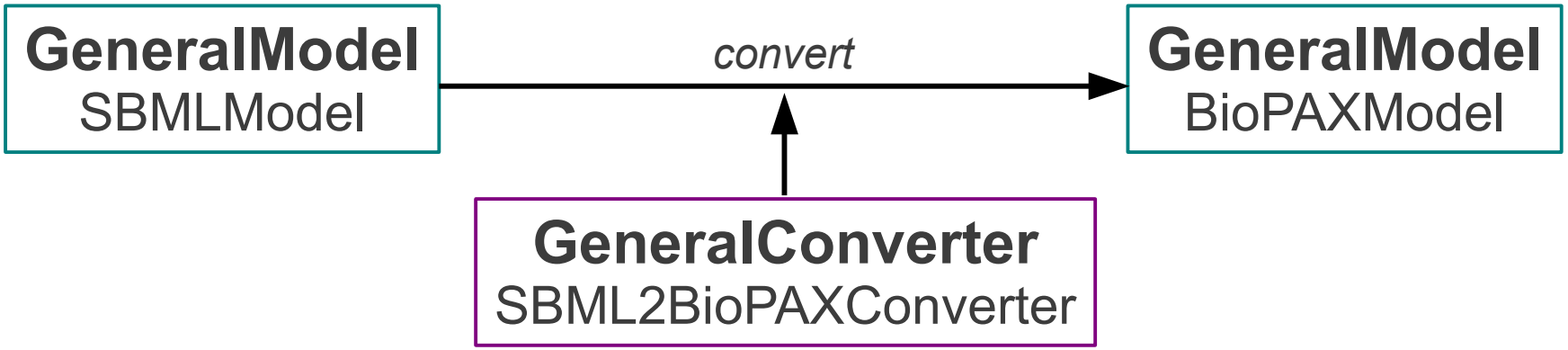
- Interfaces:

  `GeneralModel` and `GeneralConverter`

SBFC API

- Interfaces:

  `GeneralModel` and `GeneralConverter`



SBFC API

- `GeneralModel`
  - read and write methods (from file or from string)
  - method to get the file extension
- `GeneralConverter`
  - input model and output model
  - convert method
  - method to set converter specific options

Modularity

- Recently developed prototype with OSGi
  - Modular and generic framework
  - Easily to add new converters
  - Code reuse
  - Easy integration of converters (or the complete framework) in existing tools
  - Implementation of converter pipelines

Introduction to OSGi

Problem:

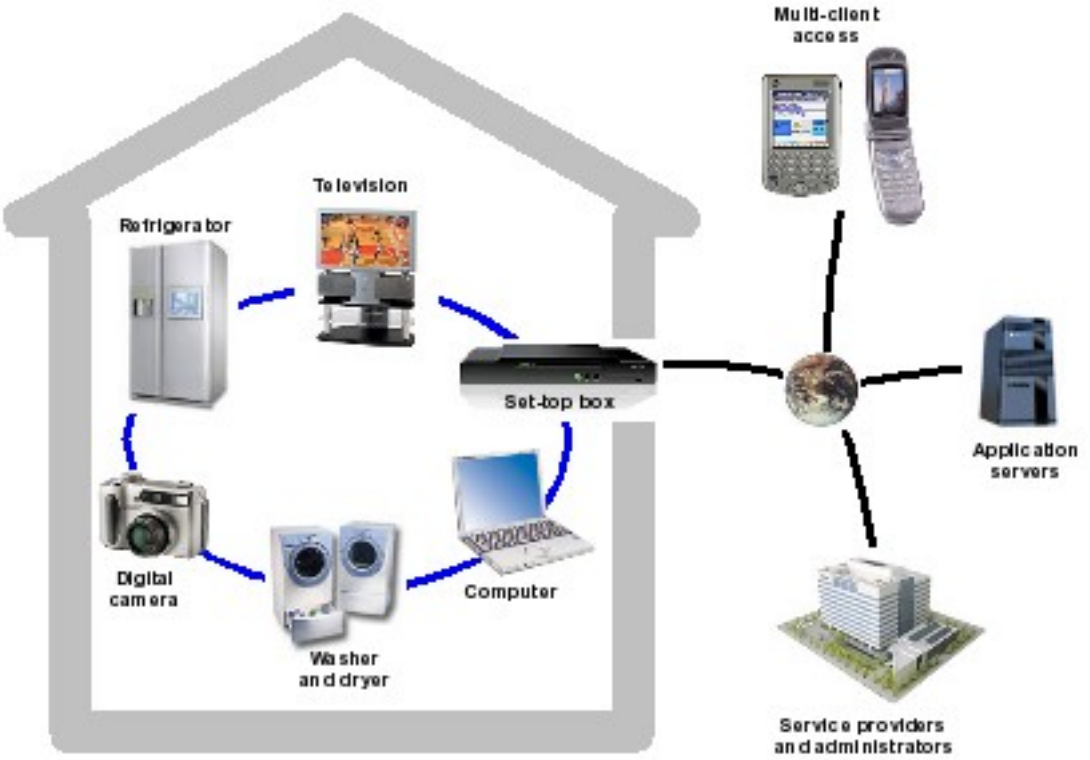
- Increasing software complexity
- Resolving right library at runtime
- Difficult to test/deploy monolithic software
- Developers around the world

Solution:

- Dynamic module system for Java
- Components can declare version of dependencies
- Easy to add/remove modules
- Code reuse

OSGi Architecture

OSGi Architecture



OSGi Architecture

What does change?

- Simple SBFC-API module
- Each model and converter is a separate module (OSGi bundle)
- Main bundle takes care of correct conversion process and knows which converters are available

Applications

How can you use SBFC?

Command line tool

- Convert one file into another format
- Framework automatically loads correct converter
- Use framework as a conversion pipeline (if there is no direct converter available)
  - Not yet implemented!

Integration in Java App

- Every module is a separate jar file
- OSGi application: just load SBFC bundles
- Java Application: add jars to the build path and you can use SBFC functionality

Web Application

- Accessible from internet browser:
  http://www.ebi.ac.uk/compneur-srv/converters/converters
- Models from files, URL or copy/paste
- Running on a server at the EBI
- Conversion jobs are very demanding in terms of CPU and memory

Web Service

- Convert file from one format into another from within an existing program
- Interoperable – can be used from within any application (independent of programming language)

Conclusion

- Collaborative Project
  - Framework to combine format converters
  - Open Source: http://sourceforge.net/projects/sbfc/
- Provide your converters as SBFC converters (implement SBFC-API interfaces)

Acknowledgements

- EBI
  - Nicolas Rodriguez
  - Gaël Jalowicki
  - Jean-Baptiste Pettit
  - Nicolas Le Novere
- Maastricht University
  - Chris Evelo