The ERATO Systems Biology Workbench

Michael Hucka, Hamid Bolouri, Andrew Finney, Herbert Sauro

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 ($\Rightarrow$ 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 goal: to 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 Markup Language

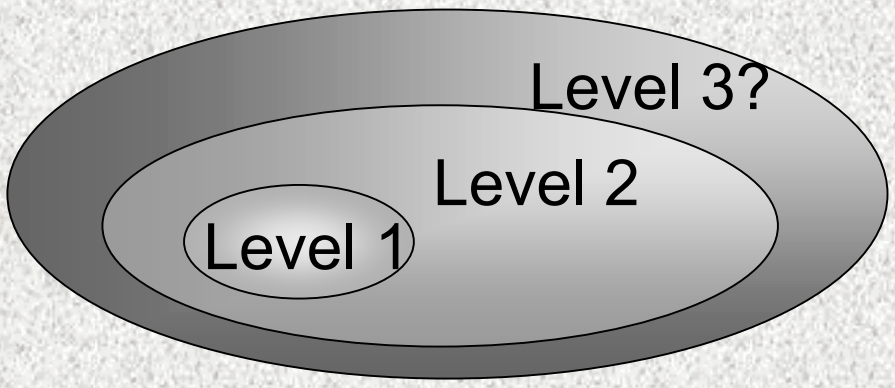
- **Problem:**
  - Many software tools, few common exchange formats
    - Difficult to take advantage of multiple tools
    - Difficult to establish repositories of models
- **A Solution (In Principle):**
  - Define a common exchange language
    - Use a simple, well-supported, textual substrate (XML)
    - Add components that reflect the natural conceptual constructs used by modelers in the domain

Structure of Model Definitions in SBML

- **Domain: biochemical network models**
- **A model is described using a list of components:**
  - Beginning of model definition
    - » List of unit definitions (optional)
    - » List of **compartments**
    - » List of **species**
    - » List of parameters (optional)
    - » List of rules (optional)
    - » List of **reactions**
  - End of model definition
- **Each component has a specific structure**

Some Points About SBML

- **Users do not write in XML — software tools do!**
- **SBML is being defined incrementally**
  - SBML Level 1 covers non-spatial biochemical models
    - Intentionally kept **simple** for maximal compatibility
  - SBML Level 2 will extend Level 1 with more facilities



E.g.:
- Composition
- Geometry
- Arrays

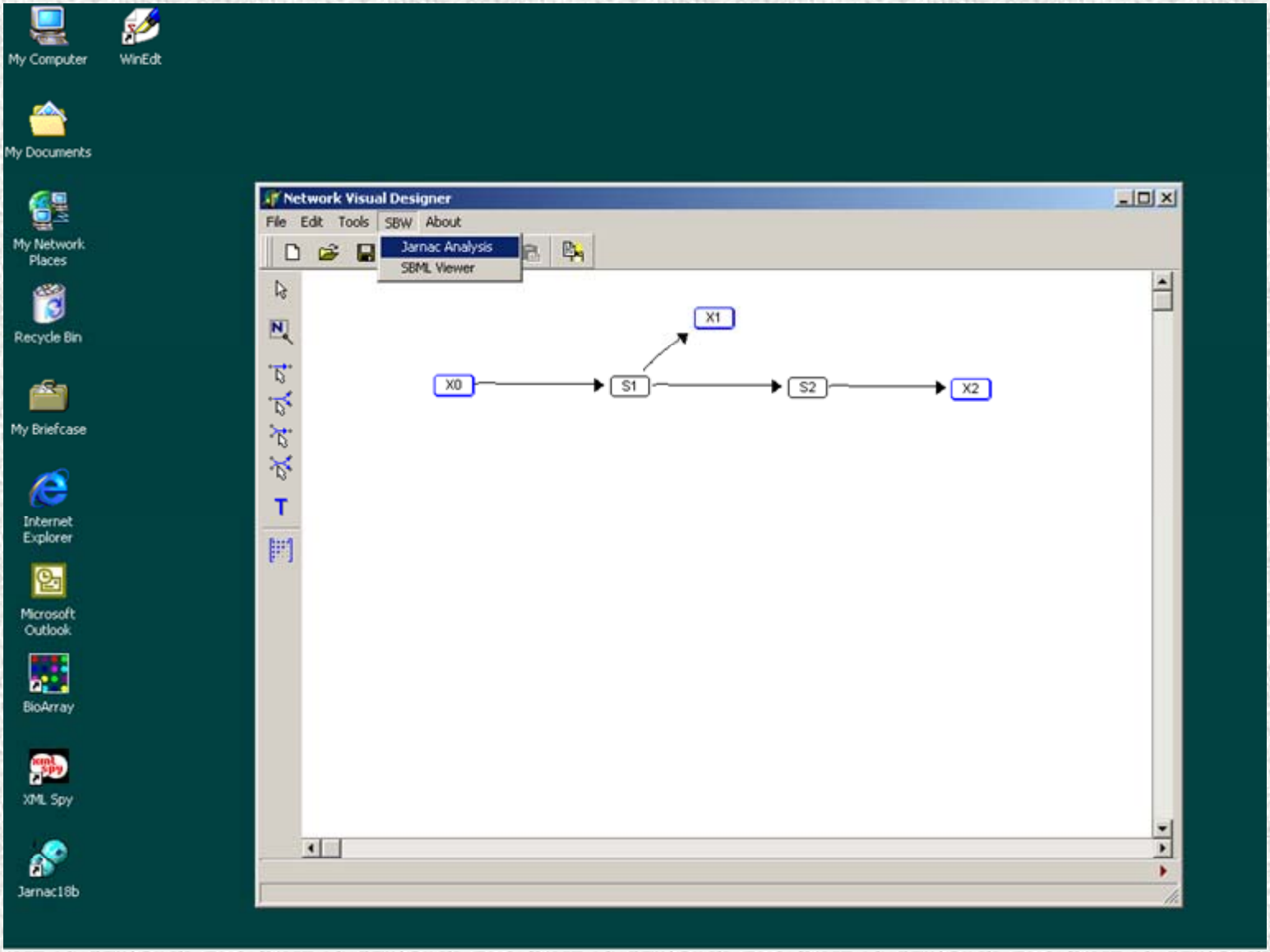
… others

- **Defined in abstract form (UML) + textual descriptions**
  - Used to define XML encoding + XML Schema

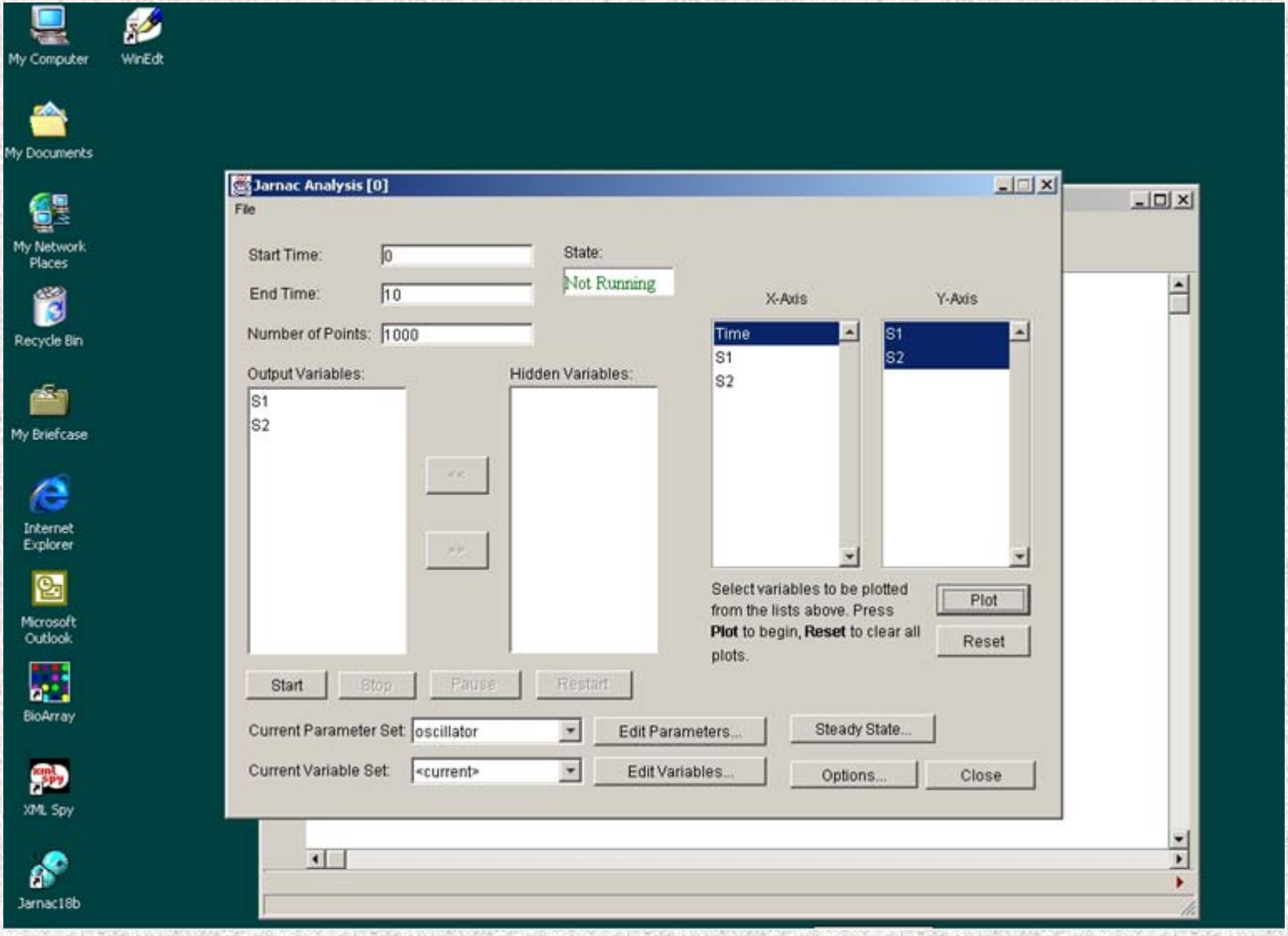
Systems Biology Workbench (SBW)

- **Open-source, integrated software framework that enables sharing of computational resources**
  - Allows software developers to build inter-application communications facilities into their tools
- **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.

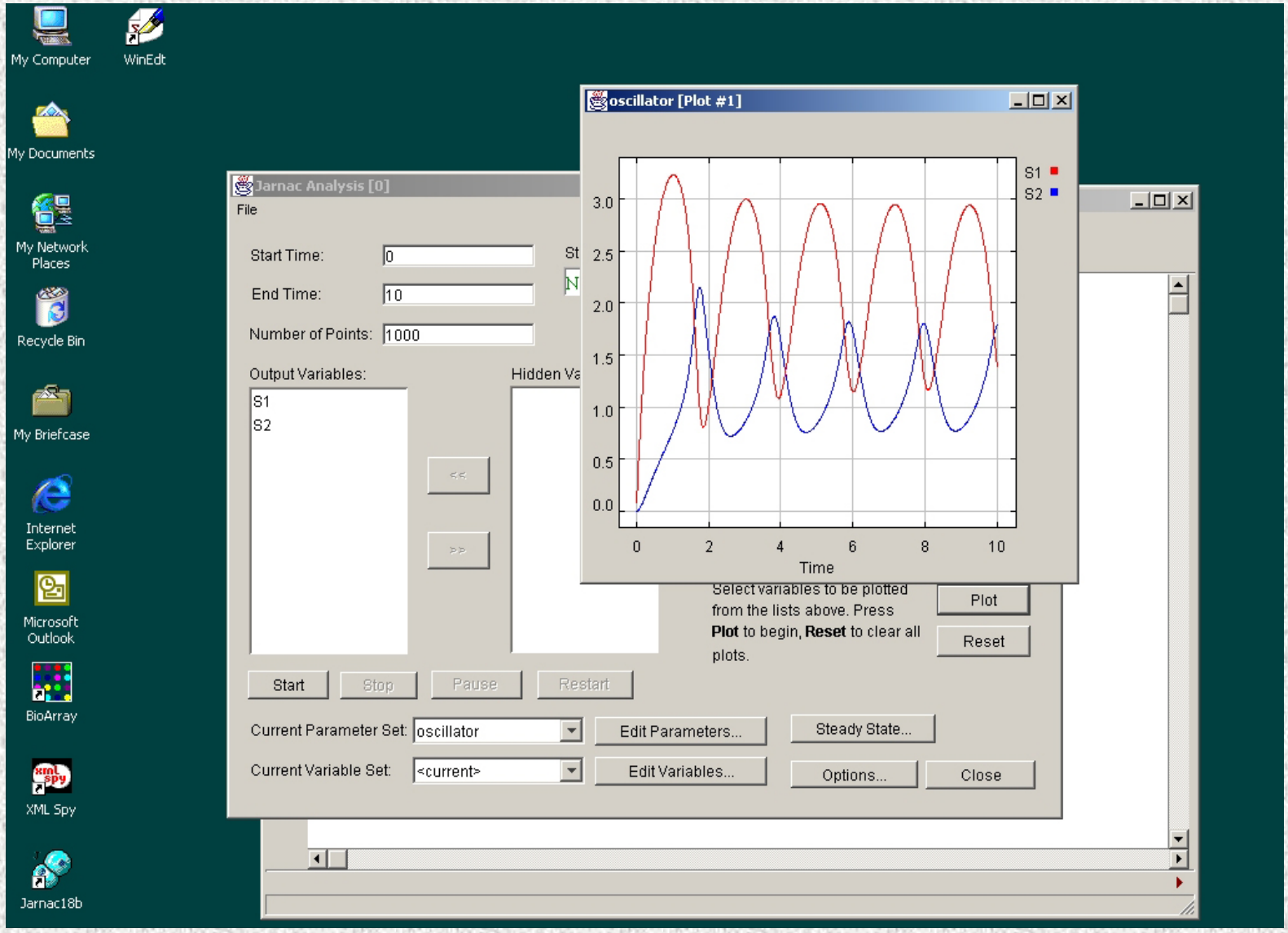
From the User's Perspective



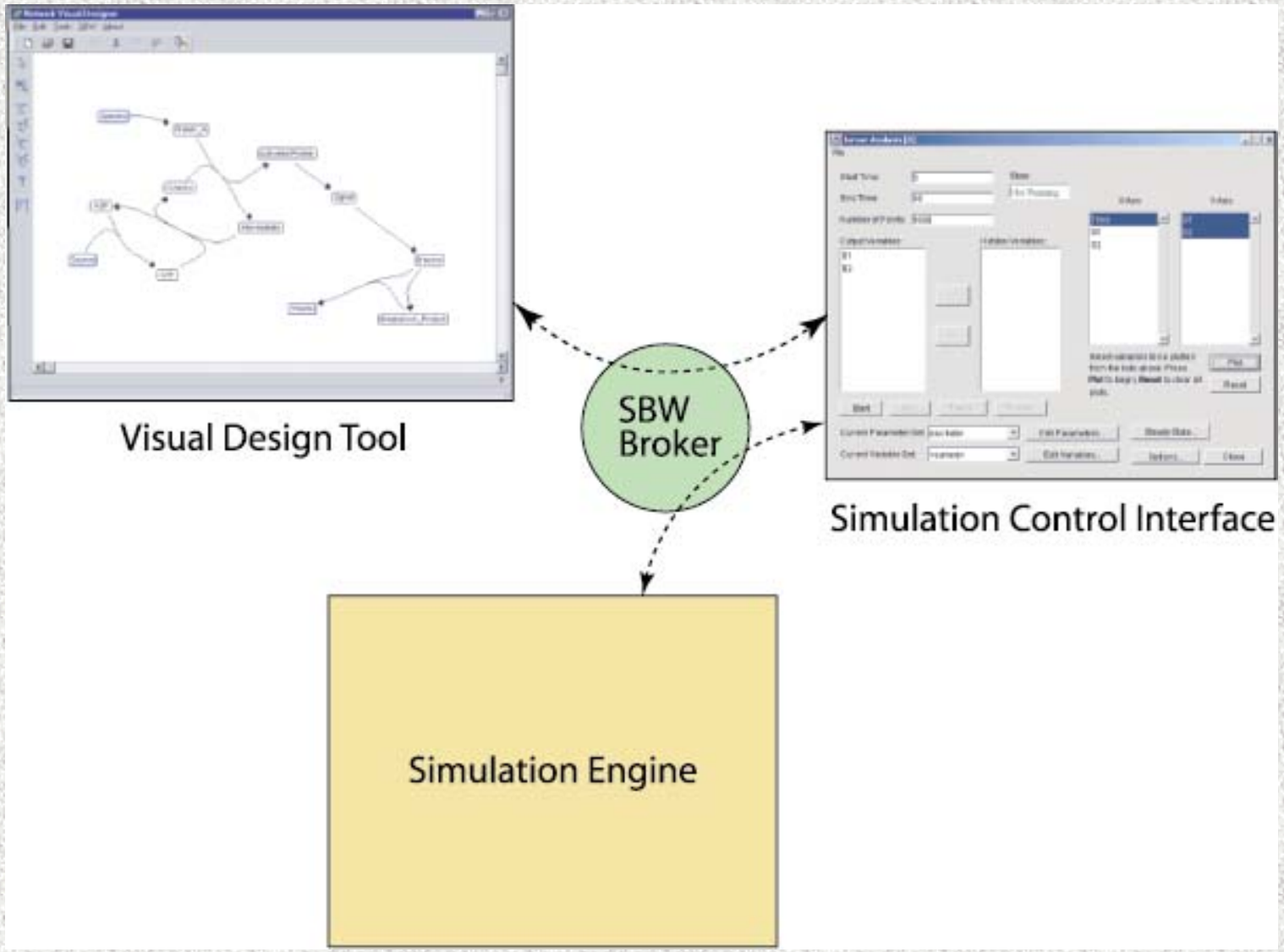
From the User's Perspective



From the User's Perspective



Behind the Scenes



From the Programmer's Perspective

- **Desirable features**
  - Small application programming interface (API)
  - Libraries that implement inter-program communications
  - A registry of services for applications to query
  - XML-based model representation (SBML)
- **Uses well-known, proven technologies**
  - Communications via message-passing over plain sockets
  - Modular, distributed, broker-based architecture
- **API provides two styles:**
  - "Low-level": call/send operations + directory services
  - "High-level": object-oriented interface layered on top

Driving Principles

- **Keep it language-neutral**
  - We'll provide C++, Java, Delphi libraries for Win, Linux
    - But can be implemented in any language
- **Keep it simple**
  - Simple message-passing scheme
    - Avoid complexity & size of CORBA
    - Easy to make cross-platform compatible
    - Easy to make distributed
  - Simple low-level API, convenient higher-level API
- **Make sure contributors benefit**
  - Open source development
  - Symmetric infrastructure: no application dominates

Modules Planned

- **Data filtering & preparation**
- **Database support**
  - E.g.: web searching, storage management
- **Model definition & manipulation**
  - E.g.: scripting languages, visual editors
- **Equation solvers**
  - E.g.: ODEs, DAEs, stochastic
- **Analysis & visualization tools**
  - E.g.: bifurcation, 2-D/3-D/4-D plotting
- **Optimization & parameter searching**

Summary & Availability

- **SBML**
  - Level 1 specification is publicly available
    - http://www.cds.caltech.edu/erato
  - Support being added by other groups to their apps
- **SBW**
  - Preliminary test implementation completed
  - Production version is now in development
    - Draft API definition & other info available
      - http://www.cds.caltech.edu/erato
    - Expect first public release in November at ICSB 2001 (http://www.icsb2001.org)