Atego Artisan Studio integrations with Simulink

- Artisan Studio with parametrics and Simulink synchronizer
- Artisan Studio with parametrics, and state behavior, and code generation using Simulink synchronizer, Simulink code gen to C/C++ lib
- Artisan Studio using SySim use Simulink during model execution
- Artisan Studio with Atego Parasolver uses Simulink as a solution engine for parametrics, and returns values to Artisan
- Artisan Studio state behavior, Internal block definition and ports using ACE to tie into Simulink blocks and ports
Atego Solutions

- **Atego Process Director**: Process Definition, Deployment & Management
- **Atego Exerpt**: Requirements Interchange
- **Artisan Studio**: Code Generation
  - **Artisan Studio SySim**: Artisan Studio Add-On
- **Atego Ace**: Co-Simulation
- **Atego Workbench**: Tool-Chain Collaboration & Hosting
- **Atego Global Services**: Professional Services
- **aonixPerc**: Java VM IDE
  - Ultra
  - Pico
  - UltraSMP
  - Raven
- **blueriver X32plus**: C / C++ IDE
- **aonixObjectAda**: Ada IDE / Runtime
  - Real-Time
  - Raven
- **Atego High Rely**: Certification Solutions
  - HighRely Check
  - HighRely Trace
Artisan Studio Architecture Standalone

Standalone Installation

Artisan Studio

Add-ins

Artisan Models Neighborhood

maps to repositories

ten controls repositories

Repository

Enabler Administration

Enabler Repository Server

stores repositories

each repository contains models

Model
Artisan Studio Architecture Client/Server

Client Only Installation

Artisan Studio

Artisan Models Neighborhood

Add-ins

Enabler Administration

Server Only Installation

Enabler Repository Server

stores repositories

controls repositories

Enabler Administration

Repository

controls repositories

each repository contains models

Artisan Models Neighborhood

initializes repositories

Model

© 2012 Atego. All Rights Reserved.
DOORS Synchronizer

- Multiple modules per Studio model
  - Able to bring requirements from multiple modules into a single Studio model
  - Create traceability in Studio
  - Export back to DOORS (including surrogates)
Repository-based CM – Using Private Sandboxes
Repository-based CM – Using Private Sandboxes

- Artisan Studio includes a 3-way conflict resolution tool:
  - Including a diagram differencer for resolving conflicts
Modeling
SysML (System Modeling Language)

- Same as UML
- Modified from UML
- New for SysML

Diagram

Structure
- Block Definition
- Internal Block
- Parametric

Requirements
- Package
- Activity
- State Machine
- Sequence
- Use Case

Behavior
The Four Pillars of SysML

1. Structure

```
ibd [Block] Anti-Lock Controller1

<use>

d1 : Traction Detector

m1 : Brake Modulator

<use>

c1:modulator interface
```

2. Behavior

```
<interaction>

<use>

act PreventLockup

<stateMachine>

Detect Loss Of Traction

TractionLoss

Modulate Braking Force

<activity/>

<function>

Detect Loss Of Traction

TractionLoss

Modulate Braking Force

<action/>

PreventLockup

<use>

<parametricDiagram>

StraightLineVehicleDynamics

{\[ f = (tftb)*(1-tl) \]}

{\[ F = ma \]}

```

3. Requirements

```
<requirement>

id# 102

txt The vehicle shall stop from 60 mph within 150ft on a clean dry surface.

<requirement>

id# 337

txt The Braking subsystem shall prevent wheel lockup under all braking conditions.
```

4. Parametrics

```
<parametricDiagram>

StraightLineVehicleDynamics

{\[ f = (tftb)*(1-tl) \]}

{\[ F = ma \]}

```

© 2011 Atego. All Rights Reserved.
Data Mining/Data Analysis (ICD Generator)

Generate traditional Systems documentation from modeling information
Model Execution and Code Generation
Why use modeling tools?

- An altogether different need is to support the automatic generation of “things” from the model

ACS/TKD offers combination of:

- Speed
- Flexibility (to define the mapping)

ACS/TKD offers combination of:

- Speed
- Flexibility (to define the mapping)
Code Generation Customization (Transformation Model)

**Rules**

- **4GGenerator::4GStructure::Class**
  - Name : %string
  - Visibility : %string

- **4GGenerator::4GStructure::Attribute**
  - Data Type : %string
  - Name : %string
  - Default Value : %string
  - Access : %string

- **4GGenerator::4GStructure::Operation**
  - Name : %string
  - Short Return Type : %string
  - Access : %string
  - Text : %string

**QClass**

```
Gen -> "Class" _ Name _"{" CR +1 dConstructorGen CR dDestructorGen CR ClassBody -1 "};" CR CR;
ClassBody -> {Attribute.Gen} {Operation.Gen};
implGen -> "//TESTING ONLY";
dConstructorGen -> "void" _ Name "(){" CR +1 dConstBodyGen -1 CR "};
D DestructorGen -> "void ~" Name "(){" CR +1 dDestBodyGen -1 CR "};
dConstBodyGen -> "stuff iam here;";
dDestBodyGen -> "// don't allow destruction" CR +1 "hi";
```

**QInterface**

```
Gen -> "Interface" _ Name _"{" CR +1 dConstructorGen CR dDestructorGen CR ClassBody -1 "};" CR CR;
```

**QAttribute**

```
Gen -> Access":" _ Data Type _ Name [_"=" _ Default Value"] ;" CR;
```

**QFakeOperation**

```
Gen -> Short Return Type _ Name "(" [Parameter.Gen {"," Parameter.Gen}] "){" CR;
```

When (in O : %object) : %numeric
Parametric Diagram

- **Rationale**
  - $v(n+1) = v(n) + \Delta v = v(n) + a\Delta t$
  - $\{v(n+1) = v(n) + a\cdot30\cdot3600/5280\cdot\Delta t\}$

- **Rationale**
  - $tp(hp) = wheel\ power - drag - friction$
  - $\{tp = whipower(Cd^2v) - (Cf^2v^2)\}$

- **Rationale**
  - $x(n+1) = x(n) + \Delta x = x(n) + v\Delta t$
  - $\{x(n+1) = x(n) + v\cdot5280/3600\cdot\Delta t\}$

**Equations**

- $a(g) = F/m = (P/t)/m$
- $(a = (550/32)\cdot tp(hp)\cdot \Delta t/\Delta t)$

**Variables**

- $v$: velocity
- $a$: acceleration
- $tp$: power
- $x$: position

**Constraints**

- $Cf$, $Cd$, $whipwr$
Add Simulink profile to your model
Matlab Syncronizer

- Allows mapping of types
- Creates parametric block to Matlab block mapping
Matlab sample

![Matlab sample image with diagrams]
Early Validation of Complex Behavior

- Use your existing SysML designs for live model execution
- Enable incremental design, starting with basic behavioral features, adding more capabilities over time
- Enforce consistency of system models with ongoing, dynamic checks

Block Reuse

- Easily reuse your existing subsystems and combine them into different executable scenarios
- Use instance parameters for differentiating the behavior of reused blocks
Executable Models: Modeling Use Cases

- Model the executable system
  - Set up simulation scenarios using IBDs
- Model system behavior
  - Use predefined control blocks for input and output
- Use predefined action language (ASAL)
- Use Atego Structured Action Language (ASAL)
- Use Matlab/Simulink
- Use VB.NET

Tools:
- MS Visual Studio™
- Atego SySim
- MATLAB/Simulink™
Executable Models: Simulation Use Cases

- Systems Engineer
- create an executable version of the model
- simulate the System
- define Simulator Form using Drop & Play
- co-simulate with Simulink
- Single-stepping
- Animation State Machines
- Simulation log in MS Excel
- interact with Simulation I/O
- MS Visual Studio™
- MATLAB/Simulink™
- MS Excel™
Executable Model Example: Speed Controller
What’s Needed for System Validation Using Constraints?

**Typical SE Tasks**
- Simulation
- Analysis
- Trade Studies
- Optimization
- Requirements Checking
- Risk Assessment
- Etc.

**Features needed**
- Solvable or executable language for constraints
- SysML instance model to express design alternatives in slot values
- Solver technology, which can handle the acausal parametrics
- Input/Output to common systems engineering tools
Artisan Studio ParaSolver Parts

- ParaSolver Profile
- User’s Guide
- Tutorial
- Example Models
In Artisan Studio

- Importing the ParaSolver Profile causes the SysML Profile and the UML Profile to be included as well.

- The Tools Menu shows the ParaSolver functions.
A Simple Example

- bdd
- par
- od
Don’t Forget...

- SysML Parametrics are acausal, so you can calculate
  
  \[ c = a + b \] with given \( a \) and \( b \)  
  or

  \[ a = c - b \] with given \( c \) and \( b \)  
  or...

- WITHOUT changing anything in the model

- Of course this is a VERY simple example, but this works also for more complex ones, of course

- The Instance model holds all calculated values, enabling the model to contain more than one alternative
SysML Parametric Model Execution: Orchestrating External Solvers & Trade Studies

Simple math relation

MATLAB/Simulink model

Excel

Mathematica function

Key System MoE

+ acausal modeling
Current and Planned Integrations

- Mathematica
- OpenModelica
- Matlab
- Matlab/Simulink
- Excel

Planned:

CAD/CAE Systems, e.g. Catia
Demos

- Satellite Power and Weight Analysis
- Little Eye UAV – Area Covered Analysis
Artisan Studio Benefits

- **SysML benefits:**
  - Requirements documentation and traceability to view the system for completeness and adherence to the customer needs.
  - Express and document your design decisions.

- **Parametrics benefits:**
  - Build a mathematical model of the solution to analyze and to verify requirements against.

- **ParaSolver benefits:**
  - Enables the systems engineer to run instances of their design decisions against the parametrics models created.
  - Can run multiple scenarios to figure out the right solution.