

SimBionic[®] for Simulation-Based Training

Automated Student Evaluation, After-Action Review, and Role-Players

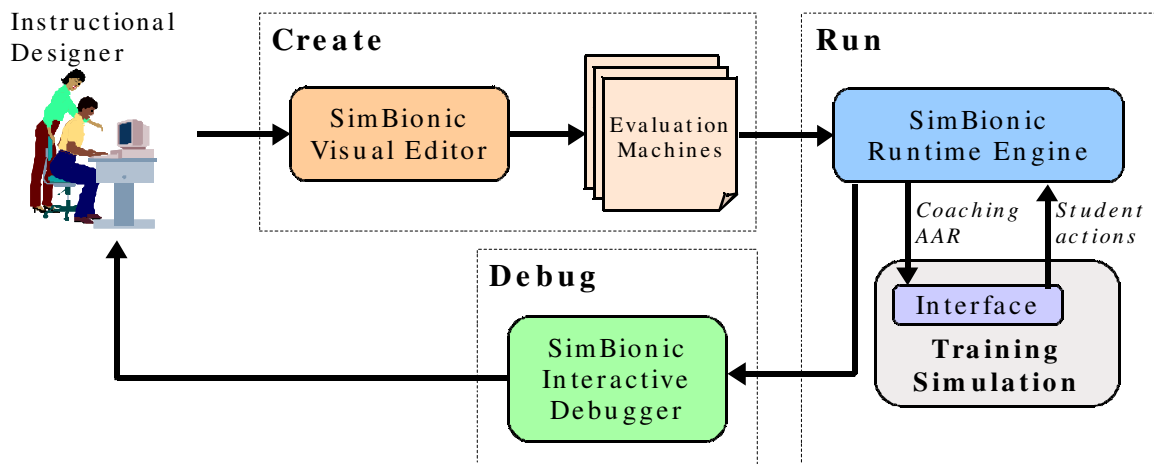
SimBionic[®] is an award-winning AI toolset designed to help you build better training simulations. SimBionic is a key part of Stottler Henke's intelligent simulation and training technology platform and has been employed to provide coaching, automated after-action review (AAR), and automated role-players in a wide variety of training simulations. SimBionic has been used in training systems for:

- Navy Tactical Action Officers
- Army battalion commanders
- Unmanned vehicle operators
- Helicopter pilots
- NASA payload operations personnel
- Emergency response medical personnel

Automated Student Evaluation

SimBionic enables you to build automated student evaluation capability into your training simulation so that it can monitor student actions and evaluate their performance to support hints and feedback during exercises as well as after-action review.

With SimBionic, you can easily create *evaluation machines*, which are flowchart-like bits of logic that observe the student and determine when he or she has correctly or incorrectly exercised a particular skill being trained. SimBionic's visual editor enables instructional designers and subject matter experts to create and understand these evaluation machines without any programming expertise. The SimBionic runtime engine then executes these evaluation machines while the student runs your simulation, automatically detecting correct or incorrect student performance and notifying your training system so that it can take appropriate action.



Automated Role-Players

For many types of training, many instructors or observer/controllers are required to play the various roles in a training scenario. The resulting cost and scheduling difficulties often limit the availability of training for students. With SimBionic, you can replace some or all of the human role-players in your training simulation with automated equivalents. SimBionic is specifically designed to simplify the process of capturing complex decision-making and behavior logic so that it can mimic both teammates and adversaries.

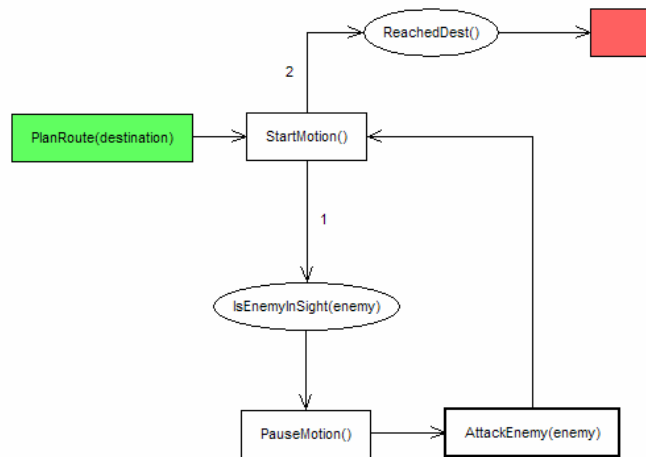
Visual Authoring

Traditional development methods require logic to be coded in software by programmers, so it is difficult for instructional designers to directly specify the doctrinal logic for evaluating a student's actions. Instead, the designers must describe the desired logic to computer programmers who then translate these descriptions into software. This multi-step approach is cumbersome, time-consuming, and error-prone.

By contrast, the SimBionic editor presents logic graphically, so users can create, edit, and review them easily, without programming. Graphical representations can be understood by experts and by software programmers alike, so they can speak the same language, enabling more effective and collaborative training development. The end result is faster, cheaper, and more maintainable training systems.

Powerful Reasoning Capabilities

In SimBionic, you implement evaluation and decision-making logic as Behavior Transition Networks composed of actions and conditions. In the example behavior below, rectangles represent actions, arrows show transitions from one action to the next, and ovals represent decision conditions that must be true before a transition can occur.



Behavior Transition Networks extend the basic notion of finite state machines by allowing behaviors to invoke other behaviors hierarchically, resulting in modular behaviors that can be powerfully combined. SimBionic also provides a number of extensions that increase the power and expressiveness of the basic engine, including:

- A unique stack-based condition-checking scheme enabling higher-priority behaviors to override lower-priority behaviors.
- Global and local variables.
- Interrupt transitions for temporarily interrupting the current behavior to perform some higher-priority task.
- Blackboards and message-passing for sharing knowledge among simulation entities.
- Polymorphic indexing for runtime selection of behaviors based on entity state.
- In-behavior exception handling.
- Ability to directly access Java classes and methods from within behaviors (Java engine only).

Efficient Runtime Engine

The runtime engine, available in both C++ and Java versions, provides a flexible application programming interface (API), so software developers can easily interface it with a wide range of training systems. The SimBionic engine is highly scalable: each behavior runs efficiently and occupies a small memory footprint, so SimBionic can execute many evaluation machines simultaneously.

Operating Systems

The SimBionic editor runs on Microsoft Windows 98/NT/2000/XP operating systems. The SimBionic runtime engine runs on these operating systems as well as Linux, and can be deployed in both Java applets and on a server.