Environmental Modelling & Software 80 (2016) 265–280

Contents lists available at ScienceDirect

Environmental Modelling & Software

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

A minimally invasive model data passing interface for integrating legacy environmental system models

A.Q. Dozier*, O. David, M. Arabi, W. Lloyd, Y. Zhang

Colorado State University, Fort Collins, CO, USA

ARTICLE INFO

Article history:
Received 19 February 2015
Received in revised form
31 December 2015
Accepted 26 February 2016
Available online 22 March 2016

Keywords:
Framework invasiveness
Integrated assessment and modeling
Integrated environmental modeling
Inter-process communication

ABSTRACT

This paper presents the Model Data Passing Interface (MODPI). The approach provides fine-grained, multidirectional feedbacks between legacy environmental system models through read and write access to relevant model data during simulation using a bidirectional, event-based, publish-subscribe system with a message broker. MODPI only requires commented directives in the original code and an XML linkage file with an optional custom data conversion module. Automated code generation, compilation, and execution reduce the programming burden on the modeler. Case study results indicated that MODPI required less code modifications within each model code base both before and after automated code generation, outperforming a baseline subroutine approach. Performance overhead for MODPI was minimal for the use case, offering speedup in some cases through parallel execution. MODPI is much less invasive than other techniques, potentially encouraging adoption by the modeling community in addition to maintainability and reusability of integrated model code.

Data and Software Availability

Software developed for the purposes of this paper are open-source and publicly available. The implementation of MODPI presented in the paper is found at https://bitbucket.org/adozier/fortmodpi (2.1 MB), and the implementation of events is found at https://bitbucket.org/adozier/fortevents (11 KB). A summary of the packages is found at https://www.erams.com/resources/Platform/MaaS/Model_Integration. This software requires an implementation of the Message Passing Interface (MPI) or ZeroMQ. Models (3 MB), performance test results (370 kB), a shell script that reproduces the results (2 kB) in this paper are found at https://erams.com/resources/Platform/MaaS/Model_Integration.

1. Introduction

Model integration frameworks, or environmental modeling frameworks, allow a plug-and-play methodology in connecting submodels of an environmental system to enhance model representation of the overall system. However, framework invasiveness restricts reuse and maintenance of framework-dependent models (Donatelli and Rizzoli, 2008; Lloyd et al., 2011). Thus, instead of using the frameworks, many environmental system model developers incorporate transplanted, and often outdated, submodels of other disciplines into their codes (Laniak et al., 2013). Although modeling frameworks have been used to integrate such models across disciplines, model developers often maintain the model code base separate from its support of the framework.

Lloyd et al. (2011) defines framework invasiveness as “the quantity of dependencies between model code and a modeling framework”. Modeling frameworks that aim at minimizing imposed dependencies on a legacy model improve maintainability and reuse (Lloyd et al., 2011). Historically, modeling frameworks that attempt to be less invasive have focused on coarse-grained interaction between “components” or submodels of a larger model (Donatelli and Rizzoli, 2008; Lloyd et al., 2011). Finer-grained feedback schemes that exchange data from within a component or submodel have previously taken more invasive approaches or require extensive computer programming expertise (Becker and Schuttrumpf, 2011). We argue that the amount of work required within a legacy model for integration with another model or support of a framework interface is another obstacle to framework adoption. Thus, we define invasiveness here to be the dependencies

* Corresponding author.
E-mail address: andre.dozier@rams.colostate.edu (A.Q. Dozier).

http://dx.doi.org/10.1016/j.envsoft.2016.02.031
1364-8152/

within each model on either the integration platform or other models, and the amount of work required within each model for integration or implementation of a framework interface.

Attempts to incorporate multidirectional feedback between legacy models include iterative, subroutine, and inter-process communication approaches that are discussed in detail in the next section. To remain minimally invasive, inter-process communication techniques are the most promising approaches as demonstrated by Becker and Schuttrumpf (2011) in making a closed-source model compliant with the OpenMI standard (Moore and Tindall, 2005) within a timestep loop. Inter-process communication techniques have previously required too much programming knowledge for most modelers. Thus, there still remains a need for a minimally invasive, fine-grained, generic model integration interface that does not require such extensive programming expertise (Laniak et al., 2013).

The goal of developing the Model Data Passing Interface (MODPI) is to facilitate and abstract the legacy model integration process to reduce framework invasiveness while minimizing programming knowledge requirements. Objectives of this study include:

1. Develop an abstracted interface for minimally invasive model integration that simplifies complex interactions between legacy models and modeling platforms of disparate disciplines,
2. Automate code generation of MODPI-compatible wrappers for legacy models to support ease-of-use, and
3. Evaluate invasiveness and performance of MODPI as compared to other approaches.

To accomplish these objectives, a publish-subscribe concept is combined with inter-process communication to provide external processes read and write access to any relevant state variable within a legacy model during its execution. A framework is built that automates wrapper generation, and a case study serves to benchmark MODPI against another common approach to the same problem.

2. Background and related work

Many researchers have previously addressed specific model integration challenges, and some have even developed generic interfaces for model integration. However, no generic interface exists for fine-grained, multidirectional feedbacks that preserves the individuality and maintainability of legacy models. Implementing interfaces for existing frameworks requires significant work within the model, and often requires addition of code dependencies on the framework.

This section identifies previous studies that have integrated legacy models to support fine-grained, multidirectional feedbacks, which is the primary functional requirement for the integration studies we summarize here. Fine-grained feedbacks refer to linkages of internal (and relevant) data or calculations between multiple models that cannot be represented by one model as a whole, but are required to represent a particular process more accurately. Multidirectional feedbacks refer to data or calculations within one model that depend on another model and vice versa. When the need for such feedbacks between models arises, there are various implementation considerations such as 1) implicit versus explicit numerical solution techniques, 2) passing data via subroutines or put/get calls, 3) hardware mechanism for communication, and 4) single or multiple executable approaches (Valcke et al., 2012). The following framework design targets for MODPI are used to qualitatively assess the different approaches:

1. Minimally invasive
2. Minimal interface requirements
3. Interoperable across languages and platforms
4. Links closed-source models
5. Reconciles data structure differences
6. Performance overhead is minimal

Interoperability and data structure reconciliation are functional requirements for specific model integration tasks, which also may be true for linking closed-source models and performance overhead in some cases, but not for a generic model integration interface.

Framework design targets are prioritized to make the interface more acceptable to a diverse modeling community that individually maintains or uses large legacy models. We argue that the first two targets, invasive changes within a model code base and difficult or extensive interface requirements, represent the largest factors that inhibit maintenance and reuse of integrated modeling systems (Lloyd et al., 2011). The design target for minimal interface requirements is aimed at reducing the amount of programming work and knowledge required to be able to implement MODPI for a model. Since modelers are often limited by an unfamiliarity with advanced programming techniques to improve interoperability or link closed-source models, minimizing difficult code changes and refactoring requirements may provide a path to encourage adoption and reuse of model integration frameworks. Ensuring interoperability of languages, platforms, and linkages with closed-source models would also broaden the applicability of an integration platform within an increasingly diverse community of modelers (Laniak et al., 2013).

2.1. Implicit versus explicit approaches

Both implicit and explicit solution approaches have advantages and disadvantages. Although solving equations explicitly may intuitively seem numerically faster, implicit approaches may utilize assumptions to solve much more efficiently without sacrificing too much accuracy (Balaji, 2012). Within hydrology, several approaches based on successive approximations allow models to be run separately, maintaining model individuality in partial fulfillment of Target 1 (Fredericks et al., 1998; Ibanez et al., 2014). Lagrangian relaxation techniques are systematic implicit numerical methods that allow for parallel execution of submodels Dozier (2012). Because implicit approaches utilize original forms of equations, model individuality may be more easily attained than explicit approaches addressing Targets 1, 3, 4 and 5. However, most explicit solutions can improve geophysical model integration through guaranteeing numerical solutions for feasible inputs (Dozier, 2012; Balaji, 2012).

2.2. Subroutines versus producer-consumer approaches

In model integration, data can be passed through subroutine arguments or through an exchanging mechanism such as a buffer that handles producers and consumers through put/get calls (i.e., publishers and subscribers). These are distinguished from hardware communication mechanisms because both subroutine arguments and buffers could potentially utilize memory, hard disk, or network communications, although there are typical implementations.

Implementing a subroutine approach often entails decomposing submodels into smaller components: initialization, run or update, and finalization (Argent, 2004). For example, the Basic Model Interface (BMI) specifies initialize and finalize methods in addition to an update function that is used to advance a model or component