XMILE: towards an XML interchange language for system dynamics models

Vedat G. Diker** and Robert B. Allen

Abstract

We propose that XML should be used as an interchange language for system dynamics models and we have designed and implemented a demonstration that we call XMILE. We believe that there will be many benefits for the system dynamics community in developing a full-scale interchange language. This paper discusses the motivation factors behind the idea of developing an interchange language for system dynamics models, and explains why XML is a good candidate framework. An initial implementation of XMILE is also presented along with initial findings about its applicability to models built on various system dynamics software platforms. Copyright © 2005 John Wiley & Sons, Ltd.


Introduction

It has been argued that the field of system dynamics would benefit from an “interchange language” that would enable conversion of system dynamics models and model-based applications between available software packages. This paper proposes an XML-based representation scheme for system dynamics models. We call the scheme XMILE (XML-based System Dynamics Model Interchange Language Entity), after the term SMILE (Simulation Model Interchange Language Entity; see the Acknowledgements section for the origin of the term SMILE). The scheme can be used as an interchange standard to enable conversion of models among system dynamics modeling and simulation software. XML (W3C-XML) has many advantages as a framework for an interchange standard: It is flexible, widely accepted, and there are many tools to support its use.

Motivation

An interchange standard offers many advantages to individual modelers and to the field as a whole. Individuals will be able to view and run models built by others without the extra effort of manual conversion between software-specific...
formats. Modelers will be able to reach a wider audience not only with written work based on models but also with running models. The field of system dynamics as a whole will also benefit from an interchange standard. Specific examples illustrate the potential benefits.

Reinforcing peer review

An important requirement for a scientific field to stay “healthy” is the ample availability of venues for peer review. For a field like system dynamics, which has the activity of building and analyzing models in its core, peer review of models is as crucial as peer review of papers and reports. There are many “passive” ways of reviewing a model, including critically reading the equations of the model and investigating the printed graphs of model behavior from simulation runs carried out by the modeler. However, an “active” approach that involves the actual simulation of the model by the reviewer is by and large the most rigorous approach.

A serious barrier for such an active approach occurs whenever a person reviews a model that is built in a software platform different than the one he/she uses. System dynamics software companies are generally very good in supplying free limited-edition versions, model readers, or trial versions of their software. So, obtaining the “right” software to run the model is not an insurmountable barrier. However, locating, downloading and installing the “new” software can become a time-consuming and tedious task. Even more time-consuming is learning the new software, if the reviewer is not familiar with it.

The reviewer may simply give up at some point of the lengthy process and go back to “passive” reviewing methods (that is, if artifacts about the model that lend themselves to conventional reviewing methods such as papers do exist). A model interchange language could foster a higher level of active peer review within the community by facilitating conversion of models between different software platforms.

Lowering barriers to entry for valuable niche software

The field of system dynamics has always benefited from high-quality, highly accessible software. New tools for model development and analysis provide new ways of looking at our models and improve the speed and depth with which we understand the problems at hand and develop plausible solutions. System dynamics software companies constantly strive to incorporate new tools in their respective software. However, some potentially valuable additions to the system dynamics “toolset” cannot make their way into mainstream use if the inventor fails to convince one of the few major system dynamics software companies about the value of adding the new tool to their software as a feature. In such cases the inventors have the option of delivering the new tool as a stand-alone, “niche” software product. This comes with a number
of challenges, and a really big challenge is to handle models built on other software platforms. Without effectively handling all such models, the new niche software cannot reach its full “market” potential. The lack of effectively handling different file formats is an impeding factor in such cases. A model interchange language would remove this barrier and make it easier for new, useful software tools to reach a wide audience.

Fostering increased collaboration between different research segments that have developed around specific software

Certain segments of the field of system dynamics have very specific choices about modeling software platforms they use. As an example, Stella (Richmond et al. 1987) has long been the choice of researchers and practitioners working on K-12 education. This means that the users of other system dynamics software might have problems studying the K-12 applications of system dynamics, or finding a wide audience for their K-12-related models built on software other than Stella. This kind of barrier can be observed in other application areas as well. An interchange standard, which will facilitate easy and accurate conversion of models between system dynamics software, will improve the connections between different research segments within the community.

A standard for system dynamics models

As system dynamics software platforms evolve, they involve a growing number of platform-specific built-in functions and specialized variable types. Consequently, system dynamics models include a growing number of specialized structures or equations that are specific to the software platforms they are built in. This brings about a situation where the “same” model is not (or in some cases cannot be) represented “the same” in two different platforms. More and more, we are building “Stella models,” “Vensim models,” “Powersim Models”, rather than system dynamics models. While it is beneficial to take advantage of specialized features offered by different software platforms, this trend should not take us to a point where the semantics of a system dynamics model is ultimately defined by a given software platform. An interchange standard can help define and maintain the core semantics of system dynamics models. This would help distinguish the platform-specific features of a model from the core features.

Facilitating model conversion

between different modeling software There is no doubt that easy and simple model conversions between different modeling software would make life easier for modelers who need to migrate a model from one software application to another. However, a more indirect, but equally significant consequence of
simple and easy model conversions would be the possibility of using two or more modeling software applications in parallel while working on a single model. This would bring about a whole new way of building, analyzing and presenting system dynamics models. Modelers would be able to use one application to build, another to analyze, and yet another to present their models without extra effort, thus making use of the best features of each application. The option of analyzing or testing a model on several applications, making use of application-specific analysis features, would provide a wider spectrum of insights about the model, and arguably lead to a more robust model. Modelers also would be free to choose the most suitable application to present their models, based on considerations such as audience characteristics and model size.

**Between different versions of the same modeling software** In addition to problems involved in converting models between different software applications, converting models between different versions of the same modeling software can sometimes be a challenge, too. Modelers who have needed to study or improve a model that had been built several years earlier are rather likely to have experienced those challenges. Easy conversions between different versions would not only save time and effort on the part of modelers when such a conversion is necessary, but also make archiving and reusing models much easier. There would be a certain level of assurance that a model archived today will be available for use many years later. And when the time comes to revisit and reuse an old model, there would be no major compatibility problems between the model and the new version of the software application. This, hopefully, would motivate individual modelers and research segments within the community to invest more time and energy in archiving and disseminating their models.

*Facilitating collaborative modeling*

Last but not least, a standard, XML-based representation of system dynamics models would provide opportunities to use versioning tools such as CVS (Vesperman 2003) and sub-version (Collins-Sussman *et al.* 2004) to version models. This would facilitate large-scale collaborative model-building endeavors.

**Open standards**

XMILE will not be a translation engine. It will be a platform-independent, stand-alone representation of a system dynamics model. Translation from one model format to another will be realized by first translating the model file to a XMILE representation and then from XMILE to the other model format. The two translation “engines” will be independent. In fact, we expect translation
applications (to and from XMILE) to emerge independently, possibly through efforts of separate people or groups.

We envision XMILE as an open standard. This will motivate individuals and groups to take the initiative and develop the needed translation applications and it will shift the burden from the software companies to the community as a whole. Furthermore, any given software vendor or developer will only need to build two applications—one to parse the native data format to XMILE and one to transform XMILE to the native data format—instead of having to build separate applications to translate to and from each of the other native model data formats. XMILE’s payoff increases as the number of native formats increase. Of course, the number of translation applications needed to translate everything among $N$ platforms is $N \times (N - 1)$, and thus grows geometrically, while the total number of parsers and transformers between $N$ platforms and XMILE is $2N$, which grows arithmetically.

**XML and XMILE**

The key insight behind XML is that the structure should be cleanly separated from the display format. In XML, the structure is defined as a set of hierarchically organized tags. The structure we developed is shown in Figure 1 and forms a type of XML DTD.$^2$ An example of the tagging is shown in Figure 2.

**Goals of the implementation**

We employed a simple population growth model to show how it can be converted from several implementations (Vensim and Stella) to XMILE. We also show how the XMILE version can be re-created as a Vensim (Eberlein and

---

**Fig. 1. Tag hierarchy for XMILE**

```xml
<File>
  <Model>
    <Variables>
      <Name>Type</Name>
      <Value>Units</Value>
      <Comments>Control</Comments>
      <Value>Units</Value>
      <Comment>Display</Comment>
      <Unit Equivalents>Games Controls</Unit Equivalents>
      <Data.File/>
  </Variables>
</Model>
```
One of the virtues of separating the content from the format is that an XML content representation can be displayed in a variety of presentation languages. For documents, this translation is usually accomplished with the XML Style Transformation language (XSLT). However, we felt that the current implementation of the XSLT scripting language would not be flexible enough for this special application. Thus, we developed a C program to translate XMILE to Vensim format to demonstrate that such translation is feasible.

Peterson 1992) file and can also be used by tools such as Digest. Our example is intended more as an illustration of this activity than as a comprehensive solution. We were also able to convert from Stella to XMILE based on the Stella text file. It would have been better to employ the Stella binary file but we decided not to spend time on the time-consuming job of analyzing the file structure at this stage. Thus, the “units” were not available and we had to use default units.

Fig. 2. Fragment of tagging for a population model, including representations for a constant, a flow, and a stock.
XMILE to Digest

An example of the utility of this work would be to facilitate importing models into tools such as Digest (Mojtahedzadeh et al. 2004). Digest identifies the dominant structure that determines the behavior of a given variable in a system dynamics model. Currently, Digest imports most Stella models processed through a special treatment. The equation listing of a Stella model should be copied and pasted in a text file, which then can be imported by Digest. However, Digest fails to import most Vensim models. XMILE could solve this by serving as an interchange language. The Vensim model could be converted to XMILE and from there to a format that Digest can read. Although we do not have a complete solution to the problem at this time, we were able to demonstrate the potential of translating from one format to another through XMILE, by running our XMILE population model extracted from Stella on Digest.

Discussion

We have demonstrated the feasibility of exchange but we have not constructed a complete system for this exchange. We believe an interchange standard and associated tools should be built on an open standard developed by the contribution of all the interested parties within the field. We see this as crucial for several reasons:

1. Wide participation will allow many good ideas to emerge and compete. Moreover, contribution by different groups and individuals will help address the needs and concerns of a wider portion of the community.
2. The development of a complete XMILE will require a substantial amount of time and effort. A high level of participation will ensure that the burden of development is duly shared by the system dynamics community.
3. One way to foster a high level of participation would be to form a consortium endorsed by the Policy Council of the System Dynamics Society.

Ultimately, the development of an interchange is a chance for the community to define its scope and priorities. As a related project, we believe that an online system dynamics model repository should be developed using an archive such as SourceForge (www.sourceforge.org). Such a repository would be facilitated by an interchange language for the models. The models could then be ported across platforms.

An underlying philosophical issue is at what level of description XMILE can or should be aimed. Ideally it would be a common “interlingua” which would mediate the semantics of all approaches. For instance, some system dynamics software employ discrete and complex model elements such as “queues” and “ovens,” while other software does not. Is discrete event simulation integral to
system dynamics? What about agent-based simulations? These questions are similar to those encountered in attempts to develop an interchange language for mechanical engineering designs (Sykman et al. 1999) and in the attempt to develop a conceptual description associated with mathematical equations represented in XML (W3C MathML). While Content-MathML identified natural units of mathematical expressions, it does not attempt to develop a full mathematical interlingua.

Conclusions

In this paper, we discussed why we think an interchange language for system dynamics models is a critical concept for the advancement of the field, and why XML is a good candidate for this task. We demonstrated that an XML-based interchange language can indeed be used as a platform-independent representation of system dynamics models. Development of such an interchange language, which we tentatively called XMILE, requires a lot of time and effort. It is our position that XMILE should be developed with broad participation of all parties that have a stake in the advancement of the field of system dynamics.

Notes

1. It is our position that a system dynamics model exists independent of the platform on which it is built and run, an argument which deserves more discussion and should be addressed by another article.
2. A DTD is a Document Type Definition. This defines the legal elements of a document type. Another way of implementing DTDs is with an XML Schema structure. Our version is closer to XML Schema.

Acknowledgements

We build on many discussions in the community. One of the most recent open discussions on this subject took place on the System Dynamics Listserv over the months of February and March in 2003. The discussion started with a query from Carolus Grütters, who coined the term “Simulation Model Interchange Language Entity (SMILE).” Will Glass-Husain, Jim Hines, Raymond Joseph, Leonard Malczynski, Michael McDevitt, Magne Myrteite, Anastassios Perdicoulis, George Richardson, George Simpson and Jim Thompson contributed to the discussion. We build on and add to their perspectives. Readers are referred to the System Dynamics Listserv Archives at http://www.ventanasystems.co.uk/forum/ for details of that discussion. Another recent discussion took place on July 22, 2003 at the 21st International Conference of the System Dynamics Society in New York City. The session, titled “A Big SMILE
Discussion,” was facilitated by Jim Hines, and was included in the formal conference program. Vedat Diker thanks Luis Luna-Reyes for valuable discussions and comments on the subject.

References


