Migrating from Modelica 2 to Modelica 3

Many users that started using Modelica several years ago have migrated from earlier versions of Modelica (and corresponding tools) to Modelica 3, or have at least considered migrating.

There are several reasons for upgrading to a recent version of the Modelica language and its libraries, but the effort is not trivial and requires some planning.

The suggested process gives complete control of each step and the possibility to verify that models produce the same simulation results as before the migration.

Reasons for upgrading to Modelica 3

There are good reasons for upgrading from Modelica 2 to Modelica 3. The Modelica language is in constant development, and new capabilities continue to be added to both the language and the libraries. The changes that created Modelica 3 were focused on improving model quality, delivering better diagnostics and making it easier to produce correct models.

The two key improvements in Modelica 3 are:

► Models at every level must now be "balanced", meaning that they must have the same number of unknowns and equations. This enables local checking and more precise error messages to be generated for incorrect components.

► Rules for model interfaces (type system, replace-able) have been redesigned and more precisely defined. The result of which is that is that a redecla-ration can no longer introduce a language error.

The Modelica Standard Library (MSL) was also been greatly enhanced, growing from 740 to 1300 models and blocks.

Balanced models

The main reason for introducing balanced models (at every level, not only for the top-level model) is to make it easier to write verifiably correct models. It also gives the tool a much better chance to deliver more meaningful diagnostics. Whereas with Modelica 2, Dymola could give you a message like:

```
RobotR3.fullRobot is structurally singular,
It has 1459 scalar unknowns and 1457 scalar equations.
```

in Modelica 3 the message would be:

```
RobotR3.fullRobot uses components that are not correct:
RobotR3.Components.GearType1 has 3 missing equations.
RobotR3.Components.Controller has 1 equation too many.
```

The notion of a "balanced model" has several aspects. One of the most significant is the restriction on connectors that the number of flow variables in a connector must be equal to the number of non-flow variables that do not have input, output, parameter prefix. For example, this is ok:

```
connector Pin
   Real u;
   flow Real i;
end Pin;
```

but the following is not:

```
connector Flange
   Real      angle;
   Real      speed;
   flow Real torque;
end Pin;
```

Every model must be "locally balanced" which imposes a restriction on the number of equations: i.e. the number of equations must equal the number of unknowns minus the number of flow variables.

Reasons for not upgrading

It is hard to find good reasons for not upgrading to Modelica 3, but of course such an effort must be scheduled to fit with overall workload, project dead-lines, etc. Changing library or tool versions may reveal small changes in the simulation results, even if the migration is correct. Often the Modelica language migration project is a good reason to review models developed in-house. This often leads to changes in structure or simulation results that may be bigger than strictly necessary, thus further complicating the verification process.

Migration best practices

The migration takes place in several steps. The process described here gives you control of each step, and allows you to verify simulation results to avoid hidden or unwanted changes. An important intermediate step is a version of your models that will run with both an old Dymola and a new one, giving the same simulation results.

Use version control. If you haven't done so before, this is an excellent opportunity to deploy a version control system for your Modelica models. This is useful even if you are the only developer, because it makes it easy to track all changes and check for differences if any issue arises; make sure you "commit" code often. If you branch off a separate development path, the migration can be done with minimum interruption of normal work.

Plan your references. Later in the process you must verify simulations of the migrated models against results created by the original baseline versions. It is probably unrealistic to check every meaningful combination of parameters and models, so you must make an intelligent choice of good test cases that you believe will test both the basic function of models and the tricky combinations. Plan for a quick minimal test and an extensive (and slower) test sequence.

Verify the simulation results to avoid hidden or unwanted changes.

Check your model for errors. Even before upgrading from Dymola 6 to a later version, make sure your models are error free. The "check" command is your best friend.

Check your model for warnings. Error-free is not good enough. Many problems reported as warnings in Dymola 6 will become errors in later versions of Dymola which are based on Modelica 3.

When these steps are completed, we have the best-possible model given our existing tools. The next step is to take advantage of a recent version of Dymola.

Install Dymola 2013 with compatibility libraries. Select MSL 2.2.2 in Edit>Options to get a Modelica 2 environment.

Run check in Dymola 2013. The new version of Dymola may report additional issues that need fixing. Note that because you are still working in Modelica 2, your models will be compatible between old and new Dymola.

Simulate in old and new Dymola. This will ensure that model behavior does not change in an uncontrolled way. When you simulate in Dymola 6, you can be certain that any differences compared to your reference are caused by model changes.

If you get slightly different simulation results in Dymola 6 and 2013, you need to check if your model is numerically sensitive, that initialization is correct, etc. Running a model with different tools and settings increases your confidence that the model uniquely specifies the problem and is numerically sound.

Switch to Modelica 3 and check. This step will probably highlight a number of things that need fixing. However, the full benefit of Modelica 3 is not achievable until the model has been completely migrated and checks without warnings.

It should be noted that Dymola, by default, allows certain constructs which strictly speaking are not legal in Modelica 3, but issues warnings instead of errors. This makes the transition easier, but at the cost of reducing error checking. For best result, switch to pedantic checking of the model later in the migration.

Use automatic conversion for libraries. Dymola provides excellent support for automatically converting models to use new versions of existing libraries. Opening a model developed for Modelica 2 will automatically convert all library model references to Modelica 3 as needed.

During a transition period your model may be used in both Modelica 2 and Modelica 3, but eventually the old environment should be dropped in order to take full advantage of Modelica 3.

Conclusion

Migrating your models from Modelica 2 to Modelica 3 is both feasible and well worth the effort. The suggested process may seem long, but it gives complete control and the possibility to check each step to verify that the models produce the same simulation results as before the migration.