AnyDynamics 8

AnyDynamics is the Rand Model Designer 8 Standard Free. It differs from RMD 8 Professional only in that it does not allow the creation of models embedded in the application.

Why «any dynamics»?

Because it is a versatile object-oriented modeling tool that allows you to create computer models of dynamic systems of all kinds. Other versatile modeling tools have significant limitations:

- tools based on the Modelica language have serious limitations on the models of hybrid systems and systems with variable structure (the total number of unknown variables and solved equations should not change when switching);
- complex "MathLab + Simulink + StateFlow" is limited by the set of available libraries of typical elements;
- “AnyLogic” tool has serious limitations on continuous systems models (component models with undirected links are not supported).

Differences from Rand Model Designer 7 Standard

There are three groups of differences::

1) new functionality;
2) modeling language extension;
3) new project storage format.

New functionality

The user has the ability:

- use step-by-step debugging of sequences of statements in a visual model, similar to visual programming tools;
- use symbolic differentiation;
- see the Jacobi matrix and the matrix of eigenvalues of the full system of equations;
- save the current state of the model to a file, load the state of the model from the file and compare the current state of the model with the previously saved one;
- use the timing of the executed model to analyze the structure of time costs.

Modeling language extension

The user got the opportunity to use when writing functions and sequences of operators in discrete actions, not only the Ada language syntax traditional for modeling systems, but also familiar for a significant number of users, the C # language syntax. The choice of syntax is made using the switch on the toolbar.

Extended features of the input language:

1. declaration of a local ("working") variable in the statement block;
2. an empty list of parameters for the function is allowed (for example, “pi ()”);
3. or Ada syntax, the index countdown mode in the loop operator is added using keyword "reverse" (all identifiers "reverse" in convertible models are automatically replaced by "reversal").
4. the use of the same names of enumerated literals in declarations of different enumerated types is allowed (for example, “type Color is (green, yellow, blue)”, “type Lights light is (red, yellow, green)”). To overcome the ambiguities, you can use the literal name with the type name (for example, “Color.Yellow”, “Lights.Yellow”).

Due to the fact that the vertical bar "|" in C # syntax means the bitwise "or" operation; in an iterative literal, the colon ":" is used instead. When saving a model in a version 7 compatible file "* .mvl", a vertical bar is used.

New project storage format

In version 8, instead of the object-oriented database “mvBase”, the project is used to store the project object-oriented database "mvBaseX" with textual XML representation instead of binary. Accordingly, the project files have the extension “* .mvbx” instead of “* .mvb”.

When you open the “* .mvb” project in version 8, a dialog appears with a proposal to convert it into a view "* .mvbx" (the old project is saved). In some cases, re-translation of the project through text is also necessary.

By default, the text MVL-representation of the project is saved in a file with the extension "* .mvlx", and the current syntax selected by the user (Ada or C #) is used. If the user chooses to save a file with the "* .mvl" extension, then the syntax of the Ada language is used when generating the MVL view. Thus, if the models do not use extensions of the input language, then through the “* .mvl” file, backward compatibility with version 7 is implemented.