Environmental Modelling & Software 94 (2017) 48–62

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Modular and spatially explicit: A novel approach to system dynamics

Patrick Wingo [a, *], Allen Brookes [a], John Bolte [b]

[a] US Environmental Protection Agency, Western Ecology Division, Corvallis, OR, USA
[b] Oregon State University, Corvallis OR, USA

ARTICLE INFO

Article history:
Received 14 November 2016
Received in revised form
10 March 2017
Accepted 11 March 2017

Keywords:
System dynamics
Spatially-explicit
Simulation engine
Open-source
Simile
Vensim
XMILE

ABSTRACT

The Open Modeling Environment (OME) is an open-source System Dynamics (SD) simulation engine which has been created as a joint project between Oregon State University and the US Environmental Protection Agency. It is designed around a modular implementation, and provides a standardized interface for interacting with spatially explicit data while still supporting the standard SD model components. OME can be run as a standalone simulation or as a plugin to a larger simulation framework, and is capable of importing Models from several SD model formats, including Simile model files, Vensim model files, and the XMILE interchange format. While it has been released, OME is still under development, and a number of potential future improvements are discussed. To help illustrate the utility of OME, an example model design process is provided as an Appendix.

1. Introduction: what is OME?

The Open Modeling Environment (OME) is a System Dynamics (SD) model simulation engine that has been developed as a joint project between the department of Biological and Ecological Engineering (BEE) at Oregon State University and the Office of Research and Development of the United States Environmental Protection Agency (USEPA). At its core, OME is an engine for solving SD models using either Euler or 4th-order Runge-Kutta (RK4) integration methods. SD modeling involves characterizing, through the use of ordinary differential equations, the response of a set of system states through time (Lane, 2000). These states can be thought of as discrete buckets holding a specific quantity of something that is pertinent to the question the model is trying to answer, and that describe the state of the system at a given moment. Several common SD modeling tools are Simile (Simulistics Ltd, 2015), Vensim (Ventana Systems, Inc, 2015), and STELLA (isee Systems, 2013). All of these tools allow for the construction of a Stock-Flow diagram, a method of visual model construction that uses a set of standard symbols to describe the processes that drive the simulation (Wingo, 2015). OME does not yet have its own tools for constructing Stock-Flow diagrams, but instead relies on existing tools to create models to be run (Wingo, 2015). A significant goal of OME's development has been to address two shortcomings that are apparent in SD modeling tools commonly used for authoring and running SD models: 1) the absence of an elegant means of representing spatially explicit models and associated datasets, and 2) the insular nature of many extant tools, which creates unnecessary challenges when attempting to couple them with other modeling software (Wingo, 2015).

Many modelers have included explicit spatial relationships in the Stock-Flow diagrams used to define their SD models (Costanza and Voinov, 2004). Traditional approaches to incorporating spatially-explicit data into SD models are tedious and taxing, often requiring the model authors to re-implement well-established spatial relationships specific for each model implementation (Voinov et al., 2004). Typically, such approaches are haphazard solutions that take advantage of a particular modeling environment's implementation of multi-value variables, which are commonly known as lists, subscripts, collections, or submodels (Simulistics Ltd, 2008) (Ventana Systems, Inc) (isee Systems, 2520). None of these approaches to multi-value variables are part of the traditional SD model approach, and are not implemented equally across SD modeling tools (Wingo, 2015). Ideally, incorporating spatial data and relationships into SD models would be

* Corresponding author.
E-mail address: pwingo@gmail.com (P. Wingo).

http://dx.doi.org/10.1016/j.envsoft.2017.03.012
1364-8152/

accomplished using a standardized interface that managed details associated with reading, writing, accessing, and querying spatial information. Such an approach would allow for the use of spatially detailed inputs and outputs without requiring the SD model to be concerned about the specific representation of space used. OME provides such a service through the Spatial Data Provider (Wingo, 2015), discussed in a later section of this paper.

There have been previous attempts to create ways to manage spatially explicit data within the confines of a SD model. The Spatial Modeling Environment (SME) (Costanza and Voinov, 2004) is a tool that would take a model created in the SD modeling tool STELLA, and apply the model once to each cell in a spatially explicit grid (Costanza and Voinov, 2004). Similarly, SimARC (Mazzoleni et al., 2003) represents another attempt to capture spatially explicit data by bridging the simulation engine of the SD modeling tool Simile (Simulistics Ltd, 2015) with the general Geographic Information Systems (GIS) tool ArcMap (Environmental Systems Research Institute, Inc, 2016), allowing the SD model to be applied to each polygon in an ArcMap map layer (Mazzoleni et al., 2003). While both SME and SimARC focus on applying a SD model as a process internal to each discrete spatial unit (be it a grid cell or polygon), OME applies aspects of a specific spatial coverage to the dynamics captured within a single SD model (Wingo, 2015). This approach effectively addresses a different explicit spatial usage case than is covered by SME and SimARC, targeting a different set of models and research questions (Wingo, 2015).

The second shortcoming being addressed originates from the fact that most extant SD modeling tools are largely insular in design (Wingo, 2015). Cross-interaction and integration with other modeling frameworks is challenging, as there is no standard approach to tool integration, even when various SD modeling tools are largely accomplishing their simulations in similar ways (Lane, 2008). While the XMILE standard is attempting to establish a common interchange format for model definitions (OASIS XML, 2015), OME takes cross-interaction one step further by providing an architecture and application programming interface (API) for interacting in real time with other processes, allowing for it to be embedded as a component in other software tools (Wingo, 2015). Additionally, OME's source code is freely available, allowing OME to be extended or modified if the original implementation is insufficient for a given purpose or integration scenario (Wingo, 2015).

1.1. Example model information

In order to demonstrate a number of the features supported by OME, a demonstration model has been created: The Simple Critter Model. A complete description of the model and a walkthrough of how spatially explicit information is incorporated into this model can be found in Appendix B; an abbreviated introduction to the model follows. This model captures the dynamics of a population moving between square sample plates in response to a stimulus. These plates can be assembled in any configuration, but can only act as traversable neighbors across one of the four edges (top, left, bottom, and right). While the configuration of the plates can be hard-coded into a model (see Appendix B), this approach is inflexible and difficult to maintain; the addition, removal, or rearrangement of sample plates would require significant restructuring of the stock flow diagram. Ideally, the dynamics represented by the model could be applied without any knowledge of the explicit spatial configuration, while still being influenced by it.

OME currently provides no visual interface for defining SD models; rather, models are defined in XML-conformant text files which are then read by the OME runtime. To simplify model creation, the Simple Critter Model was created in Simile, and then converted to OME-conformant XML. The base Simile model used in the conversion process is included in Appendix B. The conversion process involves providing a model file from Simile, Vensim (Ventana Systems, Inc, 2015), or any tool that supports XMILE to the OME translation tool UniversalConverter, an executable that is part of the OME package. For more information on the model conversion process, see the “Support for Multiple File Formats” subsection in the “Modular Design” section.

2. Modular design

In order to appeal to the broadest audience possible, OME needs to provide support for a number of usage cases. Experience throughout the course of OME's development has revealed a number of potential usage cases, such as 1) generating and exporting simulation results for later use by an external program, 2) generating results for immediate perusal by a human being, and 3) sharing intermediate results incrementally during a simulation run as a piece of a larger simulation. The architecture of OME is conducive to all of these potential usage cases.

OME is fundamentally a series of interconnected dynamically-linked libraries and executables, with different configurations used to accomplish different tasks (Wingo, 2015). The interconnected nature of the modules is outlined in Fig. 1, and a brief description is provided for each distinct module. This approach to the framework's architecture is useful for ensuring the efficient use of resources by only using parts necessary for the task at hand. This is exemplified by the OMEEngine, OMESimRunner, and <plugin> modules in Fig. 1. OMEEngine is a simple command-line tool for running a simulation and exporting the results, satisfying the first identified usage case. OMESimRunner, by contrast, is intended to not only run a simulation, but also provide an interface for a user to organize and browse simulation results, satisfying the second identified usage case. Both OMEEngine and OMESimRunner provide the ability to export all values from all incremental timesteps in a comma separated values (.csv) file, allowing for further processing of model generated values using external tools. The <plugin> entry represents OME configured as a plugin to a larger simulation (discussed later in this paper), which would satisfy the third identified usage case. OMEEngine and the <plugin> stand-in for external tools do not require any descriptive details about model components, so they do not link against the OMEDraw utility library. OMESimRunner, by contrast, does require access to tasks unrelated to running the simulation, as it graphically displays the results of a simulation through a series of text-based tables, so it links against OMEDraw. This modular construction also provides flexibility for future implementations; if a new usage case were to arise, such as the need to visualize OME models, most of the necessary parts already exist in the OMERuntime and OMEDraw libraries. The user-facing front end would be the only piece that would need to be added to meet the task's demands (Wingo, 2015).

The Simple Critter Model utilizes the OMEEngine executable, the OMERuntime library, and the CSVSpatialDataProvider library modules when executed; see Fig. 1 for a description of each module. The details of how the model will be executed are largely dependent on the contents of the control file, discussed in the next section.

2.1. Specifying a model in OME

OME relies on an XML-based specification for model declarations (.omem files) and control and simulation details (.omec files) (Wingo, 2015). Parameter values can be provided through comma separated values (.csv) files and/or Simile's .spf files (Wingo, 2015). The .omem file provides an intermediate model declaration that