Environmental Modelling & Software 94 (2017) 112–126

Contents lists available at ScienceDirect

Environmental Modelling & Software

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

Designing the Distributed Model Integration Framework – DMIF

Getachew F. Belete[*], Alexey Voinov, Javier Morales

University of Twente, ITC, 7500 AE Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 22 November 2016
Received in revised form
28 March 2017
Accepted 2 April 2017

Keywords:
Integrated modeling
Web services
Wrapping
Service oriented architecture
Semantic mediation

ABSTRACT

We describe and discuss the design and prototype of the Distributed Model Integration Framework (DMIF) that links models deployed on different hardware and software platforms. We used distributed computing and service-oriented development approaches to address the different aspects of interoperability. Reusable web service wrappers were developed for technical interoperability models created in NetLogo and GAMS modeling languages. We investigated automated semantic mapping of text-based input-output data and attribute names of components using word overlap semantic matching algorithms and using an openly available lexical database. We also incorporated automated unit conversion in semantic mediation by using openly available ontologies. DMIF helps to avoid significant amount of reinvention by framework developers, and opens up the modeling process for many stakeholders who are not prepared to deal with the technical difficulties associated with installing, configuring, and running various models. As a proof of concept, we implemented our design to integrate several climate-energy-economy models.

Software/data availability

DMIF
Software Developer Getachew F. Belete
Address University of Twente, ITC, 7500 AE Enschede, Netherlands
Tel +31-684-838-692
E-mail getfeleke@gmail.com
First available 2015
Hardware requirements 1 GHz CPU 512 MB RAM
Software requirements Windows 7 Or Newer
Availability Open source
Cost Free
Program language C# and Java
Program size 250 MB
Software Access http://owsgip.itc.utwente.nl/projects/complex/index.php/2-uncategorised/46-model-integration-framework

1. Introduction

Models are simplifications of reality and are developed with the objective to understand a concept or system, to analyze what its future states and trends may look like, and if possible to come up with appropriate management decisions and mitigation or adaptation strategies. Thousands of computer models have been developed. However, complex problems such as climate mitigation require interdisciplinary knowledge and data from many domains including climate, hydrology, energy, economy, land use, behavioral sciences, etc. The complex and interrelated nature of such real-world problems requires holistic system-of-systems thinking (Laniak et al., 2013). In this case it is not practical, perhaps impossible, to construct a single model that could simulate such complex processes (Gijsbers and Gregersen, 2005). Integration of models and tools may be a solution (Stoorvogel, 1995). It also reuses existing models and it is faster and less expensive than reengineering legacy systems (Madni and Sievers, 2014).

During integration, we should understand that component models can be developed using different assumptions and semantics, different methodologies, tools and techniques, may operate at different temporal and spatial scales, may have different levels of complexity, etc. Integration of models assumes linking such heterogeneous models together into an operational model chain (Knapen et al., 2013), or rather a network with loops and feedbacks, where one model down the chain can also feed input back into a model above. This requires addressing interoperability at technical, semantic, and dataset levels (Belete et al., 2017). Based

[*] Corresponding author.
E-mail addresses: getfeleke@gmail.com (G.F. Belete), aavoinov@gmail.com (A. Voinov), j.morales@utwente.nl (J. Morales).

http://dx.doi.org/10.1016/j.envsoft.2017.04.003
1364-8152/© 2017 Elsevier Ltd. All rights reserved.

on this we define a model integration framework as a set of software libraries, classes, and components that enable one to manage technical, semantic, and dataset aspects of interoperability.

Integration of models requires mediation that goes beyond merging information and data that use different schemas. Computer-based models contain sophisticated knowledge statements, which may be represented in different ways. Due to this, integration of models requires understanding of the different contexts of the models involved. It also requires mechanisms to modify the incoming information so that it fits to the assumptions, conditions (rules), and processes in the data-receiving model. Best practice indicates that this can be achieved by providing dedicated components or modules that handle context-based interpretation and semantic mediation. For example, such a mediator component is known as the Knowledge Manager in SEAMLESS (Athanasiadis and Janssen, 2008) or the Semantic Discovery Broker in eHabitat web processing service (Dubois et al., 2013). One of the objectives of modeling is to provide information to decision makers. Despite the large number of available models, decision makers lack easy access to models to evaluate alternative scenarios (Booth et al., 2011). Models that do not require installation of special software, and that do not need special training, are more accessible but rare. With the advance of web technology, presenting models and integration frameworks on the web, i.e. when a model can run in a normal web browser requiring no additional software installation, is becoming more popular since it can significantly increase model accessibility and sharing. However, the volume of data involved, complexity of the software platform required, computational demand for model execution are identified as barriers that prevent sharing and reusing models over the web (Brooking and Hunter, 2013).

Although availing models through web pages is useful, usage of models will still be constrained by the way the web page presents the model. For example, users may want to directly access model output and display it as part of another application. This leads to the idea of providing a “Model as a Service” (Geller and Turner, 2007; Geller and Melton, 2008; Roman et al., 2009). Presenting models as web services has the benefits of making models and their outputs more accessible, easier for model comparison, scalable, and implementable using a variety of approaches (Nativi et al., 2013; Peckham and Goodall, 2013). A web service developed using one programming language can be accessed and consumed by applications using a number of other programming languages, i.e. without requiring intermediary language interoperability tools. The Interoperable nature of services gives the opportunity to integrate legacy models by presenting them as web services (Goodall et al., 2013; Granell et al., 2010).

Currently, one of the main challenges facing the integrated environmental modeling community is lack of interoperability across independently built systems (Goodall et al., 2011; Laniak et al., 2013). This means that besides improving accessibility of models and data, we also need a mechanism for integration across disciplines and models developed using different platforms. There is an increasing need for methodology that enables to build a system-of-systems (Butterfield et al., 2008) by connecting independent component systems and making them interoperable.

The context in which a certain model is used can vary significantly. Besides, a model can be linked to a number of models in different ways. Users have their own integration requirements. Integration scenarios identified and designed by a certain group of modelers or developers may not satisfy integration requirements of the whole user community. Even one particular user can come up with a number of integration requirements. On the other hand, only a small subset of end users may have the programming skills needed to modify the source code of integration frameworks in accordance to their requirements. Due to this, there is a need for tools in which users can select certain models and link them without the need for additional design, coding, debugging, etc.

In this paper, we present the design methodology of a Distributed Model Integration Framework (DMIF). We present the approach used to convert heterogeneous and independently built models into plug-and-play components, and the mechanisms used to automate some of semantic mediation tasks. In addition, we present a case study in semantic mediation using semantic matching algorithms and lexical databases. We also introduce interfaces that enable runtime access and integration of web service based models without the need of additional coding. We also discuss the limitations of such approach. After all, models are always built for a purpose and there is no guarantee that when we reuse a model we will be using it in the same way as intended by the original model construction. This is how so called ‘integronsters’ (Voinov and Shugart, 2013) are created, which we certainly want to avoid. User mediation and pre-integration assessment are the only ways that we can safeguard ourselves from unintended misuse of models. We explore how integration interfaces and semantic mediation can assist in such user guided model evaluation.

The remainder of the paper is organized as follows: Section 2 presents the design criteria of model integration frameworks. Section 3 describes the architecture of DMIF. Section 4 provides information on how we provided for technical interoperability by presenting models as web services. Section 5 describes the methodology for building the semantic mediation module of model integration frameworks. This section also presents algorithms for semantic matching of text-based input-output data and for searching of components using attribute names of components. Section 6 introduces runtime access and integration of web service based models at the GUI level. Section 7 discusses additional issues that we should consider in developing integration frameworks, and we give our conclusions in Section 8.

2. Design criteria for DMIF

Our aim is to provide a verifiable design of a model integration framework that helps to link multidisciplinary heterogeneous models distributed over various software and hardware platforms in a meaningful way. Design should follow user requirements, and it should be verifiable against those user requirements. There are many factors that a model integration framework should consider (Belete and Voinov, 2014, 2016; Belete et al., 2014, 2017). There is no ideal integration technique, which best suites all kinds of model integration requirements. However, as a guideline we know that an integration framework should consider the following design criteria. It should:

- support models developed using different programming languages,
- include models hosted on different hardware and software platforms,
- access models located anywhere on the Internet,
- keep independently developed models autonomous,
- avoid reinventing whenever reuse of resources is possible,
- provide functionalities as reusable components,
- be extensible without disturbing the existing system,
- be accessible on the web.

Given these criteria, our design focuses on interoperability of heterogeneous models. To realize this we need to establish a few well-known dependencies (Rosen et al., 2008) among such models that enable them to exchange data and to collaborate.