

Comparison of Visual Studio's VMSDK and AToM3

Matthias De Cock

University of Antwerp, Computer Science Department

matthias.decock@student.ua.ac.be

Abstract

This paper starts off by briefly underlining the importance of model-driven engineering and then introduces two tools which can be used in this context: it compares Microsoft Visual Studio's Modeling and Visualization SDK (VMSDK, 2010) and AToM3 (2008). Using the example of a production system constructing APCs we explore these two tools both from a conceptual and a practical point of view. Special attention is given to the support for: modeling the system, validating the consistency and correctness of the system and simulating an actual instance of such production system.

Keywords: Modeling, Meta-modeling, Model-driven Engineering, Validation, Simulation, AToM3, VMSDK

1. Introduction

As software systems become increasingly complex the software development techniques used today may prove to be inadequate to tackle this increased complexity. There is a need for a new approach, which may be found in model-driven engineering. However for this to become a viable alternative to the traditional software development techniques good tools are required. While other papers such as Pérez-Medin (2007) have compared tools with specific goals in mind (in the case of Pérez-Medin (2007) Human Computer Interaction) the goal of this paper is to compare two of these tools in the most general way, the two tools are:

AToM3 A tool designed by the Modeling, Simulation and Design Lab of McGill University in Canada. Currently a new version of the tool, AToMPM is under development.

Visualization and Modeling SDK A software development kit that can be used in Microsoft's Visual Studio. This SDK was formerly known as DSL Tools in Visual Studio 2005 but has been renamed to VM SDK in the latest installment of the IDE.

To compare the two an example system is needed, in this paper we will use a production system assembling armored personnel carriers (APCs). The system consists of conveyor belts that carry components and connect different machines, some of which require an operator in order to function properly. As with most systems some constraint apply:

- A conveyor belt should not be connected to itself.
- A machine can have at most one output, with the only exception being a quality control center, which has one output for the working components and one for the broken components.
- A component generator has no incoming conveyor belts, it simply produces items which it deposits on the outgoing conveyor belt.
- Similarly a garage, storing the finished APC's, should have not any outgoing links as it is the end of the production line.
- Last but not least there is a set of cardinalities that need to be respected, for example a component should only be connected to a single conveyor belt. The same holds for operators and conveyors, an operator can't operate two machines at the same time.

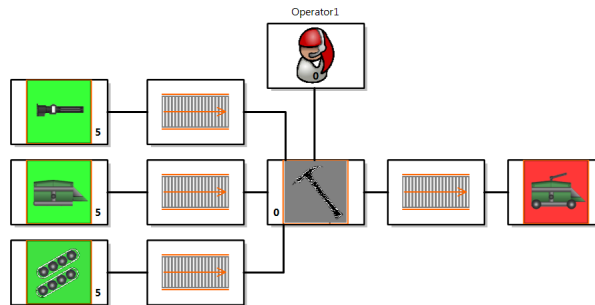


Figure 1: A very basic example of a valid system.

The remainder of this paper will mostly focus on how to model, validate and simulate this system in the 'Visualization and Modeling SDK', and how this compares to the same process in AToM3. In the next section we will discuss the process of modeling the production system, after which validation will be covered before proceeding to actual simulation of a constructed model in section 4. In section 5 a global comparison of the two tools will be made, finally leading to the conclusion in section 6.

2. (Meta-)Modeling

2.1. Class Diagram

In this phase one most often starts by declaring the different entities that make up the system and this time it's no different. Class by class we sculpt a DslDefinition (Domain-specific Language Definition) as it is called in the SDK. Each of these classes can be attributed with some domain properties. For example the component 'Body' has a property indicating the loadout of this APC, since an APC can have either no loadout, a 'Riot' loadout (water cannon and wheels) or an 'Army' loadout (machine gun and tracks instead of wheels). Also it inherited the property 'isBroken' from its superclass.

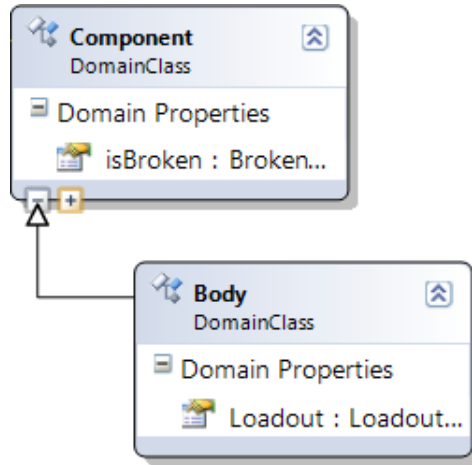


Figure 2: The domainClass 'Body', with properties 'Loadout' and 'isBroken'.

Components need to be able to be linked in order to create models. In the VM SDK this is achieved by connecting two DomainClasses with a referencing relationship. In this relationship each class performs a specific role. A rather

simple example is that when connecting two conveyor belts together one will act as the source and one acts as the target. As is expected from a modeling language each relationship can be bounded by cardinalities, let us for a moment consider the case when a machine and a conveyor belt are linked together. One constraint stated that a machine (apart from a quality control) should have at most one outgoing conveyor. So only once can the machine act as the source of a 'MachineIsConnectedToConveyorBelt' relationship, thus the cardinality should be set to '0..1'.

This is the first limitation of the Visualization and Modeling SDK, where AToM3 allows arbitrary cardinalities the VMSDK limits the options to:

- 0..*, an arbitrary number of times
- 0..1, zero or one times
- 1..1, exactly one time
- 1..*, at least one time

There is a way to enforce other cardinalities, using constraints as is described in the next section, but official support would be desirable.

2.2. Visual Language

The construction of a visual language is achieved by creating 'Shape' objects and linking them to specific domain classes. These shapes can be extended using decorators, either text-based decorators, simply showing some string, or icon decorators, displaying an image or icon. Based on the values of the domain properties you can either show or hide a specific decorator or change the value of a textual decorator (for example displaying the property value itself).

In general the construction of the visual language is pretty intuitive, although you need to know where to look. You can simply position a decorator by choosing a region of the shape where it should be displayed. This is both an advantage and a disadvantage of the tool, it is very easy to relocate a decorator to another general location (InnerTopLeft, LowerBottomRight, ...) but if you want to fine-tune the position of the decorator you can't just drag it to where you want to place it. You have to enter a manual offset (in inches) relative to the location you specified. Dragging and dropping is actually supported in AToM3, making it more suited for higher levels of customization.

Another limitation is that while you can change the font size and style of a text decorator the text color and the actual font are fixed, which might be a hindrance when displaying the text atop darker icons.

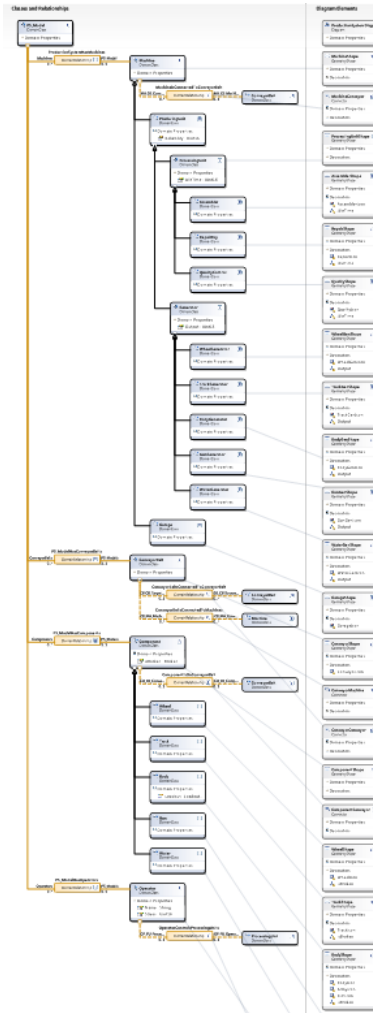


Figure 3: An overview of the entire diagram, with Classes and Relationships on the left-hand side and Visual Elements on the right. Notice the links between the classes and the visual elements.

Once the domain classes, the relationships and the visual objects have been created you can launch an instance of the meta-model, where you can drag and drop new Objects from the toolbox, connect them using the relationships you designed earlier and create a actual model of the system.

3. Validation

There are three types of validation supported in the Visualization and Modeling SDK:

Validation on request This type of validation allows for the model to be in an incorrect state, but when a specific condition is triggered (the model is opened/saved or the user presses the 'validation' button) the model is checked against these constraints and the user is notified of any inconsistencies and is free to interpret the results as he sees fit.

Interactive constraints The first type of constraint actually prohibiting the user from creating inconsistencies. In our example, should a user want to connect a component to a conveyor that is already carrying another component the cursor will change to reflect the fact that this is not possible. If the user still tries to connect the two no connection will occur, keeping the model consistent.

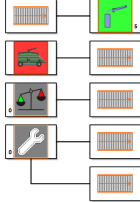
Hard constraints These constraints change the model in such a way that the internal consistency of the model is guaranteed. Should a value in the model change, rules may fire, changing other parts of the system to reflect the new value.

In the example of the the production system the first two types of validation were used. A model can be checked for consistency using validation on request and links between the models may or may not be allowed depending on the situation (using interactive constraints).

It should be noted that adding constraints to a meta-model is not very straight-forward in the VM SDK. In AToM3 actions and constraints form an important part of a class description (and actually show up in the Class Diagram), whereas in the Visualization and Modeling SDK the following steps need to be taken to enable validation on request for a specific class:

1. (optional) Add a new subfolder to the project called 'CustomCode'
2. (optional) In this folder create another subfolder 'Validation'
3. Add a partial class description for the class you wish to validate
4. To this class description add a method checking the specific constraint
5. Mark this method as part of the validation process

Once you are familiar with the approach it's not very complicated but the fact remains that this still feels like a workaround while it should be an integral part of the class itself. The same thing holds for the interactive constraints where you have to create custom connectionBuilders and tell the model that in the case of a connection between a machine and a conveyor belt it should use the custom 'MachineIsConnectedToConveyorBeltBuilder.cs' and not the automatically generated one.



	Description	File
✖ 3	A Quality Control station should have exactly two outgoing links	Validation.pr
✖ 4	A Garage should not have any outgoing links	Validation.pr
✖ 5	A Generator should not have any incoming links	Validation.pr
✖ 6	A Machine (with exception of the Quality Control) should have only 1 outgoing link	Validation.pr

Figure 4: Validation on request: The error messages refer to all four of the constrain validations present in this model.

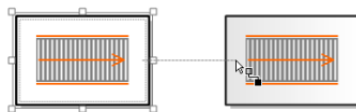


Figure 5: Interactive constraints: A legal connection between two conveyor belts, notice the cursor indicating the validity of the connection.

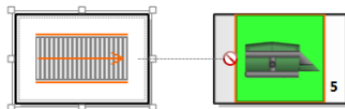


Figure 6: Interactive constraints: notice how the cursor changed to reflect the fact that this connection is illegal.

4. Simulation

In terms of the comparison between AToM3 and the VM SDK this is the major letdown, just like the name 'Visualization and modeling SDK' suggests, the focus is on the visualization and on the modeling of systems rather than the actual simulation of the system. There is no support for simulation whatsoever. Very small parts of the simulation process can be achieved. Using T4 text templates you are able to decide which machines are ready to produce, which operator has the highest stress value and so on. But this too feels like a workaround, boiling down to adding a T4 text template file, rightclicking it and then clicking on 'Run Custom Tool'.

```
=====
Analysis of the file MachinesCantFire.pr
=====

The following is a list of all the machines currently operated by operators
also indicating if they are able to function or if there is an obstruction.

The assembler operated by Amy is _NOT_ ready to produce,
it requires additional components to begin production.

The repair bay currently operated by Bruno is _NOT_ ready to repair the supplied component,
either there is no conveyor belt leading out of this machine or there is already another item on it.
```

Figure 7: Sample output of a T4 text template.

5. Global Comparison

Even though the last few sections may have convinced you otherwise the VM SDK certainly is a decent tool, this comparison was slightly biased in favor of AToM3 since we started from a project made in AToM3, and tried to replicate it in the VM SDK. Basically we didn't answer the question: 'Which is the better tool?' (which is a silly question to ask anyway since they both have different goals in mind). Instead we answered the question: 'Given the functionality of AToM3, is VM SDK able to do exactly the same?' and the answer was no. Had the tables been turned AToM3 probably wouldn't have been able to match all the features contained in the Visualization and Modeling SDK either. This section will attempt to give an objective view of both programs, describing VM SDK's strong points and the areas where there is still room for improvement, compared to AToM3.

- Strong points of the VMSDK:

Easy to use from a programmer's POV The Visualization and Modeling SDK is integrated in an IDE, an environment most programmers feel comfortable with. This results in a tool that is easy to use and does exactly what you expect it to do.

Everything under the same roof Once you know your way around the SDK it is fast to use. All the files can be accessed at once without having to open multiple instances of the tool or without having to close a file to make way for another one.

Easy design of graphical object Graphical objects are easy to create and to design. The only two flaws are the fact that you can't change the font(color) and that you can't simply drag and drop decorators. Apart from that it is very intuitive and powerful.

- VMSDK: Room for improvement?

Support for simulation As mentioned before there is no support for simulation, which is quite a letdown from this otherwise good tool.

Validation as a core concept Though validation is present in the VMSDK it seems much more as an 'add-on' than as a core concept. In AToM3 they are directly linked to classes (or the entire model depending on the scope).

Automatic Link Detection In the VMSDK you explicitly have to choose a link type before choosing the source and target object. However when connecting two objects often there is only one type of relationship applicable given the classes of both the source and the target. In AToM3 when you connect two items it automatically detects which link is applicable and connects the objects. In the case of multiple possibilities it asks the user which link should be used to connect the two objects.

Learning Curve/Documentation The learning curve may prove to be rather steep for people who have never used Visual Studio before. This is further amplified by the fact that good documentation is rather hard to find.

More options for cardinalities It would be desirable to have more options instead of being forced into one of 4 categories.

6. Future work

As mentioned earlier it may be interesting to do a second comparison of both tools, this time starting from a project specifically tailored for the Visualization and Modeling SDK and then trying to achieve the same results in AToM3, this may deepen our insight in both tools, uncovering shortcomings which may lead to additional features in future versions of both AToM3 and VM SDK.

7. Conclusion

Both tools offer a unique perspective on model-driven engineering, and demonstrate the viability of this approach. When comparing the functionality of both tools with respect to actual model-driven engineering AToM3 is certainly ahead. Mostly from a conceptual point of view (everything is a (meta-)model and validation is an integral part of this model) but also in practical terms, read: simulation. However with the Visualization and Modeling SDK, Microsoft certainly made a step in the good direction, providing a solid tool that could very well serve as a good basis for a stand alone program, expanding on its current functionality.

As a final remark; everybody interested in learning more about the Visualization and Modeling SDK should definitely take a look at Wills (2011). It is a very instructive tutorial on how to get started with the SDK and will greatly improve your insight in the subject.

References

- AToM3, January 2008. Last release.
URL <http://atom3.cs.mcgill.ca/>
- Pérez-Medin, J.-L., 2007. A survey of model driven engineering tools for user interface design. Tech. rep., Laboratory of Informatics of Grenoble.
- VM SDK, April 2010. Last release.
URL <http://archive.msdn.microsoft.com/vsvmsdk>
- Wills, A. C., June 2011. Visualization and modeling sdk - intro lab. Online.