Visual Studio Visualization and Modeling SDK Lab – Part 6 / 6

6 Conclusion

We have just seen a concrete example that served as our common theme: how to create a non-trivial DSL. We could go much further, particularly with more advanced aspects such as:

- **Enrichment of the UI**:
  - How to hide a property from the properties box (for example, the Label property from the Transitions).
  - How to create specialized property editors (for example, so that the names of events of transitions already used in the model appear in a drop-down list).
  - How to add interactive commands to the shortcut menu.

- **Enrichment of the *State Machine* model**
  - Take into account sequential transitions.
  - Addition of the concept of sub-states. (This will introduce, in particular, a modification to the custom code of the Diagram so that sub-states can be viewed in the form of interlinked states.)
  - Generating the code for all of these aspects: What is especially beneficial at this point is that the code generation that we have carried out does not have to be modified, because we can carry out model **transformations to** find the single cases that we have dealt with. These model transformations can even be carried out "in place" in a transaction that will be canceled following code generation!

- **Enrichment of the interaction with Visual Studio**.
  - Advanced validation: It is possible to validate the code of the actions. Because it is contained in a Visual Studio project, our DSL can access everything that our solution is familiar with: the project, the references, the types, the documentation, etc. We will examine the different possibilities in further detail. We can, in particular, validate the conditions and actions.
  - Reverse engineering of state machines by using the structure provided by Drag & Drop, (by dragging and dropping a class or a file in the Designer).
  - Interactions with the debugger (in particular, we can insert break points in the diagram).

The possibilities in this are vast; all that's missing is our imagination.

Meanwhile, you have understood the concepts and the approach, and I have given you the keys that will let you learn more by yourself by using the online VMSDK documentation and the Visual Studio SDK. Here are some resources:

- Samples, updates, and more labs at the VMSDK Home
- Discussion and problem-solving at the VMSDK Forum
- Visualization and Modeling User Guide
- Jean-Marc Prieur's blog

It is now up to you to create your vertical or horizontal DSLs, and target your Framework tasks and techniques!
But in the meantime, if you want to do some training yourself, here are some bonus exercises.

Bonus

If you have a little time and would like to make sure that you have understood everything, you may want to complete the following exercises.

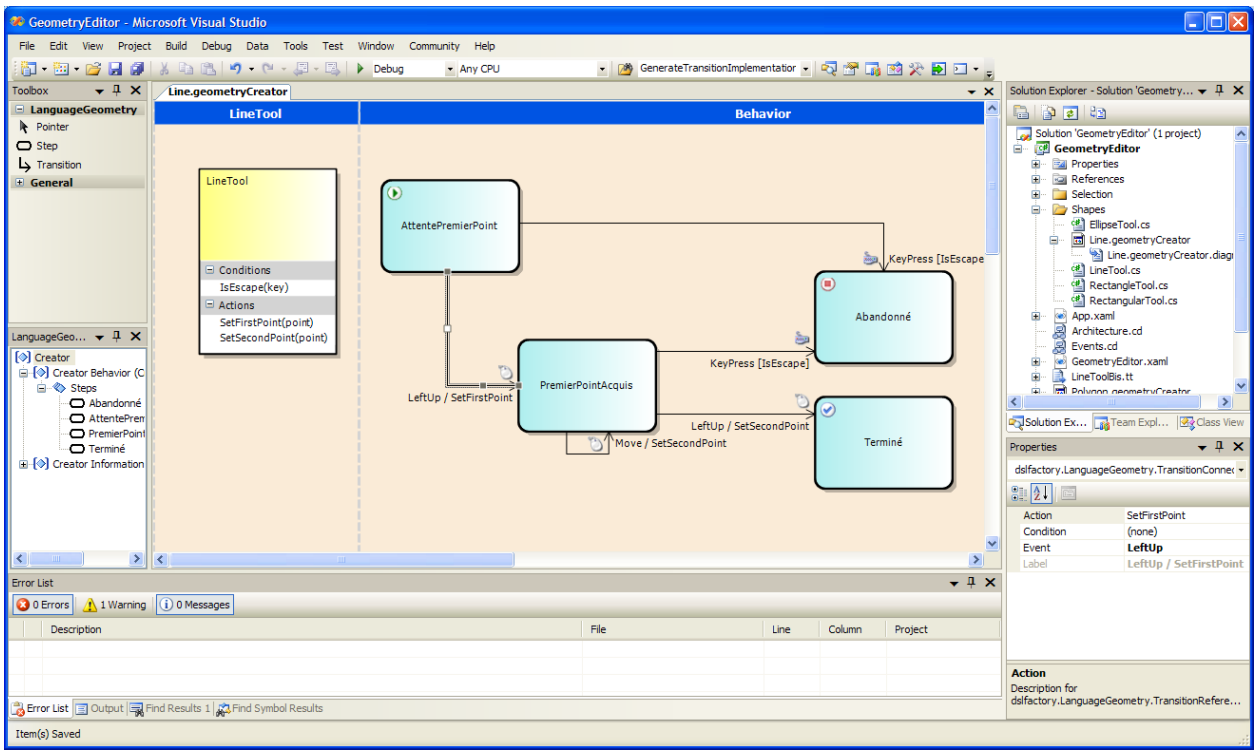
Statements

1. (Easy) Write a validation rule to check that the `StateMachine` has a non-blank Name property.

2. (More work) Change the domain model to use swim lanes in order to have:
   a. In the first lane, a compartment shape that has a compartment that contains the conditions, and another that contains the actions.
   b. In the second lane, the dynamic behavior.

This will let the user have a much better user experience because he will choose conditions and actions on the transitions, rather than inputting code that will be mixed with the generated code.

As an example, your DSL could now resemble this:
(Notice that the conditions and actions are now domain classes. This is easier for the user.)



3. (Difficult) Have the Condition and Action list filled in from your problem space, or through code re-abstraction.