Overview of the 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), Les Loew (Virtual Cell), Pedro Mendes (Gepasi/Copasi), Masaru Tomita (E-Cell)
Background

• Modeling, simulation & analysis are critical
  – Huge volumes of data
  – Many disparate findings

• Rapid rate of software tool development
  – Roles: data filtering, model creation, model simulation
  – Many groups are creating many tools
    • Different packages have different niche strengths
      reflecting expertise & preferences of the group
    • Strengths are often complementary to those of other packages
Problems

• No single package answers all needs of modelers
• No single tool is likely to do so in the near future
  – Range of capabilities is large
  – New techniques (⇒ new tools) evolving too rapidly
• Researchers are likely to continue using multiple packages for the foreseeable future
• Problems in 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

• Two-pronged approach:
  – Develop a common model exchange language
    • SBML: Systems Biology Markup Language
  – Develop an environment that enables tools to interact
    • SBW: Systems Biology Workbench
Systems Biology Workbench

• Open-source, integrated software environment that enables sharing of computational resources
  – Allows software developers to 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.: ODE solution, time-based simulation, visualization, etc.
From the User’s Perspective
From the User’s Perspective
From the User’s Perspective
Behind the Scenes

Visual Design Tool

SBW Broker

Simulation Control Interface

Simulation Engine
From the Programmer’s Perspective

• 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’ll provide C, C++, Java, Delphi, 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
• **SBW libraries implement** RPC mechanisms
  – Provide *language bindings* for SBW
    • C, C++, C++ Builder, Java, Delphi, Python, etc.
  – Implement underlying message-passing protocols
Communications in SBW

• Message types:
  – **Call**: blocking
  – **Send**: non-blocking
  – **Reply**: reply to a call
  – **Error**: exception handling

• Message payloads:
  – **Call**, **send**, **reply**: one or more data elements
  – **Error**: error code and diagnostic messages

• Data elements are tagged with their types

• Supported data types:
  - Byte
  - Boolean
  - Integer
  - Double
  - String
  - List (heterogeneous)
  - Array (homogeneous)
The SBW Registry

• Registry records info about modules
  – Module name
  – How to start module
  – Which service categories the module provides

• Hierarchy of service categories
Why?

• Why not use CORBA?
  – Complexity, size, compatibility
  – SBW scheme does not require IDL

• Why not use SOAP or XML-RPC?
  – Performance, data type issues, quality of implementations

• Why not Java RMI?
  – Java-specific

• Why not COM?
  – Microsoft-specific, low portability

• Why not MPI?
  – Designed for homogeneous distributed systems rather than heterogeneous
Summary & Availability

• Preliminary test implementation completed
• Production version is now in development
  – Draft API definition & other info available
    • Your hand-outs
    • http://www.cds.caltech.edu/erato/sbw/docs
• Expect first public beta release in November at ICSB 2001 (http://www.icsb2001.org)