Environmental Modelling & Software 55 (2014) 156–163

Contents lists available at ScienceDirect

Environmental Modelling & Software

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

# Standardised and transparent model descriptions for agent-based models: Current status and prospects

Birgit Müller [a,*], Stefano Balbi [b], Carsten M. Buchmann [c], Luís de Sousa [d], Gunnar Dressler [a], Jürgen Groeneveld [a,e], Christian J. Klassert [f], Quang Bao Le [g], James D.A. Millington [h], Henning Nolzen [a], Dawn C. Parker [i], J. Gary Polhill [j], Maja Schlüter [k], Jule Schulze [a], Nina Schwarz [c], Zhanli Sun [l], Patrick Taillandier [m], Hanna Weise [a]

[a] UFZ, Helmholtz Centre for Environmental Research – UFZ, Department of Ecological Modelling, Permoser Str. 15, 04138 Leipzig, Germany
[b] Basque Centre for Climate Change (BC3), Alameda Urquijo 4, 4°, 48008 Bilbao, Spain
[c] UFZ, Helmholtz Centre for Environmental Research – UFZ, Department Computational Landscape Ecology, Permoser Str. 15, 04138 Leipzig, Germany
[d] Resource Centre for Environmental Technologies, Public Research Centre Henri Tudor, Technoport Schlassgoart, 66 rue de Luxembourg, P.O. Box 144, L-4002 Esch-sur-Alzette, Luxembourg
[e] School of Environment, University of Auckland, Private Bag, 92019 Auckland, New Zealand
[f] UFZ, Helmholtz Centre for Environmental Research – UFZ, Department of Bioenergy, Torgauer Str. 116, 04347 Leipzig, Germany
[g] Natural and Social Science Interface (NSSI), Institute for Environmental Decisions (IED), ETH Zurich, Universitaetstrasse 22, CH-8092 Zurich, Switzerland
[h] Department of Geography, King's College London, Strand, London WC2R 2LS, UK
[i] School of Planning, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada
[j] The James Hutton Institute, Craigiebuckler, Aberdeen AB15 8QH, UK
[k] Stockholm Resilience Centre, Stockholm University, Kräftriket 2B, 10691 Stockholm, Sweden
[l] Leibniz Institute of Agricultural Development in Central and Eastern Europe (IAMO), Theodor-Lieser-Str. 2, 06120 Halle (Saale), Germany
[m] Department of Geography, UMR IDEES, CNRS, University of Rouen, 7 rue Thomas Becket, 76130 Mont Saint Aignan, France

ARTICLE INFO

*Article history:*
Received 3 April 2013
Received in revised form
20 January 2014
Accepted 21 January 2014
Available online 11 February 2014

*Keywords:*
Agent-based modelling
Domain specific languages
Graphical representations
Model communication
Model comparison
Model development
Model design
Model replication
Standardised protocols

ABSTRACT

Agent-based models are helpful to investigate complex dynamics in coupled human–natural systems. However, model assessment, model comparison and replication are hampered to a large extent by a lack of transparency and comprehensibility in model descriptions. In this article we address the question of whether an ideal standard for describing models exists. We first suggest a classification for structuring types of model descriptions. Secondly, we differentiate purposes for which model descriptions are important. Thirdly, we review the types of model descriptions and evaluate each on their utility for the purposes. Our evaluation finds that the choice of the appropriate model description type is purpose-dependent and that no single description type alone can fulfil all requirements simultaneously. However, we suggest a minimum standard of model description for good modelling practice, namely the provision of source code and an accessible natural language description, and argue for the development of a common standard.



## 1. Introduction

Agent-based models are argued to be helpful to investigate complex dynamics in coupled human–natural systems (Hare and Deadman, 2004; Liu et al., 2007; Balbi and Giupponi, 2010; Filatova et al., 2013). However, the production of research using agent-based modelling has not been as efficient as it could be up to now. Reasons include that model assessment, replication, and comparison are hampered to a large extent by a lack of

* Corresponding author. Tel.: +49 341 235 1708; fax: +49 341 235 1473.
*E-mail addresses:* birgit.mueller@ufz.de (B. Müller), stefano.balbi@bc3research.org (S. Balbi), carsten.buchmann@ufz.de (C.M. Buchmann), luis.a.de.sousa@gmail.com (L. de Sousa), gunnar.dressler@ufz.de (G. Dressler), juergen.groeneveld@ufz.de (J. Groeneveld), christian.klassert@ufz.de (C.J. Klassert), quang.le@env.ethz.ch (Q.B. Le), james.millington@kcl.ac.uk (J.D.A. Millington), henning.nolzen@ufz.de (H. Nolzen), dcparker@uwaterloo.ca (D.C. Parker), gary.polhill@hutton.ac.uk (J. G. Polhill), maja.schlueter@stockholmresilience.su.se (M. Schlüter), jule.schulze@ufz.de (J. Schulze), nina.schwarz@ufz.de (N. Schwarz), sun@iamo.de (Z. Sun), patrick.taillandier@univ-rouen.fr (P. Taillandier), hanna.weise@ufz.de (H. Weise).

1364-8152/$ – see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.envsoft.2014.01.029

transparency in model descriptions. Further, code developed for one project is rarely reused for other projects, even for closely related research. To overcome these problems, standardised model description protocols, ontologies and graphical representations have been created. The various model description types have been developed to achieve different purposes, including facilitation of in-depth model comprehension, assessment, replication, design and communication.

In this contribution we address the question of whether an ideal standard for describing agent-based models exists. We first present a classification of the prevalent types of model descriptions and give an overview of their different purposes. We then review available model description types, evaluating each on its utility for the different purposes. Finally, we discuss advantages of combining these different types, suggest a minimum standard of model description for good modelling practice and discuss future challenges. Note that we set the focus on providing an adequate description of the model itself and not on the description of model results. Appropriate documentation of the model results is beyond the scope of this paper (but see "Transparent and comprehensive ecological modeling (TRACE) documentation" in Schmolke et al. (2010), pp. 482 which suggests a standard for all parts of the modelling process).

The idea for this article came about at a workshop at the 6th International Congress on Environmental Modelling and Software (iEMSs) 2012 in Leipzig, Germany, and the article reflects the perspectives of the participants, who are members of the integrated social and environmental modelling communities.

## 2. Current state of the art: different types of model descriptions in use

We classify the prevalent types of model descriptions in three categories: natural language descriptions, formal language descriptions and graphics (cf. Fig. 1 for an overview). In the following paragraphs the different description types are briefly outlined:

**Natural language descriptions** present models in everyday language with or without a prescriptive structure. The **prescriptive approach** divides the model description into categories, each explaining a particular part of the model. One example of such an approach is the ODD protocol (cf. Grimm et al., 2010 and its extension to include a description of human decisions in ABMs, ODD + D in Müller et al., 2013). ODD describes the model in a hierarchical way using three main categories: *Overview*, *Design concepts* and *Details* that are themselves subdivided into several subcategories such as (in the case of design concepts) *sensing* or *interaction*. ODD is being widely used for the description of ABMs (for examples see Balbi et al., 2013; Caillault et al., 2013; Marohn et al., 2013; Smajgl and Bohensky, 2013). In contrast, a **non-prescriptive natural language description** puts no constraints regarding content and form of the model description on the author (see exemplary model descriptions in Becu et al., 2003; Deadman et al., 2004). Furthermore, non-prescriptive descriptions can also be used to present the source code in a more intuitive way. Examples are literate programming (cf. Knuth, 1984), documentation generators such as Doxygen or Javadoc that assemble source code comments into a structured document, or, in principle, any form of source code documentation that uses natural language.

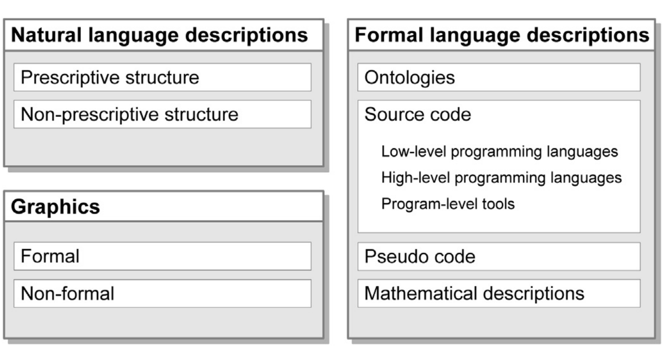


**Fig. 1.** Classification for structuring the prevalent types of model descriptions.

**Formal languages** describe models in an abstract and self-consistent way with formal syntax and semantics that avoid ambiguity. Model descriptions written in formal languages may therefore be used to describe important aspects of a model specifically. Formal languages that we consider here include ontologies, source code, pseudo code and mathematical descriptions.

An **ontology** can be defined as "an explicit specification of a conceptualization" (Gruber, 1993, p. 199) that describes entities and their structural interrelationships, often using a hierarchical categorisation. They specifically allow logical inferences to be drawn. Various formal languages are available for writing ontologies – OWL (Web Ontology Language) being currently the most popularly used (Horrocks et al., 2003; Grau et al., 2008). OWL has been argued to improve the transparency of formal descriptions of model structure in comparison with source code, since the latter is focused on programmer and compilation convenience rather than using logics to reflect common-sense perceptions (Polhill and Gotts, 2009). One example of ontologies applied to agent-based modelling is that of Christley et al. (2004). A second example is the MR POTATOHEAD ontology developed by Parker et al. (2008), which describes the components that appear in agent-based models of land use/cover change. It identifies key model elements and their alternative instantiations, based on a broad review of models. MR POTATOHEAD has an OWL implementation which facilitates evaluating conceptual completeness.

Providing **source code** is another formal way to communicate models. The following subcategories are listed according to their readability, from cryptic to simple-to-read. **Low-level programming languages** (e.g. assembly language) are characterized by their strong linkage to the computer's hardware and are often platform-dependent. Though unlikely to be used for an entire ABM implementation, these can be useful for computationally intensive functions where bespoke code improves on compiler optimisation. Assembly language is necessary where higher-level programming language libraries are not available for specialised hardware operations. For example, it is common in Linux distributions not to provide C libraries for accessing floating point arithmetic utilities stipulated by the IEEE 754 (1985) standard (IEEE, 1985). Polhill and Izquierdo (2005, footnote 2) note that implementation of these utilities using assembly language is necessary in a Cygwin environment.[1]

**High-level programming languages** in their basic form are platform-independent (especially where governed by standards) and improve the readability for the user by providing algorithmic constructs such as loops or conditional statements. Popular examples of high-level programming languages are Java and C++. In addition, **program-level tools** extend the functionality of high-level programming languages by "providing useful software libraries for building specific classes of models" (de Sousa and da Silva, 2011, p. 170) and can further improve the readability of the source code. Usually they are tailored to specific fields of modelling.

[1] The utilities they implemented for this purpose are now available at https://github.com/garypolhill/ieeefp.