# XMILE: A quick overview

XMILE (**X**ml **M**odel Interchange **L**anguag**E**) is a standard for storing System Dynamics models in a form that is complete, unambiguous and usable by anyone (that is, it is an open standard that does not require any licensing by vendors or individuals using it). Having a standard will be a big benefit to the System Dynamics community as it will allow broader sharing and use of models between users in the community and outside of it. It will also allow us to archive important models in an authoritative manner so that anyone can replicate the result of those models. Finally, it should speed development of specialized tools that will enable greater expansion of the ecosystem supporting work in the field.

The standard has two basic elements - the language for representing the way in which variables should be computed and the language for representing the way in which the model should be presented to the user. The two parts overlap, since models are generally built graphically, but they are distinct enough that tools designed for working with only equations or diagrams will be able to make use of models saved according to the standard.

While the full specification for the standard is approaching 100 pages in length, it basic elements can be explained with a fairly simple example. In this case a model of the growth of the field (a very optimistic in the sense that there are no limits placed on the ultimate number of practitioners). Below is an annotated listing of such a model.

 <?xml version="1.0" encoding="utf-8" ?>

<xmile version="1.0" xmlns="http://docs.oasis-open.org/xmile/ns/XMILE/v1.0" xmlns:isee="http://iseesystems.com/XMILE">

 <header>

The first lines introduce version info - notice the oasis reference. Then the header tells use where the file came from including the software name and version.

 <options namespace="std, isee" />

 <name>Practitioners</name>

 <uuid>f3921add-36a7-4fa9-a651-4b3dfaee6d64</uuid>

 <vendor>isee systems, inc.</vendor>

 <product version="10.1.0" lang="en">STELLA</product>

 </header>

 <sim\_specs method="Euler" time\_units="Year">

 <start>0</start>

This model will run from 0 to 12 years by a quarter of a year. If a file contains more than one model each model can have its own sim\_specs, which might be different.

 <stop>12</stop>

 <dt>0.25</dt>

 </sim\_specs>

 <model\_units>

 <units name="People">

Unit definitions allow software that support units checking to determine if units match by reducing units equations attached to each variable using the aliases and definitions specified here. For example is a has units people and b has units per\_year then a\*b would have units People/Year/

 <alias>person,persons</alias>

 </units>

 <units name="Years">

 <alias>yr,year</alias>

 </units>

 <units name="Per\_Year">

Here is the picture. It shows up in the section <view> below - but there is no space for it on the second page.

 <eqn>1/Years</eqn>

 </units>

 </model\_units>

 <model>

 <variables>

 <stock name="Practitioners">

These are the equations for the model written out in mostly the same way you would see them in a software package. The eqn entry for a stock is the initial value. Here we are just using \*, but there is a pretty good list of standard functions (MIN, MAX, SMTH1...) including specification of how they compute.

The standard also supports arrays, though these were left out of this example for simplicity. Note that the header will change to indicate that the model is using different capabilities so that software that does not support them can give the appropriate message.

 <eqn>100</eqn>

 <inflow>adopting</inflow>

 <non\_negative />

 <units>Person</units>

 </stock>

 <flow name="adopting">

 <eqn>Practitioners \* adoption\_rate</eqn>

 <non\_negative />

 <units>Person/Year</units>

 </flow>

 <aux name="adoption\_rate">

 <eqn>0.03</eqn>

 <units>Per Year</units>

 </aux>

 </variables>

 <views>

 <style color="#FFAAFF" background="white" font\_style="normal" font\_weight="normal" text\_decoration="normal" text\_align="center" vertical\_text\_align="center" text\_background="white" font\_color="blue" font\_family="Arial" font\_size="9pt" padding="2" border\_color="black" border\_width="1" border\_style="none">

 <stock font\_color="#FFAAFF" label\_side="top" label\_angle="0" />

This controls appearance - everything that is drawn can override these value, but if it does not then these will be used (cascading styles).

 <flow font\_color="#FFAAFF" label\_side="bottom" label\_angle="0" />

 <module font\_color="#FFAAFF" label\_side="top" label\_angle="0" />

 <aux font\_color="#FFAAFF" label\_side="bottom" label\_angle="0" />

 <group font\_color="#FFAAFF" />

 <connector color="#55AA00" font\_color="#55AA00" />

 </style>

Zoomed in

 <view show\_pages="false" page\_width="768" page\_height="1001" zoom="220">

 <connector uid="1" color="#55007F" font\_color="#55007F" x="79.75" y="124" angle="153.435">

 <from>adoption\_rate</from>

The curved arrow connectors. One overrides the color, one uses the color specified for connectors in the style section. Connectors are given ids so they can be referred to (most entries are referred to by name to make it easier to read).

 <to>adopting</to>

 </connector>

 <connector uid="2" x="146" y="51" angle="149.421">

 <from>Practitioners</from>

 <to>adopting</to>

 </connector>

 <stock name="Practitioners" color="lime" background="yellow" font\_color="#0055FF" font\_family="Britannic Bold" x="168" y="64" />

 <flow name="adopting" x="87.75" y="64">

 <pts>

The variables and their positions. The stock has a different font, the flow and aux (converter/auxiliary) do not. These could also have size info (as could the style).

 <pt x="30" y="64" />

 <pt x="145.5" y="64" />

 </pts>

 </flow>

 <aux name="adoption\_rate" x="87.75" y="128" />

 </view>

This file has one model with one view. A model can have multiple views (different parts or different representations of the same part) and a file can have multiple models (which would normally be organized in containing modules).

 </views>

 </model>

</xmile>

That is a complete little model. The standards committee is trying to keep a balance between brevity and readability. To learn more go to xmile.systemdynamics.org and you will find a link to the complete draft specification.