Simplified Modeling of Dynamic Systems with Simulink

By Michael Carone, MathWorks

R2012b, released in September 2012, features major updates to Simulink® and Stateflow™—notably, the new Simulink Editor and Stateflow Editor. These updates represent the biggest user experience upgrade to Simulink and Stateflow since they were introduced in 1990 and 1997, respectively.

In this article, Jay Torgerson, senior development team lead, explains some of the thinking behind these updates. We then present an example showing how Simulink R2012b simplifies model navigation, simulation analysis, and model editing.

Michael: Jay, what would you say is the theme of this release?
Jay: The overarching theme is the unification of the two different environments, Simulink and Stateflow. But there are several subthemes. We wanted to address usability in three key areas: navigating large models, creating diagrams, and debugging.

Michael: What impact do you think this release will have on users?
Jay: They will be able to create and edit diagrams much more easily. Users can view and navigate through both the Simulink and Stateflow hierarchies within the Model Browser. And we introduced tabs and windows reuse, to reduce window pollution.

Michael: Which feature are you and your team most proud of?
Jay: I think the feature we’re most proud of is the quality of the editors. We were able to rewrite the Simulink and Stateflow editors and create a new foundation while preserving existing modeling workflows. Early on, we decided not to change the API. The scripting APIs to Simulink R2012b are exactly the same as they were before. As a result, we were able to reuse 170,000 test points written for the previous editors to qualify the new editors. That was pretty significant.

Michael: What was the most significant improvement made to Simulink in R2012b?
Jay: Probably the smart routing of lines. As you’re connecting blocks in a diagram, Simulink automatically creates lines that avoid blocks, labels, and even annotations. Users were reporting that every time they moved a block they had to spend an hour tidying up the model; the "twiddle factor," as one user called it, was very high. We tried to attack that at a very low level for the router, making the router smarter at all times, and always automatically routing signal lines unless you specifically use the manual, user-prompted mode.

Michael: How about Stateflow?
Jay: Probably the most significant improvement to Stateflow is that it now renders in the same canvas as Simulink. You can graphically interleave Simulink and Stateflow layers in the same view, tightly integrating the products and reducing window pollution. We also fixed some long-standing design problems. For example, previously you could not select and move an entire superstate and all its contents with one click. To solve that problem we devised the concept of drag margins. With drag margins, you can move a super state around simply by selecting its border. Beyond that, we upgraded the look and feel of the new Stateflow canvas to enhance the readability of complex logic expressed in Stateflow charts. This logic is very often used for safety-critical systems, so clarity and readability are extremely important.

Michael: How do you see this release fitting into an overall vision for the direction of Simulink?
Jay: Simulink is an incredibly expressive multi-domain platform, and it’s being used to make some of the most advanced applications the world has ever seen. Supported domains range from signal-based block diagrams, to state machines, to discrete-event systems, to physical modeling systems . . . you get the idea. The systems that Simulink users build are becoming ever larger and more complex. We wanted to simplify the user experience when working in the presence of these large multi-domain models. Simultaneously, we wanted to
lay a strong graphical language editor foundation that would enable us to support even more modeling domains. The new editors in R2012b do exactly that.

**Model Navigation and Debugging Made Easy—An Example**

Let’s suppose that we’re collaborating with a colleague on a project. The colleague sends us an archive of their Simulink Project saying that the simulation is producing unexpected results. Our task is to understand why this happening and fix the problem.

We begin by extracting the project from the archive and opening the main model. We simulate the model and view the results. The Scope plot immediately shows us that the output signal is exhibiting unstable oscillatory behavior (Figure 1).

![Scope plot showing oscillatory behavior](image1)

*Figure 1. Simulation results showing oscillatory behavior.*

To investigate the simulation further, we turn on the Simulation Stepper and specify that the simulation should pause at time $T = 20$, the point at which the unstable behavior occurs.

We rerun the simulation, which pauses at $T = 20$. To understand where the oscillation is occurring, we open the Stateflow chart in the same window. We press the Step Forward and Step Back buttons in the Simulation Stepper, view the animation of states, and notice that an oscillation is occurring within the pressure state.

![Stateflow chart showing pressure state](image2)

As we hover over the state, we notice that the value for pressure is very low.
Going back to the top level of the model, we click the input signal for pressure and select **Show Value Label of Selected Port** to monitor the value of that input signal. Stepping forward and backward again, we notice that pressure is low throughout the simulation.

![Dиagram](image1.png)

The next step is to trace the signal’s origin. We right-click the signal and select **Highlight Signal to Source**. The source block opens in a separate tab.

The connection from the Source MATLAB Function block to the output port is correct, but for some reason, the pressure signal has been connected to the oxygen signal. Tracing the signal and moving up the model hierarchy via the Explorer Bar, we notice that the signal has been connected incorrectly at the top level of the model.

The solution to this problem is to reconnect the signals properly, which we can easily do using smart signal routing.

![Dиagram](image2.png)

We rerun the simulation (Figure 2).
After confirming that the problem has been fixed, we archive the project and send it to our colleague, who can then ensure that the team will be working with the latest version.

**Figure 2. Simulation results showing correct behavior.**