Automatic translation from System Dynamics to other formalisms with application to socio-bio-physical systems

John Tinnerholm[b,*] Mariano Zapatero[a] Adrian Pop[b] Peter Fritzson[b] Rodrigo Castro[a]

[a]*Universidad de Buenos Aires and ICC-CONICET, Argentina,* [b]*Linköping University, Linköping, Sweden*
[*]john.tinnerholm@liu.se

Abstract

System Dynamics is a modelling paradigm that has been applied to a wide range of systems, from economic to physical and from managerial to ecological. The main strength of the paradigm is its ease of use.
A System Dynamics modeller does not need to focus on equations; instead, models are expressed in terms of stocks and flows. Modelica, on the other hand, is an equation-based modelling language capable of multi-domain modelling using equations. It gives the user more freedom but requires more mathematical focus and skills.
Therefore, a unification of equation-based modelling and the System Dynamics paradigm is seen as highly beneficial. Advantages include the ability for System Dynamics modellers to use the tools available in the Modelica ecosystem. Furthermore, it allows the integration of System Dynamics models into Modelica models.
To achieve this goal, we developed an XMILE-to-Modelica translator that maps System Dynamics models represented in the XMILE standard exchange format to Modelica models. We also applied a Modelica-to-Julia translator to demonstrate the generality of the approach.
We translated several models to test the correctness of the translator. In particular, the Earth System Climate Interpretable Model (ESCIMO) was translated from its original version in the Vensim toolkit into the OpenModelica toolkit, and a correct validation was obtained by comparing simulation results between simulators.
Our work improves tool interoperability and further demonstrates the feasibility of using Modelica as a unified, standard language to integrate models created using System Dynamics, including large and complex socio-bio-physical systems.

1 Introduction

Currently, there exists no single unified environment for Modeling and Simulation (M & S). Instead, modelers utilize several heterogeneous modeling environments and paradigms. Consequently, modeler know-how and expert knowledge encoded in specific models are not necessarily available in all environments. The resulting branching for modeling practice imply that unnecessary idealizations and simplifications are made when modelers integrate concepts outside their area of expertise in their models. While there exist frameworks and methodologies such as the FMI-standard[1] that allow modelers to utilize models developed in different tools by constructing *Functional Mockup Units* (FMUs), these FMUs can then be simulated together using co-simulation or importing using model exchange (Gomes et al., 2018). However, this introduces additional complexity into the resulting composite model and thus complicates the analysis of the model variables and equations. Furthermore, the modeler may need to maintain models across a set of heterogeneous tools.

[1]Accessed 2023-05-02: fmi-standard.org

In order to facilitate modeling knowledge sharing and unify tools across different ecosystems, we present our approach of integrating models from the modeling paradigm *System Dynamics* (SD) into both the Modelica and Julia ecosystems by means of automatic translation: from XMILE into Modelica, and then from Modelica into Julia demonstrating the usefulness of Modelica as a unified equation-based language. To this end, we translated a rather complex SD model, the climate model *ESCIMO* (Earth System Climate Interpretable Model) (Randers et al., 2016) to test the aforementioned mapping. Furthermore, our approach also includes:

- The possibility to export SD models encoded in XMILE to FMUs.
- Integration of preexisting Modelica models into SD models.
- Increase model exchange between the Modelica, Julia, and SD communities.
- Increased tool support for SD models including bifurcation analysis and visualization.

- Interactive scripting of SD models in the Julia and Modelica environments.

1.1 Organization

The remainder of this paper is organized as follows: We introduce System Dynamics, Modelica and Julia in Section 2, Section 3, and Section 4 respectively. Following these sections, an extended motivation behind our work is presented in Section 6. The climate model, ESCIMO used as the motivating example for this work is presented in Section 5, the implementation of the translator from XMILE to Modelica is presented in Section 7, and the simulation results are presented in Section 8. Finally, we outline directions for future research in Section 9.

2 System Dynamics

System Dynamics (SD) is a modeling paradigm developed by the group of Jay Forrester at MIT in the 1950s (Forrester, 2007). In SD, modelers develop their models as webs of interacting positive and negative feedback loops, using the notion of stocks and flows as building blocks. Stocks represent the accumulation of an inventory (such as fish in a lake or cars at a dealership). Flows represent rates of change to such inventories. Using these notations SD modelers can develop models for complex systems, such as climate models or large socioeconomic models, a well-known model being the World3 global model (D. L. Meadows et al., 1974). Several environments support SD modeling, including *Stella*[2], *Vensim*[3], and *Simantics System Dynamics*[4].
To increase operability between different tools in the System Dynamics community, OASIS developed the *Interchange Language for System Dynamics* (XMILE) standard[5]. XMILE is an XML-based open exchange format that encodes System Dynamic models.

3 Modelica

Modelica is an equation-based object-oriented acausal modeling language developed by the Modelica Association. Modelica aims to be a unified language for equation-based modeling of (but not limited to) cyber-physical systems (Fritzson, 2015). Several toolkits support the Modelica language, including: *Dymola*[6] by Dassault Systèmes, *Modelon Impact* by Modelon[7], and the OpenModelica environment (Fritzson et al., 2020) by the Open Source Modelica Consortium (OSMC).

Modelica differs from the SD paradigm because it supports both causal and acausal modeling and also object-orientation. As a consequence, SD models may be expressed in Modelica; however, not all Modelica models can easily be expressed using classic SD notation since Modelica is a more universally applicable formalism.
Due to the universal application of the equation notation of Modelica, there are libraries that can be used for the development of SD models with Modelica, one of the most well-known being the *System Dynamics* visual library (Cellier, 2008). Also, Modelica tools can be used for simulation as the backend of other tools such as the Simantics System Dynamic Tools (Lempinen et al., 2011). In this case, the models are expressed using the SD formalism, and they are automatically translated internally to Modelica to be simulated using the OpenModelica framework.

4 Julia

Julia is a programming language developed with a strong focus on numerical computing along with powerful metaprogramming capabilities (Bezanson et al., 2017). In recent years, the Julia language has received increased attention being awarded the Wilkinson price for Numerical Software in 2019. Due to this focus, several M&S environments have been developed for the Julia Language, with *ModelingToolkit.jl* (MTK) (Ma et al., 2021) being one of the most well known. To combine the power of Modelica and Julia, we have previously developed a framework capable of translating Modelica models into Julia (Tinnerholm et al., 2022).

5 The Earth System Climate Interpretable Model (ESCIMO)

ESCIMO (Randers et al., 2016) is an SD model that represents the global climate system, focusing on a time range from 1850 to 2100, and including factors such as fluctuations in sea levels and global temperature.
In an article published in Nature in 2020, the model was extended to simulate the global climate up to the year 2500. The model's forecast predicts that even if man-made greenhouse gas emissions were to stop in 2020, the global temperature would still continue to rise (Randers & Goluke, 2020).
The main components of the ESCIMO climate model and their interactions are depicted in Figure 1. The ESCIMO model consists of 1181 equations and variables.

5.1 ESCIMO and Earth3

ESCIMO has previously been integrated as a submodel in a larger socio-bio-physical model named

[2] Accessed 2023-05-06 Stella
[3] Accessed 2023-05-02: Vensim
[4] Accessed 2023-05-06: sysdyn.simantics.org
[5] Accessed 2023-05-06: XMILE
[6] Accessed 2023-05-02: Dymola
[7] Accessed 2023-05-02: Modelon Impact

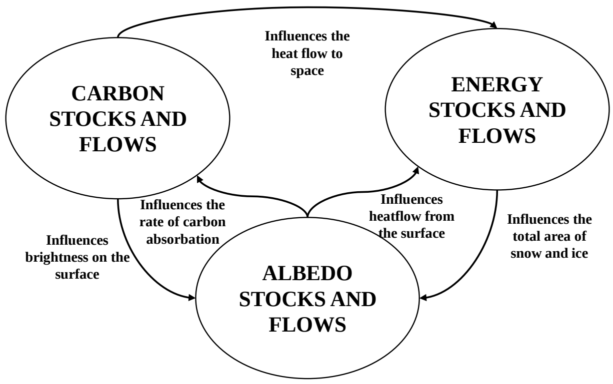

Figure 1. The three sectors of the ESCIMO climate model as described by (Randers et al., 2016).

Earth3.
The *Earth3* model was developed to examine the effect on the planetary boundaries if 14 out of the total 17 global Sustainable Development Goals (SDGs) agreed by the UN in 2015 were to be fulfilled (Randers et al., 2019). The conclusion of the simulation experiments are that it is not possible for humanity to achieve these 14 SDGs while at the same time not violating the planetary boundaries by 2030 or 2050 if the business-as-usual scenario (as defined by the author) continues.

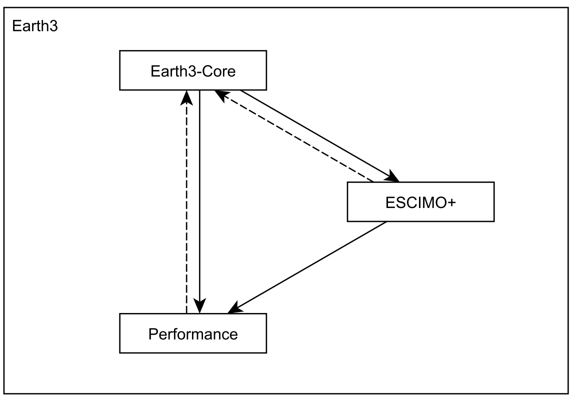

Figure 2. High-level overview of the Earth3 model where a variant of the ESCIMO mode called ESCIMO+ and Earth3-Core are sub-models. The dashed lines illustrate possible future feedback loops.

A high-level overview of the model is depicted in Figure 2. The model consists of three key sub-models:

- The ESCIMO model that models the global climate.
- The Earth3-Core model that models the socio-economic behavior.
- The performance model that calculates the performance with respect to the SDGs.

The Earth3-Core model (Randers et al., 2019) was developed in Microsoft Excel as a spreadsheet model, whereas ESCIMO+ was developed using the SD paradigm, as previously discussed. In its current formulation, without flows closing loops from ESCIMO+ back to Earth3-Core, a changing global climate will not affect the behavior of humanity as represented by the Earth3-Core model. Still, Randers et al. emphasize that the lack of these feedback loops has a greater effect on model variables after the year 2050, which was beyond the duration of the simulation experiment presented in the paper. This fact serves as a motivation behind the work presented in this article and, as explained in the introduction, translators from one modeling paradigm to another can yield substantial benefits.

6 Motivation

There exists a plethora of heterogeneous modeling and simulation tools. Although solutions exist that allow modelers to integrate models from different tools, such as the FMI standard, not all tools support this. In other cases, such as for the Earth3 model in Section 5, this is achieved by using the integration capabilities of existing tools; however, as discussed, this also imposes different limitations for each case. Simulating models using co-simulation adds extra complexity such as the selection of suitable master algorithms and a suitable step size, and might require mastery of several modeling paradigms as well as domain-specific tools in order to develop and maintain several models in tandem. Also, different scientific disciplines are accustomed to using different tools and languages to express their models; this leads to an ecosystem of modeling techniques that can undermine the development of more complex systems. Hence, in this work we propose to use Modelica as a unified, formal, and standard language to integrate models created both in spreadsheets and System Dynamics (SD).

6.1 Why Modelica

We argue that Modelica is a good fit for a unified language given it is open, standardized, object-oriented, and equation-based. As a consequence, it supports both acausal and causal modeling. This allows causal models encoded for instance, in SD to be encoded in Modelica. An example of an SD model is given in Figure 3.
A Modelica model for the SD model in Figure 3 is available in Listing 1. This exemplifies how Modelica's inheritance and composition permit an advantageous component-based approach to reduce duplicated equations in our model.
While there exists research proposing a similar component-based approach for the SD paradigm

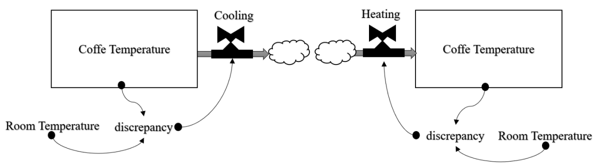

Figure 3. A simple SD model modeling heat over time in a coffee cup. This example is adapted from (D. H. Meadows, 2008).

(Bauer & Bodendorf, 2006) it has yet to reach mainstream adoption (Yeager et al., 2014).
Similarly to many SD tools, Modelica environments usually include a graphical notation that modelers can use to compose models using drag and drop. Examples of graphical modeling libraries for SD in the context of Modelica include the *System Dynamics* library (Cellier, 2008).

```
model CoffeeCup
  parameter Real cStartTemp = 60;
  parameter Real roomTemp = 18;
  Real roomTemperature(start = roomTemp);
  Real coffeeTemperature(start = cStartTemp);
  Real tmpChange;
  Real discrepancy;
equation
  roomTemperature = 18;
  tmpChange = discrepancy / 10;
  der(coffeeTemperature) = -tmpChange;
  discrepancy = coffeeTemperature - roomTemperature;
end CoffeeCup;

model Scenarios
  CoffeeCup coffeeCupHeating(cStartTemp = 5);
  CoffeeCup coffeeCupCooling(cStartTemp = 60);
  Real hotCoffeeCooling = coffeeCupCooling.tmpChange;
  Real hotCoffeeHeating = coffeeCupHeating.tmpChange;
end Scenarios;
```

Listing 1. Modelica model of the SD coffee cup model depicted in Figure 3. Here we use inheritance via modification to enable the two scenarios.

6.2 *Why Julia*

In addition to Modelica, we successfully experimented with translating the resulting Modelica version of the ESCIMO model to Julia.
There are several reasons for this translation. It exemplifies the ease of translation from a standard modeling language to other languages, and it provides access to the simulation runtime of OpenModelica.jl. The latter comes with extensions to the Modelica language for so-called *Variable-Structure-Systems* which allows conditional changing equations models during simulation. Hence, models simulated in this environment can be further modified to include scenarios where the dynamics of models radically change during the course of a simulation (Tinnerholm et al., 2022).

Access to the Julia ecosystem also comes with several advantages such as a wide set of scientific machine learning tools enabling domain-aware and physics-informed learning, state-of-the-art tools for bifurcation analysis[8] and interactive visualization[9] to name a few.
To conclude, in this section we have provided an extended discussion to exemplify the advantages of an automatic translation from System Dynamics to other formalisms. For further details, we refer to (Castro, 2019) which provides an extended discussion of this topic in the context of global models.

7 Mapping XMILE To Modelica

XMILE is a standard format that allows the interchange of SD models between toolkits. In order to map XMILE to Modelica an initial proof-of-concept translator was written in Python. The translator works by mapping entities described in the XMILE standard to corresponding entities in Modelica. For brevity, we will not describe all elements of the translator here (the full source code of the translator and the resulting models are available upon request).

```
<model>
  <sim_specs> <!-- OPTIONAL-->
  ...
  </sim_specs>
  <behavior> <!-- OPTIONAL-->
    ...
  </behavior>
  <variables> <!-- REQUIRED -->
    ...
  </variables>
  <views> <!-- OPTIONAL-->
   ...
  </views>
</model>
```

Listing 2. High level structure of an SD model encoded in XMILE (OASIS, 2015)

Listing 2 describes the overall structure of the model tag in XMILE. The current translator from XMILE to Modelica enumerates all variables and all equations of an XMILE document. Then, for each variable tag, it maps it one-to-one into a Modelica variable while keeping auxiliary information (such as units and dimensions). Currently, the translator supports the following categories of variables:

- **auxiliary**
- **stock**
- **flow**
- **delay1i**
- **delay3i**

[8] Accessed 2023-05-09 BifurcationKit.jl
[9] Accessed 2023-05-09 Interact.jl

- **delay3**
- **smooth3**
- **smooth**
- **sample_if_true**

Each category for each variable is saved both to be encoded in the final Modelica model and to generate the correct equations. Likewise, the initial values of each variable are used to construct the initial equations of the resulting Modelica model.
The equations of the model are constructed based on the category of each variable. The equations for auxiliary variables are translated *verbatim* since they may be mapped to simple algebraic equations. However, other categories of variables need to be transformed into an equivalent Modelica construct. The XMILE standard (OASIS, 2015) defines the stock as:

$$stock_t = stock_{t-dt} + dt \cdot (inflows_{t-dt} - outflows_{t-dt})$$

While this form is suitable for explicit solvers typically used in SD environments, in Modelica, the time step is not available directly during the simulation, so instead, this is reformulated as

$$der(stock) = inflows - outflows$$

where $der(stock)$ is the continuous time derivative.
The mapping for a subset of these categories to the corresponding Modelica equation is presented in Table 1.

Table 1. Subset of Modelica to SD type matchings

SD Type	Modelica Formulation
stock	$der(stock) = inflows - outflows$
smooth	$der(smooth) = \frac{input - smooth}{averagingTimeVariable}$
flow	$flow = inflow$

We tested the translator with the ESCIMO model described in Section 5.
Since ESCIMO is a part of Earth3 and takes some input from spreadsheets, we also needed to integrate an Excel parser in the translator.
The components of the spreadsheet model as defined in Excel were mapped to Modelica lookup tables[10]
Once this mapping was complete, we validated the model by running it using OpenModelica. The translation to Julia was simple, as described in Section 4; an existing compiler from Julia to Modelica was used for this purpose. The Julia compiler was validated by comparing the simulation results of the resulting simulation code with that obtained from OpenModelica. An excerpt of the ESCIMO model translated to Modelica is available in Listing 3. The full Modelica and the resulting full Julia models are available upon request.
This section has presented the XMILE to Modelica translator capable of translating a significant subset of XMILE as used by the ESCIMO model to Modelica. The next part of this paper will present the results of our validation experiments.

[10]In Julia, these tables were defined and implemented using a foreign function interface, OMRuntimeExternalC.jl, accessed 2023-05-16.

```
model ESCIMO
  constant Real Future_volcanic_emissions(unit =
  ↪ "GtVAe/yr") = 0.0 "CONST";
  constant Real Albedo_Antarctic_sens(unit = "fraction") =
  ↪ 0.7 "CONST";
  constant Real
  ↪ Annual_pct_increase_CH4_emissions_from_2015_pct_yr(unit
  ↪ = "1/yr") = 0.0 "CONST";
...
initial equation
  Antarctic_ice_volume_km3 =
  ↪ Antarctic_ice_volume_in_1850_km3   "STOCK";
  Arctic_ice__on_sea__area_km2 =
  ↪ Arctic_ice_area_in_1850_km2   "STOCK";
  C_in_permafrost_in_form_of_CH4 = 1200.0   "STOCK";
...
equation
...
  der(DESERT_Mkm2) =
  ↪ flow_Shifting_GRASS_to_DESERT_Mkm2_yr -
  ↪ flow_Sifting_DESERT_to_GRASS_Mkm2_yr   "STOCK";
  der(Fossil_fuel_reserves_in_ground_GtC) =   -
  ↪ flow_Man_made_fossil_C_emissions_GtC_yr   "STOCK";
  der(GRASS_area_burnt_Mkm2) =  flow_GRASS_burning_Mkm2_yr
  ↪ - flow_GRASS_regrowing_after_being_burnt_Mkm2_yr
  ↪ "STOCK";
...
  UNIT_conversion_for_CH4_from_CO2e_to_C =  1/(16/12 *
  ↪ Global_Warming_Potential_CH4)    "AUX";
  UNIT_conversion_for_CO2_from_CO2e_to_C =  12/44    "AUX";
  UNIT_conversion_from_MtCH4_to_GtC =  1 /( 1000 / 12 *
  ↪ 16)    "AUX";
...
  flow_SW_surface_absorption=SW_surface_absorption
  ↪ "FLOW";
  flow_GRASS_runoff=GRASS_runoff   "FLOW";
  flow_NATURE_CCS_Fig3_GtC_yr=NATURE_CCS_Fig3_GtC_yr
  ↪ "FLOW";
...
end ESCIMO
```

Listing 3. Excerpt from the translated ESCIMO model, showing initial equations and equations for some of the models' stocks and flows.

8 Simulation Results

During the course of our work, we experimented with several iterations of the ESCIMO model. We generated a corresponding Modelica model for the three ESCIMO models presented in (Randers et al., 2016, 2019; Randers & Goluke, 2020). This section, however, will present our validation of the ESCIMO model as presented in (Randers & Goluke, 2020). It should be noted that there are several configurations (or scenarios) in which one can simulate the model. The validation presented in this section concerns Scenario 1. The results were gathered from three simu-

lations: the publicly available ESCIMO model (baseline), the Modelica translation of this model, and the Julia model.
In our experiments, we compared the simulated values at every decade between 1850 to 2500 using the translated models and the reference model for the following variables:

- *Temperature surface anomaly compared to 1850* (Celsius), that is, the difference in average global surface temperature compared to 1850.
- *pH in warm surface water*, that is, the acidity of warm surface water.
- *CO2 Concentration in PPM*, that is, the concentration of carbon dioxide in the atmosphere.

The following numerical solvers were used:

- Runge-Kutta-4[11] for the Vensim simulation.
- DASSL for the OpenModelica[12] simulation with variable step-size, the absolute and relative tolerance was set to $1E-6$.
- Rodas5[13] for the Julia simulation with variable step-size, The absolute and relative tolerance was set to $1E-6$.

The resulting plots of these variables for the ESCIMO model are depicted in Figure 4. From the plots in the figure we observe that there is no visible difference between the three models. The graphs reveal that the translated models generate the same outcomes with minimal variations.
The percentage difference between the original model and the translated Modelica model for the **Temperature Anomaly** is presented in Figure 5. The graph shows that the difference between the original model and the resulting Modelica variable was, at most around 1%.
A plot highlighting the difference between the Julia and Modelica model for the variable **Temperature Anomaly** can be seen in Figure 6. As in the previous experiment, the difference between the variables was far below one percent, so it is not shown. In Figure 6, we can see that there are no significant differences between the Julia and OpenModelica environments. The largest difference in values observed between the Modelica and Julia models in Figure 6 occurred at $t = 1970$. Here, the value was ≈ 0.0552 and ≈ 0.0571 for the Modelica and Julia models, respectively, a difference around 3.5%. To compare, the value reported by the Vensim simulation was ≈ 0.0547. Hence, the difference between the Julia and Vensim model was $\approx 4.29\%$. The reason for this divergence is due to how the Julia Simulation Engine handles a series of hybrid discrete events that occur in 1970. To summarize, the experiment shows a very small divergence from the original model; furthermore, we can observe that the dynamics of the resulting equations are the same. These differences in values between the Modelica and Vensim SD Model are due to differences in the numerical solvers used in the experiment.

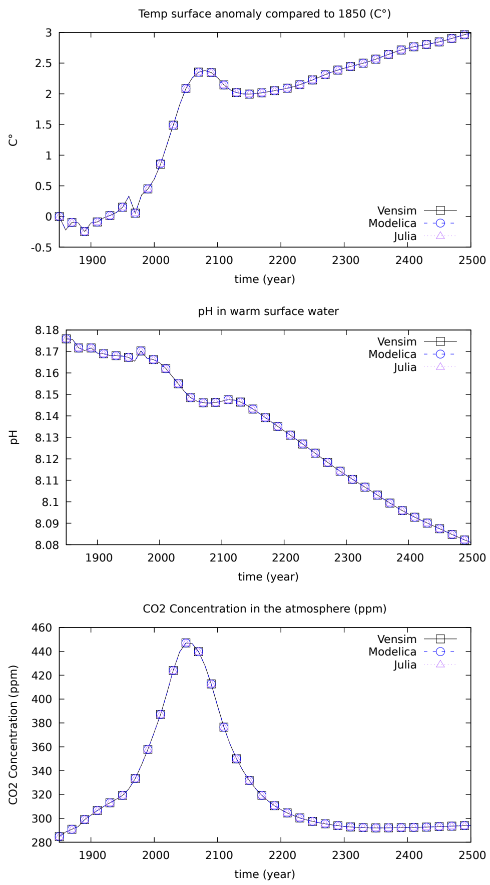

Figure 4. Graphs showing the simulation result of different variables from the year 1850 to the year 2500 for the Vensim, Modelica and Julia.

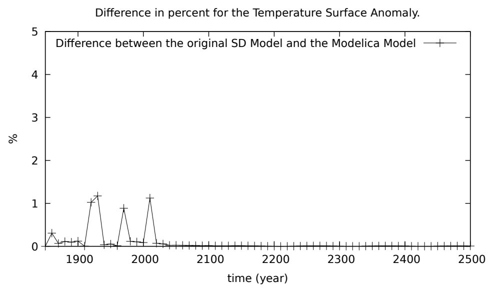

Figure 5. Difference in percent between the original SD Model simulated in Vensim and the Modelica model produced by the translator.

[11] integration.html Accessed 2023-08-21
[12] OpenModelicaUsersGuide/latest/solving.html Accessed 2023-08-21
[13] DiffEqDocs/stable/solvers/ode_solve/ Accessed 2023-08-21

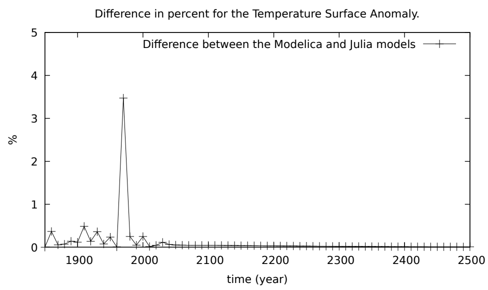

Figure 6. One excerpt for our comparison experiment that shows the difference in percent for the temperature surface anomaly variable between the Modelica and Julia models for each decade between 1850 and 2500.

9 Conclusion and Future Work

Simulation-based assessment of socio-bio-physical systems necessarily involves a wide range of knowledge domains. Different scientific disciplines tend to use different tools and languages to express their models. This leads to an ecosystem of modeling techniques that can undermine the development of more complex systems. In this work, we have presented a translator capable of translating models from one formalism, System Dynamics, to an object-oriented equation-based formalism as defined by the Modelica language. By so doing, we gained the ability to extract models from any tool that can export System Dynamics to XMILE and create a correct and equivalent Modelica and Julia model using a new automatic parser and translator. The existence of standard intermediate representation formats such as XMILE has been fundamental in achieving this goal.

We validated our efforts by comparing the fidelity of the translated model to the original Vensim model. In our experiments, it was shown that the difference between the Vensim and Modelica models was negligible, at most around 1% for the variables that were compared, see Figure 5. The overall dynamics of the translated system remained the same see the graphs in Figure 4. Hence, we can draw the same conclusions as (Randers & Goluke, 2020). As the main goal was to investigate the fidelity of the translated models to the original model, certain aspects, such as accuracy options of the numerical solvers and computational time, were not investigated in detail. However, both the OpenModelica environment and the Julia environment support a wide variety of industrial strength solvers capable of simulating models with more than tens of thousands of equations and variables under controlled accuracy (Ma et al., 2021; Fritzson et al., 2020; Rackauckas & Nie, 2017).

As future work, it remains to increase the number of functions and blocks supported by the translator between XMILE to Modelica, since for the present project the scope was set at what is necessary to meet the needs of the ESCIMO model. Further, it could be interesting to investigate the dynamics of a complete Earth3 model described in Section 5 by replacing the spreadsheet model with a Modelica model and activating the now-disabled feedback loops.

Also, as the model is now available in Julia and Modelica, it would be interesting to examine insights that can be obtained using various powerful tools in the Julia framework, such as scientific machine learning. It could also be interesting to augment the ESCIMO model using the structural variability of equations present in (Tinnerholm et al., 2022) and with other complex models available in the wider Modelica ecosystem. Robust optimization-driven sensitivity analysis could be performed for Earth3 using the OMSens plugin available for the OpenModelica toolkit (Danós et al., 2017).

We hope our work on unifying heterogeneous modeling paradigms will increase interdisciplinary collaboration in science and industry and enable a wider community to gain additional insights into a system as complex as planet Earth.

Acknowledgement

This work has been supported by the Swedish Government in the ELLIIT project and has been partially funded by Vinnova in the context of the ITEA project EMBRACE. It has also been partially funded by the Grant UBACyT 2020 Modalidad II 20620190200002BA from the University of Buenos Aires. Furthermore, the authors gratefully acknowledge the financial support of the French Transmission System Operator RTE to the Open Source Modelica Consortium, which made this initial development possible. Furthermore, we wish to thank the authors of the ESCIMO model, Jørgen Randers, and Ulrich Goluke for providing data from the ESCIMO Model and assisting with our validation effort.

We also acknowledge the comments from the authors[14] concerning the acronym ESCIMO model: ”Please note that the acronym for the software model described in the ESD paper is now recognized to be culturally insensitive and inappropriate. The editors of the journal ESD, the journal owner European Geosciences Union, and the publisher Copernicus Publications foster equality, diversity, and inclusiveness in scientific exchange and do not condone in any way racism, discrimination, or cultural appropriation. The authors did not intend to insult any ethnic groups by using the acronym for this software model.”

[14] Accessed 2023-05-11: http://www.2052.info/escimo/

References

Bauer, C., & Bodendorf, F. (2006). Enhancing system dynamics modeling using a component-based approach. *International Journal of Simulation: Systems, Science & Technology*, *7*(6).

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, *59*(1), 65–98.

Castro, R. (2019). Open research problems: Systems dynamics, complex systems. In B. P. Zeigler, A. Muzy, & E. Kofman (Eds.), *Theory of modeling and simulation* (Third Edition ed., p. 641-658). Academic Press. doi: 10.1016/B978-0-12-813370-5.00035-3

Cellier, F. E. (2008). World3 in modelica: creating system dynamics models in the modelica framework. In *Proceedings of the 6th international modelica conference* (Vol. 2, pp. 393–400).

Danós, A., Braun, W., Fritzson, P., Pop, A., Scolnik, H., & Castro, R. (2017, dec). Towards an OpenModelica-based sensitivity analysis platform including optimization-driven strategies. In *Proceedings of the 8th international workshop on equation-based object-oriented modeling languages and tools* (Vol. 17, pp. 87–93). New York, NY, USA: ACM. doi: 10.1145/3158191.3158206

Forrester, J. W. (2007). System dynamics—the next fifty years. *System Dynamics Review: The Journal of the System Dynamics Society*, *23*(2-3), 359–370.

Fritzson, P. (2015). *Principles of object-oriented modeling and simulation with modelica 3.3: A cyber-physical approach* (2nd ed.). Wiley-IEEE Press.

Fritzson, P., Pop, A., Abdelhak, K., Ashgar, A., Bachmann, B., Braun, W., … Östlund, P. (2020). The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control*, *41*(4), 241–295. doi: 10.4173/mic.2020.4.1

Gomes, C., Thule, C., Broman, D., Larsen, P. G., & Vangheluwe, H. (2018, may). Co-simulation: A survey. *ACM Comput. Surv.*, *51*(3). doi: 10.1145/3179993

Lempinen, T., Ruutu, S., Karhela, T., & Ylén, P. (2011). Open source system dynamics with simantics and openmodelica. *Proceedings of the International System Dynamics Society*.

Ma, Y., Gowda, S., Anantharaman, R., Laughman, C., Shah, V., & Rackauckas, C. (2021). *Modelingtoolkit: A composable graph transformation system for equation-based modeling.*

Meadows, D. H. (2008). *Thinking in systems: A primer*. Chelsea Green Publishing.

Meadows, D. L., Behrens, W. W., Meadows, D. H., Naill, R. F., Randers, J., & Zahn, E. (1974). *Dynamics of growth in a finite world*. Wright-Allen Press Cambridge, MA.

OASIS. (2015). *Xml interchange language for system dynamics (xmile) version 1.0.* Retrieved from `http://docs.oasis-open.org/xmile/xmile/v1.0/os/xmile-v1.0-os.pdf`

Rackauckas, C., & Nie, Q. (2017). Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *The Journal of Open Research Software*, *5*(1). doi: 10.5334/jors.151

Randers, J., & Goluke, U. (2020). An earth system model shows self-sustained thawing of permafrost even if all man-made ghg emissions stop in 2020. *Scientific reports*, *10*(1), 18456.

Randers, J., Golüke, U., Wenstøp, F., & Wenstøp, S. (2016). A user-friendly earth system model of low complexity: The escimo system dynamics model of global warming towards 2100. *Earth System Dynamics*, *7*(4), 831–850.

Randers, J., Rockström, J., Stoknes, P.-E., Goluke, U., Collste, D., Cornell, S. E., & Donges, J. (2019). Achieving the 17 sustainable development goals within 9 planetary boundaries. *Global Sustainability*, *2*, e24.

Tinnerholm, J., Pop, A., & Sjölund, M. (2022). A modular, extensible, and modelica-standard-compliant openmodelica compiler framework in julia supporting structural variability. *Electronics*, *11*(11). doi: 10.3390/electronics11111772

Yeager, L., Fiddaman, T., & Peterson, D. (2014). Entity-based system dynamics. In *Proceedings of the international system dynamics conference, delft.*