Environmental Modelling & Software 61 (2014) 59–64

Contents lists available at ScienceDirect

Environmental Modelling & Software

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

# A Computational Model Library for publishing model documentation and code

Nathan D. Rollins, C. Michael Barton, Sean Bergin, Marco A. Janssen*, Allen Lee

*School of Human Evolution and Social Change, Arizona State University, PO Box 872402, Tempe, AZ 85287-2402, USA*

ARTICLE INFO

*Article history:*
Received 23 February 2014
Received in revised form
29 June 2014
Accepted 30 June 2014
Available online

*Keywords:*
Model archive
Open source
Computational modeling
Documentation
Agent-based modeling

ABSTRACT

We present a repository for disseminating the computational models associated with publications in the social and life sciences. The number of research projects using computational models has been steadily increasing but the resulting publications often lack model code and documentation which hinders replication, verification of results and accumulation of knowledge. We have developed an open repository, the CoMSES Net Computational Model Library, to address this problem. Submissions to the library can be original models accompanying publications or replications of previous studies. Researchers can request that their models undergo a certification process that verifies that the model code successfully compiles and runs and that it follows documentation best practices. Models that pass the certification process are assigned persistent URLs and identifiers. We present the basic components of our repository, discuss our initial experiences with the library, and elaborate on future steps in the development of this cyberinfrastructure.



## 1. Introduction

Interest in computational modeling of social and biological systems—i.e., algorithmic models (Miller and Page, 2007)—as a tool to understand the complex dynamics of social-ecological systems is growing rapidly with an increasing number of research projects generating models of diverse phenomena (e.g. Alessa et al., 2006; Edmonds and Meyer, 2013; Helbing, 2012). The *Network for Computational Modeling in the Social and Ecological Sciences* (CoMSES Net) Computational Model Library (CML) <http://www.openabm.org> is an open-access public archive, established with support of the US National Science Foundation, where the code and documentation of computational models can be published by researchers. CoMSES Net is a scientific community of practice for sharing knowledge and establishing standards for computational modeling in research (Janssen et al., 2008). Recent commentaries in leading scientific venues make a strong case for including computing within the broader scientific tradition of research transparency (Barnes, 2010; Joppa et al., 2013a,b; Morin et al., 2012; Peng, 2011). This is especially important as computing—and especially modeling—is moving from being an analytical tool to a fundamental aspect of scientific research. To ensure that the scientific community can evaluate and build on the rapid growth of model-based research, it is imperative that adequate descriptions of a model and the model source code itself be publicly available.

A key criterion for transparency in science is the potential for others to replicate a set of research procedures. In order to replicate model-based research, it is necessary to have a sufficiently detailed description of a computational model, its operation, and objectives; the model code in human-readable and compilable (i.e., *source*) format; and an accessible runtime environment (or sufficient information to recreate such an environment) in which to reproduce model-based research procedures.

Ideally a description of the computational model should accompany any publication about model-based research or be referenced in the publication. However, model descriptions vary greatly in the level of detail provided, and there is considerable inconsistency in which details are even included in model descriptions. CoMSES Net and the CML strongly encourage researchers to use standardized formats, like the one developed by Grimm and colleagues (Grimm et al., 2010, 2006; Polhill et al., 2008), to ensure that model descriptions are adequate for potential replication.

Even with detailed, standardized descriptions of models, however, the program code itself is needed to exactly replicate particular operations, especially when novel and unexpected outcomes

* Corresponding author. Tel.: +1 480 727 7067; fax: +1 480 965 7671.
*E-mail addresses:* Nathan.Rollins@asu.edu (N.D. Rollins), Michael.Barton@asu.edu (C.M. Barton), Sean.Bergin@asu.edu (S. Bergin), Marco.Janssen@asu.edu (M.A. Janssen), Allen.Lee@asu.edu (A. Lee).

http://dx.doi.org/10.1016/j.envsoft.2014.06.022
1364-8152/

are claimed. Currently, it is often very difficult to obtain model code, even with helpful model developers. This has been shown in an increasing number of replication studies that were based on the description of the model in the original article (e.g. Hales et al., 2003; Polhill et al., 2005; Wilensky and Rand, 2007; Meadows and Cliff, 2012) or even an available version of the source code (Janssen, 2007; 2009). Most replication studies are able to verify the original results but often uncover contestable assumptions and implementations which challenge the robustness of the conclusions of the original paper (e.g. Hales et al., 2003; Meadows and Cliff, 2012). Access to the original source code also makes it possible to operationalize a model's assumptions, goals, and procedures with different algorithms, computer languages, or programming platforms. These can lead to cumulative improvements in model performance or help explain why different results are obtained in different computing environments.

Computational models are microcosms of dynamic, complex systems which makes it difficult to understand some aspects of models without actually running them. This requires information about the requirements needed to run a model and access to the runtime support infrastructure (i.e., dependent software libraries and/or interpretive environments). The combination of readable code, metadata, and runtime environment are critical for permitting researchers to build on each other's work so that scientific computation can evolve and improve. However, the dissemination of model description and code does not fit neatly into established publication formats.

Descriptions of computational models that are sufficiently detailed for replication can exceed the length of an associated article in word-limited journals, and a printed version of computational model source code can be even longer, making it impractical to publish this vital information in such a manner. Even if provided as the increasingly common online supplemental materials to articles, model source code is useless in most common typeset publishing formats. If not available as a compilable text file, it must be laboriously retyped or reformatted with the potential for mistakes that could alter the original code in undesirable ways. Moreover, most journals still do not permit online submission of usable files that would permit dissemination of the program code, and virtually none provides environments in which scientific code can be evaluated—either during review or after publication. To remedy this situation, Peng (2011) recommends the creation of code repositories, maintained by scientific communities of practice "...to provide a single place to which people in all fields could turn to make their work reproducible." (Peng, 2011, p. 1227). Existing generic source code and version management repositories (i.e., for any program code and not just computational models) include BitBucket (https://bitbucket.org/), GitHub (https://github.com/), and Google Code (https://code.google.com/) but these require the technical expertise to set up a site for each model published, and are not designed with the goal of long-term digital preservation. There are platform-specific repositories for specific modeling software platforms (as often are available for other software like MatLab) such as the NetLogo Modeling Commons (http://modelingcommons.org/) and the Repast Symphony archive (http://sourceforge.net/p/repast/repast-simphony-models). These repositories generally are created to support and promote particular software packages and their user communities. They also are not aimed at long-term preservation and dissemination of scientific modeling code. Finally, there are a number of domain specific efforts to create platforms to share models such as ECOBAS for ecological modeling (Strube et al., 2008), CESM for atmospheric dynamics (Gent et al., 2011) and CSDMS for surface dynamics (Peckham et al., 2012).

As with other aspects of scientific practice, there must be incentives for researchers to dedicate the time and effort required to write detailed model descriptions, providing source code and associated metadata, and ensuring the accessibility of the necessary runtime environment. These may be more effective if they can be embedded within the system of incentives already established in the academic and research world: public recognition by peers through citation and recognition by employers and scholarly organizations as evidence of valuable research activity. Related incentives include requirements by funding agencies and journals to sufficiently document and disseminate model-based research (Morin et al., 2012; Peng, 2011).

## 2. CoMSES Net Computational Model Library

The CoMSES Net Computational Model Library provides a framework to address many of these and related issues relevant to encouraging greater transparency in scientific computing. The CML provides a structured but easily accessible venue for the dissemination of model code, metadata, and associated descriptive documentation in the form of a digital library rather than a source code repository and versioning environment like GitHub. Models are represented as searchable entries in the digital library, much like books are represented in online catalogs, with a title, author(s), and brief description. Model library entries can also display visual previews of a model as an image or short animation. A formatted citation is displayed for each entry so that anyone who uses the model can credit its creator(s) in a familiar way for academia and science. A user can download a model from an entry, similar to downloading a digital manuscript from a library entry. An example of a model entry is shown in Fig. 1; the original entry can be found (and model downloaded) at <http://www.openabm.org/model/3580>. See Supplemental Information for more details about model library entries.

As described in more detail below, peer-reviewed models are assigned permanent digital record identifiers (equivalent to commercial DOI's). Models can be peer-reviewed as part of a journal article review or independently of an article review (see details below). Below, we describe the model library in more detail, and discuss the benefits of publishing models.

## 3. Details of the Computational Model Library

### 3.1. Scope of the CML

A primary aim of the CML is the preservation and dissemination of scientific code for computational models applied in the life and social sciences. By "computational models" we refer to models based on algorithms, such as agent-based models, cellular automata, network models, and discrete event simulation rather than equation-based models. Currently, most models in the library are agent-based models .

Unlike archives and community sites for particular software platforms, models in the CML are not restricted to any specific software platforms or programming languages, in order to make the library open to submissions from a wide scientific audience. Language and platform preferences change over time, of course, and a once-popular language may end up in a niche or abandoned. However, because the human-readable source code and documentation is published in the library, the model itself can remain useful for other scholars even if the platform it was originally designed for falls into disuse. Hence, we encourage the publication in the CML of models written in older, obscure languages so that they have the potential to be replicated in newer, more common languages and frameworks. A consequence of this approach is that although a model contributor must specify the programming platform and version used for the implementation, models in the CML cannot be