XMILE Support in Simantics System Dynamics

Janne Kauttio and Tuomas Miettinen

VTT Technical Research Centre of Finland Ltd
P.O. Box 1000, FI-02044 VTT, Finland
Email: firstname.lastname@vtt.fi

November 19, 2015

1 Introduction

Sharing of models and simulations has always been an integral part of system dynamics research. Initially, when system dynamics was in its infancy, the field was essentially synonymous with the programming language DYNAMO [1] (DYNamic MOdels). The language was designed to be both usable and useful in the then current system dynamics modeling community, which resulted in many advanced features that were ahead of the standard at the time. However, as the field matured and the number of available software tools increased, adherence to DYNAMO eventually diminished. Even though the basic concepts in the models remained the same, different tools gravitated towards different internal model representations, which in turn led to several problems. This was first observed by Myrtveit [2], who argues that different and incompatible file formats used by different tools (or even different versions of the same tool) waste significant amounts of model authors' time and make sharing and distributing models unnecessarily difficult. The solution proposed by Myrtveit was to create a common interchange file format for system dynamics models. Each software tool could then support this interchange format without having to worry about the native file formats used in other tools.

The idea did not gain much momentum until several years later, when Hines [3] reintroduced it (with similar arguments) coined lightheartedly as SMILE (Simulation Model Interchange LanguagE). This time the reception in

the system dynamics community seemed to be more favorable, and an initial implementation of such a file format based on XML called XMILE (XML-based system dynamics Model Interchange Language Entity) was developed [4]. The design was later refined and formalized in [5], where the actual interchange format (SMILE) is separated from its XML representation (XMILE). The development of these standards continued as two separate documents [6, 7].

In 2013, an OASIS (Organization for the Advancement of Structured Information Standards) Technical Committee was founded to develop XMILE [8] (currently called XML Interchange Language for System Dynamics), based on the existing SMILE and XMILE standards. The version 1.0 of the standard was finalized in the latter parts of 2014, and is freely available [9]. Vendor support for the standard is currently limited but gaining traction, i.e. via the SDXchange [1] effort [10].

This report presents and discusses how XMILE support was implemented in our system dynamics modelling and simulation tool *Simantics System Dynamics* [2]. The report focuses mostly on practical aspects discovered in the implementation process and outlines some limitations both in the tool and the XMILE specification. The rest of the report is structured as follows: Sections 2 and 3 outline the XMILE specification and Simantics System Dynamics respectively, which serves as the basis for Section 4, which in turn presents the actual XMILE support implementation and its limitations. Section 5 provides a brief discussion on what was learned and presents some avenues for possible future work.

2 XMILE Specification Overview

An XMILE document contains information about a whole system dynamics model. This Section describes the basic structure of an XMILE document, according to the specification version 1.0 [9]. Note here that as the purpose of this report is to simply communicate our experiences with the implementation of the standard, the following description does not cover every detail of the standard.

Even though the basic building blocks of a system dynamics model, such as the visual representations of elements and the underlying equation system, are essentially the same across the whole spectrum of different modelling tools, there are many advanced features developed through the years that are not. For example, the way the simulation results are presented is highly dependant on the tool: some tools show a separate graph or table of results, some allow

[1] `http://sdxchange.github.io/SDXlate/`

[2] `http://sysdyn.simantics.org/`

them to be included as interactive widgets directly on the diagram, and some do both. In addition, the libraries of mathematical functions provided by the tools often have several subtle differences. Since the primary purpose of XMILE is to serve as an interchange file format, these differences in features have naturally also been taken into account in the design of the specification.

The original XMILE [7] and SMILE [6] specifications defined the concept of three hierarchical *compliance levels* that a tool supporting the standard could implement. The levels were *simulation* (the underlying equation system of the model), *display* (the stock and flow diagram of the model), and *interface* (the user interface on top of the model). The levels were defined in such a way that a tool which only implements the lowest level could still understand models developed with a tool that operates on the highest level. This compliance level model is not used any more in the current XMILE standard, but the underlying idea is still there; the standard is constructed so that only a very minimal set of features is necessary to support the standard but optional constructs for many advanced features are also included and can be easily extended.

To demonstrate this idea, an example of a very basic XMILE document is provided in Figure 1. The document represents a minimal XMILE document in the sense that it only contains sections that are absolutely required by the specification. These sections are basically the header, which describes the origin of the document and the set of XMILE features it uses, and the description of the system dynamics model itself, encoded in the document as a well-defined set of variables and equations. In the case of our example, the model contains three variables (one stock, one flow, and one auxiliary) which together form a simple material decay system (a graphical representation of which is provided in Figure 2).

In addition to the basic model structure, the XMILE document can also optionally define several more advanced features of the model. These features include, but are not limited to, the following:

Simulation Specifications i.e., the integration method and start, stop, and step times used during the simulation

Units used in the model

Dimensions for array variables if they are used, which is a common feature in more advanced modelling tools

Data Connections for external data sources

```
1  <?xml version="1.0" encoding="utf-8">
2  <xmile version="1.0"
3  xmlns="http://docs.oasis-open.org/xmile/ns/XMILE/v1.0">
4    <header>
5      <vendor>Simantics</vendor>
6      <product version="1.9.0">System Dynamics</product>
7    </header>
8    <model>
9      <variables>
10       <stock name="material">
11         <eqn>1000</eqn>
12         <outflow>material_draining</outflow>
13       </stock>
14       <flow name="material_draining">
15         <eqn>material/draining_time</eqn>
16       </flow>
17       <aux name="draining_time">
18         <eqn>10</eqn>
19       </aux>
20     </variables>
21   </model>
22 </xmile>
```

Figure 1: A simple XMILE document

Macros that can be used in model equations to, e.g., implement vendor-specific features that are not currently included in the XMILE specification

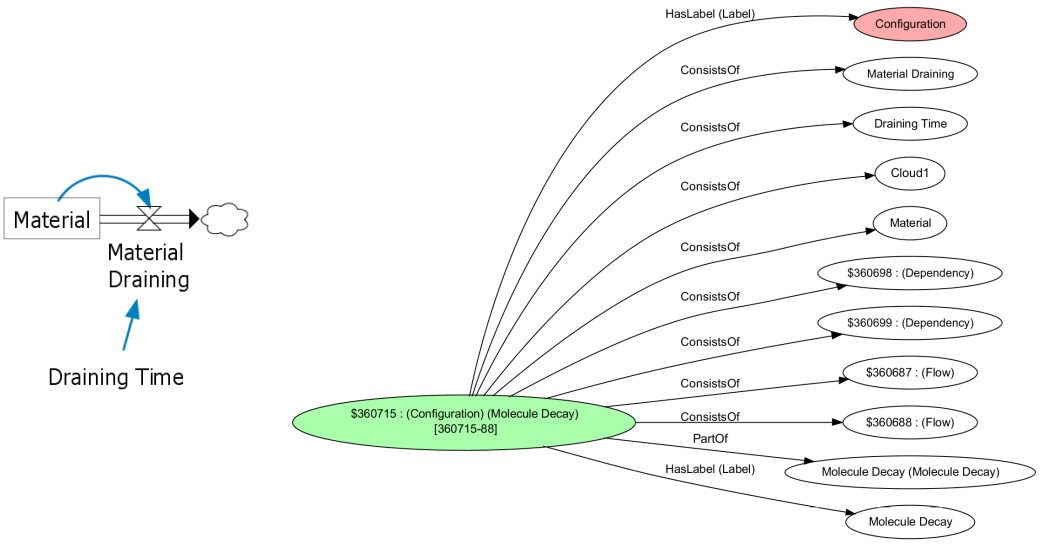
Views for diagram representations of the model structure

It should be noted that, in contrast to the old XMILE and SMILE compliance levels, the XMILE specification does not have a clear hierarchy for the additional features but instead allows them to be toggled one by one with options in the header. On one hand this is extremely useful, since i.e. in the context of Simantics System Dynamics it allows us to a large subset of the specification even though the tool itself lacks some commonly used features and handles some things in ways that are not easily translatable to XMILE. On the other hand this is a somewhat problematic design decision, as the logic by which some of the features have been chosen to be optional whereas others have not is not explicitly clear. This can technically lead into complicated situations where the sets of optional features supported by a modelling and simulation tool do not actually make much sense, and the specification does not express clearly what should be done in this case. This discussion is continued in Section 5.

3 Simantics System Dynamics

Simantics System Dynamics is a free and open source modelling and simulation tool built upon the Simantics [3] software platform, which is a generic modelling and simulation platform based on the Eclipse [4] IDE (Integrated Development Environment). Simantics System Dynamics and the Simantics platform are developed by VTT Technical Research Centre of Finland Ltd and Semantum Ltd, and released under the Eclipse Public License (EPL) [11]. In this Section, the features of Simantics System Dynamics relevant for XMILE support are presented.

The models created with Simantics System Dynamics are stored as *semantic graphs* in the *ontology-based* modelling *database* of the Simantics framework; i.e., all model components have types defined by ontologies and both the components and their types are nodes in a graph. An example model and the respective (partial) graph representation are depicted in Figure 2.



(a) Decay model (b) Semantic graph

Figure 2: The model of material decay and the respective (partial) semantic graph representation

In Simantics System Dynamics, all model variables are defined as equations written in the Modelica [12] language, which is a multi-domain language for

[3] `https://www.simantics.org/simantics`
[4] `http://www.eclipse.org/`

modelling of complex systems. The language allows the user to implement elaborate functionality to the system dynamic models, although at the same time the learning curve is very gentle as no previous Modelica knowledge is required for basic modelling. When a system dynamics model is simulated, the model is automatically converted to a Modelica model after which a Modelica solver is used to simulate the model. A complete model can also be exported from Simantics System Dynamics and imported to any Modelica environment and, furthermore, embedded to other models from different domains.

A major quality attribute which the development of Simantics System Dynamics focuses on is *interoperability*. Interoperability can be further divided into two subcategories: connectivity between software and communication between people. Firstly, the target is to enable connectivity with other software, such as other modelling and simulation tools and tools that provide end-user interfaces for the models. Secondly, the target is to enhance communication between the different stakeholders of a modelling project and also to support collaborative model development for modellers working in separate projects. [13]

For achieving these objectives, the tool enables users to export models as Modelica code, use the components and functions of the Modelica standard library, import models from other modeling and simulation tools, create end-user web interfaces with the Simupedia [5] platform, and define structural reusable module components and libraries for those components.

4 Implementation

The problem of transforming a model between Simantics System Dynamics and XMILE can be separated into a few distinct steps: reading (or writing) the XML document, transforming the model structure and equations from XMILE to Modelica (or vice versa), and writing (or reading) the semantic graph. The relatively straightforward processes of reading and writing XML documents and the semantic graph are covered in Section 4.1 and the more involved process of transforming the model structure with some of our considerations is presented in Section 4.2.

4.1 Reading and Writing XML Documents

Since the OASIS XMILE technical committee provides an XML schema for XMILE, generating a reader and writer for it is extremely straightforward using any of the well-established XML libraries. Since Simantics System Dynamics

[5] `http://www.simupedia.com/`

is written in Java, our library of choice is JAXB [14] (Java Architecture for XML Binding). JAXB can be used to automatically generate an object model from an XML schema, and it provides utilities for transforming an XML document from file to the object representation and vice versa.

Reading and writing the semantic graph is slightly more complicated than reading and writing an XML document. Even though the Simantics platform does provide utilities for graph manipulation, the structure of the XMILE document is different enough from the structure of a Simantics System Dynamics model that a direct translation between the two is not possible. What this means is that we cannot simply walk through the object representation generated with JAXB and generate a new model, adding new variables and connections as they are encountered. The primary reason for this is the separation of the visual diagram from the mathematical model in XMILE since in Simantics System Dynamics these two are tightly interwoven and, for instance, variables cannot be created without visual representations.

Our solution to the aforementioned problem was to utilize an additional intermediate object model, created previously for other model import functionalities, which can be populated and modified on the fly and then written into the semantic graph as a new self-consistent model. The object representation includes sane default values for many things that are required in the Simantics System Dynamics model but that are not necessarily included in the XMILE document, such as the aforementioned visual representations for variables and several simulation settings. With this approach we are able to simply read through the object representation generated by JAXB and populate the object model in one go, which is also fairly efficient as the object representation must only be traversed once.

4.2 Model Transformation

Now that the practical aspects of reading and writing XML documents and the semantic graph have been covered, it is time to consider the relation between XMILE and Simantics System Dynamics models more closely. Most of an XMILE document translates over in a rather straightforward fashion since for the vast majority of the elements in the document a direct equivalent can be found in the Simantics System Dynamics model. This applies, for instance, to most of the variables (stocks and auxiliaries), the diagram structure and, simulation settings. Then there are some things that have slight differences that are easily handled, such as XMILE flows which relate to valves in Simantics System Dynamics and clouds in our tool which can simply be removed since a flow does not need an element at both ends to anchor to in XMILE. There are, however, a couple of things that require some extra

attention.

Modelica is designed to be a multi-domain modeling language for complex systems and thus is in a sense much more general than what is required for system dynamics modelling and simulation. This generality comes with both advantages and disadvantages. On one hand, Modelica provides Simantics System Dynamics with, for instance, proper array support and a vast library of mathematical functions, which is also easily extensible. Then on the other hand, the Modelica language itself is relatively hard to extend, so things like subscripts (named indices for array variables), which are commonly used in system dynamics but not natively a part of Modelica, have to be implemented on top of the model as an inflexible, external layer. This, unsurprisingly, led into some problems in our implementation of the XMILE import and export functionalities. These issues are namely XMILE submodels, array operations, and side-effects in the XMILE standard functions.

In addition to the primary model, an XMILE document can also define any number of named *submodels* which can be included in the primary model. Simantics System Dynamics has a construction called *module* (not to be confused with XMILE modules which are used in a view to represent submodels) which can, in a similiar fashion, be used to implement some part of the model in a self-contained fashion which can then be connected to the rest of the model via inputs and outputs. The important functional difference between submodels and modules is that submodels are essentially complete models which can be simulated independently, whereas modules are more of an abstraction for the benefit for the modeller which can be used to make models cleaner and easier to understand through parametrization and instantiation of repeated parts of model structure. This distinction between these two constructions means that the one can not be truly used to represent the other, since modules are always simulated as a part of the whole model and submodels can not be instantiated. The best compromise we could come up with is to convert XMILE submodels into Simantics System Dynamics modules on import and simply ignore the submodel simulation strategy (i.e., assume the run mode is always *normal* with the same *time step* as the main model), and on export convert each instance of a module into a separate submodel.

Array operations in the XMILE specification are also somewhat problematic for us. In XMILE, all array operations are defined to be element-by-element operations, which makes sense in the context of system dynamics as it is the natural assumption of what the modeller wants. Modelica on the other hand defaults to linear-algebra operations when dealing with matrices (what array variables are in that context), and treats the element-by-element operations as a special case. In practice this means that every operation

that is performed on array variables in XMILE equations should be replaced with the element-by-element variant and vice versa when the XMILE model is imported and exported. This is not straightforward to do as it requires equation parsing during the import and export processes to figure out which variables are actually arrays. The fact is not made any easier by the shorthand supported in XMILE which allows the indices to be omitted when apply-to-all equations are defined. Our import and export processes do not in their current state even attempt to perform any conversion between operations, so the responsibility is left entirely to the user. Fortunately the incorrect operations often lead to inconsistencies in array dimensions which are then caught by the Modelica compiler in Simantics System Dynamics.

The last of the more notable problems in the model transformation process are the statistical functions in the library of mathematical functions each tool that implements the XMILE specification should support. Functions in Modelica are defined to be strictly side-effect-free, and functions in the library that utilize randomness in one way or another are not easy to implement without side-effects. In practice this limitation can be worked around with undocumented vendor-specific extensions in OpenModelica, which is the Modelica implementation we use, or via manually implemented C-procedures, but these solutions are hardly satisfactory. A better solution would be to provide implementations for these functions in our own internal solver which is not restricted by the semantics of Modelica, and fall back to placeholders when necessary. This is also not currently implemented but it is the solution we will adopt in the future. On a related note, Simantics System Dynamics allows the user to define and use arbitrary Modelica functions which is problematic as the problem of transforming them into XMILE macros is clearly non-trivial and out of the scope of this work. We decided to work around this issue by implementing (with slight modification that are handled automatically during the import and export process) the library of standard XMILE mathematical functions into Simantics System Dynamics, and currently only support exporting a model if it uses the library exclusively.

Another lesser issue that should be mentioned is a slight disparity in the identifier semantics (variable names) between the tools. Simantics System Dynamics variable names can not contain underscores, substrings that start with numbers, or several reserved keywords, whereas all of these can appear in XMILE identifiers. Fortunately a common ground between the two different naming schemes was not difficult to find, and all variable names can be simply normalized to this format when importing XMILE models. All in all, the current implementation of XMILE document import and export in Simantics System Dynamics is not entirely without limitations, but mostly implements the base-level conformance as specified in the XMILE specification. More

specifically, on the XMILE file conformance target, Base level conformance is implemented except for support for multiple files and model behaviours, and on the XMILE simulator conformance target, Base level is implemented apart from the limitations mentioned in this Subsection. Model view support is also included (as it is basically a requirement in Simantics System Dynamics) and optional array support is (mostly) included as well.

5 Conclusions

This report presented the current level of XMILE export and import functionalities developed for the Simantics System Dynamics tool. As a key design objective behind Simantics System Dynamics is to enhance interoperability with other tools, with the potential of XMILE becoming a widely supported model transfer file format specification among system dynamics software, supporting both XMILE export and import are considered essential. The current level of implementation is far from complete, but is usable for simple models and actually supports the vast majority of models that are currently publicly available.

The discussion around the XMILE specification itself has progressed somewhat after the implementation work described in this report was done, and there has been some dissuasion between different parties that have attempted to implement the specification. The main source of this controversy is the fact that the specification, at least in its version 1.0 form, is not sufficiently specific, has parts that are difficult to interpret and includes some unnecessary features. It is also our opinion that the specification in its current form should not be treated as complete, since many things (such as the grammar of the macro language) are not provided at all, and other things (such as the behaviours of required functions) are not specified completely. Also, as mentioned previously, the separation of required and optional features should be clarified, and at least our work would have been easier if the graphical representation of the model was part of the required feature set.

As for the future of XMILE support in Simantics System Dynamics, we plan to eventually implement support for as large subset of the specification as possible. A new declarative transformation framework was recently developed for the Simantics platform to make model transformations like this one easier to implement. The XMILE support is also probably going to be migrated into this new system in the near future, which should make adaptation to possible changes in the specification and other maintenance operations easier for us. We hope our work will inspire others to also adopt the standard so we can together make system dynamics better for everyone.

References

[1] Alexander L Pugh. DYNAMO user's manual. 1976.

[2] Magne Myrtveit and AS ModellData. Models crossing the boundaries of tools. *System Dynamics*, 95(1):170–179, 1995.

[3] J Hines. A SMILE for system dynamics. *System Dynamics Newsletter*, 16(1):1–5, 2003.

[4] Vedat G Diker and Robert B Allen. XMILE: towards an XML interchange language for system dynamics models. *System Dynamics Review*, 21(4): 351–359, 2005.

[5] Karim Chichakly. SMILE and XMILE: a common language and interchange format for system dynamics. In *Proceedings of the 2007 International System Dynamics Conference, Boston, MA. System Dynamics Society: Albany, NY*, 2007.

[6] Karim Chichakly. *SMILE: A Common Language for System Dynamics*. Information System SIG of the Systems Dynamics Society, 2013. `http://www.iseesystems.com/community/support/SMILEv4.pdf`.

[7] Karim Chichakly. *XMILE: An XML Interchange Language for System Dynamics*. Information System SIG of the Systems Dynamics Society, 2013. `http://www.iseesystems.com/community/support/XMILEv4.pdf`.

[8] Robert L Eberlein and Karim J Chichakly. XMILE: a new standard for system dynamics. *System Dynamics Review*, 29(3):188–195, 2013.

[9] Karim Chichakly, Gary Baxter, Robert Eberlein, Will Glass-Husain, Robert Powers, and William Schoenberg. *XML Interchange Language for System Dynamics (XMILE) Version 1.0*. OASIS Committee Specification, 2014. `http://docs.oasis-open.org/xmile/xmile/v1.0/xmile-v1.0.html`.

[10] Robert Ward, James Houghton, and Ivan A Perl. SDXchange: standalone translators to enable XMILE model adaptation, transportation, and exchange. *System Dynamics Review*, 31(1-2):86–95, 2015.

[11] Eclipse Public License – Version 1.0. `https://www.eclipse.org/legal/epl-v10.html`.

[12] Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 2.1*. John Wiley & Sons, 2010.

[13] Teemu Lempinen, Sampsa Ruutu, Tommi Karhela, and Peter Ylén. Open Source System Dynamics with Simantics and OpenModelica. *Proceedings of the International System Dynamics Society*, 2011.

[14] Joseph Fialli and Sekhar Vajjhala. The Java architecture for XML binding (JAXB). *JSR Specification, January*, 2003.