GeneSim: a code generator for models of dynamic systems

Giovanni A. Cignoni, Simone Masoni
AICA Conference, Cesena September 2006

http://genesim.sourceforge.net
The project

- An open source project
  - Started as a personal project
  - Carried out with student participation

- No funds and low effort
  - Sourceforge (hosting and services)
  - Dept. Informatica, University of Pisa (facilities)

- To provide an environment to
  - Model complex systems (physical ones, but not only)
  - Generate simulators (continuous, but not only)
  - Integrate simulators in custom applications
  - Directly run simulations (for test or analysis, for instance)

http://genesim.sourceforge.net
The environment

- SysML specification model
  - Defined rules to specify dynamic systems using SysML
  - XML schema for system specifications

- Code generator
  - Reads the XML and generates a library of classes
  - Library provides support for configuration and data logging
  - Library can be compiled or integrated in custom code

- Test tool
  - Simulators compiled as dynamic link libraries (.dll, .so, ...)
  - Common library interface for custom system interfaces
  - The test tool generates system interface, runs simulations

http://genesim.sourceforge.net
Steps of the code generation process:

- Syntax check (Xerces, DOM parser)
- Validation, errors & warnings about system composition
- Analysis, dependences and cycles
- Transformation in a language independent structure
- Code generation (Xalan, XSLT processor)

http://genesim.sourceforge.net
Example: a simple system

- A simple system specified as a SysML block (a class)
  - A state that comprehends an interface
  - A set of functions to update the state
  - Support for constraints like unit of measure and range

```
Resistence
{uom = "ohm"}

Compute resistance
{return Ro, coded,
C++ = "{
    if (St >= 0)
        return R;
    else
        return -1;
}"

«block»
Resistor

+ R: double = 10.0 {const}
+ Vmin: double = 10.5 {const}
+ Vmax: double = 14.0 {const}
+ compute_St (St, Vi, Vmin, Vmax)
+ compute_Ro (St (inst), R)

Ro: double = 10.0
Vi: double = 0.0
Voltage
{uom = "volt"}

St: int = 0
Resistor status
{min = -1, max = 1}
```
Example: an assembly

- An assembly specified by composition of parts
  - Parts are instances of blocks
  - Parts interact by way of connections
  - Block interface is defined by delegation of parts interface
• A robot electrical circuit
  • 5 main systems
  • 6 lead acid batteries

http://genesim.sourceforge.net
Future work

- Improvements and new functionalities
  - Model: structured data types and ports function definitions (MathML, SysML parametrics)
  - Tools: drawing tool porting to other platforms integrated IDE
  - Code: improve realtime support optimizations for discrete simulation support to other languages

- Test on more complex real case studies

- Try to find sponsors