Environmental Modelling & Software 103 (2018) 16–28

Contents lists available at ScienceDirect

Environmental Modelling & Software

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

A python framework for multi-agent simulation of networked resource systems

Stephen Knox [a, *], Philipp Meier [b], Jim Yoon [c], Julien J. Harou [a]

[a] University of Manchester, Manchester, UK
[b] Eawag, Department of Surface Waters - Research and Management, Kastanienbaum, Switzerland
[c] Stanford University, Palo Alto, CA, USA

ARTICLE INFO

Article history:
Received 23 November 2017
Accepted 17 January 2018

Keywords:
Modelling framework
Simulation
Python
Open source

ABSTRACT

Modelling managed resource systems can involve the integration of multiple software modules into a single codebase. These modules are often written by non-software specialists, using heterogeneous terminologies and modelling approaches. One approach to model integration is to use a central structure to which each external module connects. This common interface acts as an agreed mode of communication for all contributors. We propose the Python Network Simulation (Pynsim) Framework, an open-source library for building simulation models of networked systems. Pynsim's central structure is a network, but it also supports non-physical entities like organisational hierarchies. We present two case studies using Pynsim which demonstrate how its use can lead to flexible and maintainable simulation models. First is a multi-agent model simulating the hydrologic and human components of Jordan's water system. The second uses a multi-objective evolutionary algorithm to identify the best locations for new run-of-river power plants in Switzerland.

Software availability

Program title: Pynsim
Developer: Stephen Knox
Contact address: stephen.knox@manchester.ac.uk
Software Access: https://www.github.com/umwrg/pynsim
Year first available: 2015
Hardware required: Windows 7, 10 (tested), Linux, MacOSX
Program language: Python
Program size: 116k
Availability and cost: open source
License: GPL3

1. Introduction

The use of simulation to model managed environmental systems is well established (Buytaert et al., 2012; Loucks et al., 2005; Robinson, 2014; Sánchez, 2007; Thorp and Bronson, 2013), as is the need to integrate models from multiple disciplines in order to represent the complexities and interdependencies of environmental systems (Burroughs, 2007; Castilla-Rho et al., 2015; Knapen et al., 2013). Integrated modelling aims to combine models from multiple different disciplines such as ecology & sociology (Daloğlu et al., 2014), water resources & economics (Harou et al., 2009) and water resources & sociology (Barthel et al., 2008).

In addition to the integration of multiple disciplines, some models aim to enable decision making of individual actors within the modelled system. The dynamic interaction and communication between actors within the domain can be modelled using an agent-based approach (Schreinemachers and Berger, 2011). Agent-based modelling is an established methodology to resource modelling (Castilla-Rho et al., 2015; Kelly et al., 2013; Tesfatsion et al., 2017), and several toolkits and languages are available (Hiebeler et al., 1994; Luke et al., 2003; Collier et al., 2003; Tisue and Wilensky, 2004). Multi-Agent Based Simulation (MABS) is a widely used technique, with several examples of cross-disciplinary model integration (Ghazi et al., 2014; Bosse et al., 2013; Daloğlu et al., 2014).

A common approach to model integration is *component-based modelling*, in which processes within an integrated model are represented by pluggable model components. These integrated models are known as Integrated Environmental Models or IEMs. Component-based models are often implemented as an

* Corresponding author.
E-mail address: stephen.knox@manchester.ac.uk (S. Knox).

https://doi.org/10.1016/j.envsoft.2018.01.019
1364-8152/

Environmental Modelling Framework, which are standards or software systems used to build and integrate models around a single coherent structure. EMFs provide abstractions for defining the input and output of models, and the domains on which the models operate.

EMFs differ in the domain they apply to, ranging from the specific (Hill et al., 2004) to the general (David et al., 2013; Bernholdt et al., 2002), and in the degree to which they require the model to be altered (their invasiveness) (Lloyd et al., 2011; Dozier et al., 2016). The decision to integrate a model into an EMF therefore relies on how much the model itself must be modified in order to allow integration and on the value of its inclusion.

The increasing integration of models and the complexity of the software required to support advances in integrated modelling is hampered by the ability of model developers to create sustainable code-bases which are usable beyond the scope of an individual project (David et al., 2013). Development of such code-bases is often difficult to justify as academic projects tend to focus on scientific accuracy and expediency over software sustainability and scalability.

In their vision and roadmap for the future of IEM, Laniak et al. (2013) describe the need for the environmental modelling community to make software development and sharing more prominent, in addition to considering their work to be a part of a larger ecosystem.

One step towards this goal is to encourage the use of existing, open-source technologies which allow model developers to build components, and to publish them through well-established software repositories. This can only be achieved by building a structure for model development and integration which is attractive and simple enough to encourage uptake and involvement from the wider community.

Component-based modelling frameworks rely on a consistent underlying structure to which all components integrate. Some approaches use standards, where each model uses the framework to explicitly define the input and output parameters required to run the model (Gregersen et al., 2007). Others provide a common structure to which all models must comply; for example most water resource management modelling problems can be abstracted to networks of nodes and links (Letcher et al., 2007). Recent work on network modelling software generalises network structures, allowing them to be applied to more than one domain, like water resources, energy and transport and any other domain in which network structures are used (Harou et al., 2010; Knox et al., 2014; Meier et al., 2014). Such an abstraction pushes the responsibility of domain-specific representation and functionality to the model developers themselves, and in doing so can support models spanning multiple disciplines. A generic network representation could allow, for example, multiple networks stored within the same system to be combined by linking models – e.g. the output of a water resource model is fed into the input of an energy management model by specifying a commonality between two different networks (Lee et al., 2007), such as representing a hydroelectric dam which serves as both a water supply reservoir in a water model and a production source in the energy model.

Using a network structure to tie multiple models together also allows for the support of multi-agent behaviour, where the model components can not only act on the network as a whole, but where each node or combination of nodes within the network can execute code independently at each time-step.

The work presented in this paper is referred to as the Python Network Simulation (Pynsim) Framework, a Python package containing abstract classes which are designed to be extended using an object-oriented structure. Pynsim adopts a modular design, allowing multiple users to 'plug in' their code and uses a central network structure to provide a common interface, where each module interacts with the network, not directly with each other. It allows simulations to be performed, where multiple sub-models, adhere to a common representation of a network of nodes and links. Rather than imposing any particular standard for model communication, Pynsim provides a basic structure upon which network-based simulations can be built, laying the groundwork for components of those simulations to be released as public Python libraries.

Pynsim is an object-oriented framework written in Python and attempts to build on the design of existing modelling frameworks. It aims to facilitate model integration, agent-based modelling and the use of a 'component-based' design where components can be added and removed with ease. In addition to incorporating a component-based model integration through the use of 'pluggable' modules, it also supports agent-based modelling by allowing each network element (node, link or institution) to execute code individually at run time. An institution allows Pynsim to represent organisational and other non-physical hierarchies by acting as a container for nodes, links and other institutions.

The contribution of this work is a component-based simulation framework designed specifically for networks, including representation of decision-making hierarchies and support for multi-agent modelling, and which is accessible as a standard Python library. Two case-studies demonstrate how Pynsim's features and associated modelling approach aided development and led to flexible and maintainable human-environment system simulation models.

This paper is structured as follows: Section 2 describes related work in integrated and agent-based modelling. Section 3 identifies features of related frameworks and then introduces Pynsim and its unique functionality. Implementation details of Pynsim are then presented in Section 4. Two case studies are presented in Section 5. Finally a discussion and concluding remarks are presented in Sections 6 and 7.

2. Related work

2.1. Modelling frameworks

Environmental Modelling Frameworks (EMFs) support modular development of integrated models through provision of libraries of core modules and reusable tools for common tasks, such as unit conversion, language interoperability, data manipulation, analysis and visualisation (Argent et al., 2006).

The Object Modelling System (OMS) is an open-source EMF which maintains the design principles of the earlier Modular Modelling System (MMS) (David et al., 2013; Leavesley et al., 2007). OMS has a facility to build simulations, allowing definition of time steps, parameters and models for environmental modelling. OMS is Java-based and offers a lightweight, non-invasive (the models themselves change very little, if at all) framework for integrating models together. This is done by using code annotation to describe the required model parameters and a simple scripting language to connect the models. OMS focusses on reducing the amount of code required to integrate models.

The Open Modelling Interface (OpenMI) (Gregersen et al., 2007) is a standard for the exchange of data between models at runtime (Castronova et al., 2013b). OpenMI is implemented as a set of object-oriented interface classes and supports the integration of models, GUIs and data sources, where each one can be developed independently, and then integrated as 'linkable components' through use of these interfaces. Legacy models can be integrated to OpenMI through a 'wrapper' which runs the model code inside an OpenMI compliant linkable component. This minimises the