The ERATO Systems Biology Workbench

Michael Hucka, Andrew Finney, Herbert Sauro, Hamid Bolouri

ERATO Kitano Systems Biology Project
California Institute of Technology, Pasadena, CA, USA

Principal Investigators: John Doyle, Hiroaki Kitano

Collaborators:
Adam Arkin (BioSpice), Dennis Bray (StochSim),
Igor Goryanin (DBsolve), Andreas Kremling (ProMoT/DIVA),
Les Loew (Virtual Cell), Eric Mjolsness (Cellerator),
Pedro Mendes (Gepasi/Copasi), Masaru Tomita (E-CELL)
Overview of Tutorial

• Short intro to the Systems Biology Workbench (SBW)
  – Motivations
  – Technology

• SBW from a user’s perspective, demonstration of
  – SBW core components & features
  – Currently available SBW-enabled modules

• Programming with SBW
  – Using Java
  – Using C, C++
Motivations

• No single package answers all needs of modelers
  – Different packages have different niche strengths reflecting expertise & preferences of the developing group
  – Strengths are often complementary to those of other packages

• No single tool is likely to do so in the near future
  – Range of capabilities needed is large
  – New techniques (⇒ new tools) are evolving too rapidly

• Researchers are likely to continue using multiple packages for the foreseeable future

• Problems with using multiple tools:
  – Simulations & results often cannot be shared or re-used
  – Duplication of software development effort
Goal & Approach

- **Systems Biology Workbench project goal:**
  - **provide software infrastructure that**
    - Enables sharing of simulation/analysis software & models
    - Enables collaboration between software developers

- Initially focused on biochemical modeling

- **Two-pronged approach:**
  - Develop a common model exchange language
    - **SBML**: Systems Biology Markup Language
      - XML-based representation of biochemical networks
  - Develop an environment that enables tools to interact
    - **SBW**: Systems Biology Workbench
      - Uses SBML to transfer models between tools
      - Supports resource sharing
Systems Biology Workbench

• Open-source, integrated software environment that enables sharing of computational resources
  – Allows software developers to easily build interprocess communications facilities into their applications

• From the user’s perspective:
  – One SBW-enabled application can interact with another
  – Each application or module offers services to others
    • E.g.: optimization, time-based simulation, visualization, etc.
Programming SBW

• Numerous desirable features
  – Small application programming interface (API)
  – Simple message-passing architecture
    • Easy to make cross-platform compatible
    • Easy to make distributed
  – Language-neutral architecture
    • We provide C, C++, Java, Python libs for Windows & Linux
    • … but libs can be implemented for any language
  – A registry of services for applications to query
  – Use of well-known, proven technologies
The SBW Framework

- SBW libraries implement RPC mechanisms
  - Provide language bindings for SBW
    - C, C++, Java, Python, etc.
  - Implement underlying message-passing protocols
Modules & Services in SBW

• Modules are separately compiled executables
• Modules may offer one or more Services
• Services consist of one or more Methods

Module
Math

Service Trig
Double sin(Double)
Double cos(Double)

Service Log
Double exp(Double)
Double log(Double)

• Services are categorized into Service Categories
• Services do not have to be categorized
The SBW Registry

• Registry records information about modules
  – Module name
  – How to start module
  – What services the module provides
  – The categorization of those services

• Hierarchy of service categories

![Service Categories](Interface Hierarchy)

![Services](Interfaces)
Why not use CORBA?
- Complexity, size, compatibility
- SBW scheme does not require an elaborate compiled IDL
- No fully-compliant open-source CORBA ORB that supports more than one programming language
- But: we plan to provide a gateway between CORBA & SBW

Why not use SOAP or XML-RPC?
- Performance, data type issues, implementation quality

Why not Java RMI?
- Java-specific

Why not COM?
- Microsoft-specific, low portability
SBW Status

• Available Now:
  – LGPL open-source beta release from http://www.bioinformatics.org/sbw/
  – Java, C, C++, Python libraries
  – Tutorials, developer’s manuals, examples
  – Modules:
    • SBML Network Object Model
    • Gepasi optimization module
    • Jarnac ODE simulator + MCA
    • Plotting
    • Gibson stochastic simulator
    • MATLAB model generator
    • JDesigner visual editor
SBW Future

• To deliver by April 2002
  – LGPL production release
    • Improve quality of beta release code, GUIs & docs
  – C# and Perl libraries
  – Secure distributed operation
  – CORBA access

• Third-party modules under development
  – Bifurcation analysis module
  – Gillespie “Tau-Leap” module
  – GENESIS interface