Integrating Models for Complex Planning Policy Analysis: Challenges and a Solution in Coupling Dissimilar Models

CUPUM 2015 Submission 208

Abstract

It is an expensive and time consuming task to develop a new model. Besides, a single model often cannot provide answers required for integrated decision making. Therefore, coupling existing models is often used for model integration. The paper provides an overview of possible model integration approaches, briefly explains the models of a particular application and focuses on the integration methods applied in this research. While the initial attempt was to integrate all models as tightly as possible, the authors developed a much more agile integration approach. Python wrappers were developed to loosely couple land-use, transportation and emission models developed in different environments. ArcGIS Model Builder was used to provide a graphical user interface and to present the models' workflow. The suggested approach is efficient when the models are developed in different programming languages, their source codes are not available or the licensing restrictions make other coupling approaches infeasible.

Keywords: model, integration, coupling, Python, CUBE

Introduction

Policy makers are facing challenges of managing multifaceted urban and environmental systems influenced by global factors such as population growth, migration, recession, climate change as well as by local actors such as parties or companies who direct the development according to their own interest (Geertman and Stillwell 2009). Confronted with such complexity, decision makers need adequate tools to better understand and evaluate the effects of policy interventions in urban regions. Such pressure already led to the development of numerous models covering different discipline-specific areas. However, the interconnected character of human and natural systems requires an integrated approach in both decision making and modeling. Integration may have a horizontal and vertical dimension (Moeckel et al. 2015). The horizontal dimension commonly refers to the integration of various domains (such as environment, land use, transportation, etc.), whereas the vertical integration refers to various geographic layers that need to be integrated.

Nevertheless, model integration faces some scientific and technical challenges (van Delden et al. 2011). The main scientific challenges relate to dealing with different domains, paradigms, assumptions, scales, spatial and temporal resolutions used by individual models. Technical challenges include implementing software integration of the models, providing dynamic feedback loops, and developing user interfaces (Lam et al. 2004). Moreover, as Voinov and Shugart (2013) emphasize - treating models only as software in solving the integration challenge may result in perfectly valid software products, which however, are useless as models.

The presented model coupling approach has been implemented by the National Center for Smart Growth (NCSG) at University of Maryland funded by the National Science Foundation program SESYNC (National Socio-Environmental Synthesis Center). This work was built on existing models and institutional relationships between the NCSG and the USGS Eastern Geographic Science Center (EGSC) to form an integrated modeling system. Various models were coupled, including transportation, land-use, land cover and environmental impact models. The task was complicated

because all those models have been developed independently without any built-in method for linking to other models. Moreover, they have been developed in different programming languages, software environments and have various licensing restrictions, making their integration a challenging task.

The authors explored diverse coupling approaches and evaluated their applicability for the project needs. Several couplers were evaluated against the properties identified to be needed to satisfy the specific modeling requirements. However, the existing couplers have been developed with different objectives and constraints in mind. None of the couplers identified by the authors were suitable for the defined integration task. Therefore, an alternative approach of loose model coupling was applied. The suggested approach is especially efficient when the models are developed in different programming languages, their source codes are not available or the licensing restrictions are making other coupling approaches infeasible. It offers a viable solution to coupling such models without the need of change or even access to the sources codes.

Methodology

Individual Models

Maryland Statewide Transportation Model (MSTM)

The MSTM is an advanced trip-based model developed in 2008 by the Maryland State Highway Administration in conjunction with the NCSG and Parsons Brinkerhoff (FHWA 2014). It was designed to estimate the impacts of transportation investments, changes to land-use development, and impacts from factors beyond state boundaries, particularly freight.

Input data include population and employment by model zones, highway and transit networks, and data on travel behavior. The model outputs report traffic impacts on the overall system, corridors or individual links.

Simple Integrated Land Use Orchestrator (SILO)

Initially developed as a research project by Parsons Brinckerhoff for Minneapolis/St. Paul (Moeckel 2011), SILO has been implemented for the state of Maryland. It micro-simulates household relocation, demographic changes and developers who add, upgrade or demolish dwellings. SILO is designed as a discrete choice model. Thus, every household, person and dwelling is treated as an individual object. Spatial decisions, such as relocation, development of new dwellings, etc., are modeled with Logit models (McFadden 1978). Other decisions, such as getting married, giving birth to a child, etc., are modeled by Markov models that apply transition probabilities.

SILO uses the Public Use Microdata Sample to create individual households and their dwellings. The MSTM provides the zone-to-zone travel time by auto and transit. SILO generates a synthetic population with households, persons, dwellings and jobs for the base year 2000 and incrementally updates these dataset in one-year increments through 2040. Every year the MSTM runs, SILO provides updated socio-demographic data by zone.

Mobile Emissions Model (MEM)

The MEM is developed at NCSG and estimates transportation emissions by applying emission of the MOVES2010 EPA[1] model to MSTM-generated traffic flows (Welch 2013).

[1] http://www.epa.gov/otaq/models/moves/

The MEM input data includes road network, vehicle trips, temperatures by month and hour for each county in the study area, humidity, average speed distribution, the vehicle miles of travel on varied road types, fuel formulation and supply. MEM runs every time the MSTM has run.

Building Energy Consumption and Emissions Model (BEM)

The BEM estimates CO_2 emissions and energy consumption from the built environment within Maryland (Welch 2013). It is developed at NCSG and uses the building, location and climate variables of each property to determine whether the structure is likely to combust fossil fuels on site. If the probability is greater than 50%, then the model calculates CO_2 emissions from local combustion based on a set of related multipliers derived from a regression of the national building energy consumption survey data.

SILO provides the building stock for BEM. CO_2 emissions and energy consumption are the primary outputs of the model.

Chesapeake Bay Land Change Model (CBLCM)

The CBLCM was developed by the USGS within the Chesapeake Bay Program. It uses a stochastic methodology to emulate residential urban land use development in Maryland over a series of pre-defined time segments. It is as an independent cellular automata model that translates exogenous county-level projections of population and employment to estimates of urban land demand and then spatially allocates that onto 30m-resolution raster cells. The locations of future growth are informed by data on protected lands, zoning, slopes, land cover, proximity to urban centers, and proximity to locations of recent job and housing growth.

The model calculates a probability surface for growth locations, and allocates households and dwellings provided by SILO. CBLCM generates fine-grained patterns of residential urban growth across the study area.

Data Exchange

One- or two-way data flows are implemented between the models described above. For example, MSTM provides travel times and auto-operating costs to SILO and number of trips and average speed distribution to MEM models; while SILO provides population, employment and auto availability data to MSTM, building data to BEM and population, employment and accessibility to CBLCM. Currently MEM and BEM models are just users of output data from MSTM and SILO, and their output is not used by any other models. However, there are many other variables resulting from those models, which can be potentially used by other models.

At present, data exchange between those models is rather slow because the output from one model is written to a hard drive, that model shuts down, the other model starts and reads the data from the hard drive. In some cases, this form of data exchange is time consuming and may limit the intervals of data exchange to few simulation periods only. Though technically those data exchanges could be done for every simulation year, presently it is implemented only for selected simulation years. For example, auto and transit travel times are obtained from the MSTM for the year 2007 and used as constant values in SILO for the years 2007-2030. In the opposite direction, SILO writes out population and employment for 2030 and feeds those back to MSTM in 2030, and so on. Thus, while SILO runs internally in one-year increments, MSTM, MEM and BEM are run only for a few specific time points.

Table 1 Main characteristics of the used models

Model	Environ-ment	Operation System	Licensing	Simulation Period	Sim. years	Runtime
MSTM	CUBE	Windows	Scripts: Open source CUBE: CitiLabs	2007 or 2030	1	15-16 hour[a]
SILO	Java	Multi-platform	Open source	2007-2030	23	4-5 hour[a]
MEM	CUBE	Windows	USGS, CitiLabs	2007 or 2030	1	< 30 min[a]
BEM	Excel	Multi-platform	n/a	2007 or 2030	1	< 1 min[a]
CBLCM	C / C++	CentOS	USGS	2007-2030	4	3 hour[b]

[a] *20 x AMD Opteron Processor 6328 @ 3.20GHz, 42GB RAM, Windows 7 Virtual Machine*

[b] *2 x 2.56 GHz CPU's, 24GB RAM, Centos 6 Virtual Machine*

More frequent (e.g. annual) data exchange between SILO and MSTM is prohibited by the long model runtime of the MSTM (Table 1). Currently, running the MSTM for one time point takes about 16 hours. The implementation of full feedback between the models during each simulation step may require tighter integration. However, in addition to the difficulties with modifying model sources codes, language interoperability and licensing restrictions, tighter integration may also result in a loss of performance measures such as speed, accuracy, or stability (Peckham et al. 2013). As a result, limited data exchange frequency is weighted against the advantage of supporting different types of components and linking them under a single user interface without changing their source codes. In this work, models developed in Java, CUBE script, Excel and Python have been coupled, while the coupling with a C++ based model is under development.

Practical Value and Key Requirements of Integration

Integrating described models means to better represent complex interactions observed empirically between land-use, transportation, environment and economy in the Baltimore/Washington region. Multiple agencies at the US federal, state, and local level have an interest in linking such models. This includes the US Environmental Protection Agency, the Department of the Interior, and the Maryland Departments of Transportation, Environment, Natural Resources and Planning. Whereas from a scientific perspective, this work aims to improve our understanding of human activity and environmental linkages and to enable improved policy development dealing with environmental sustainability.

Taking into consideration overall goals of this research and the models involved, the following key requirements have been identified for the integrated modeling suite:

- Ability to develop models independently, such that they may be plugged-in easily.
- A modular approach supporting reusability and adding new components.
- Minimal or no change in source codes of the models.
- Capacity to link models developed in different programming languages and environments.
- Ability to deal with different licensing requirements.
- Minimizing manual data transfer.
- User friendly graphical interface.
- Compatibility with GIS for easy data visualization and spatial analysis.
- Adequate running time.

- Minimal costs and efficient timing for implementation.

Model Coupling Approaches

For the coupling of environmental models, Brandmeyer and Karimi (2000) develop a five-level coupling hierarchy, which includes manual data transfer, loose coupling, shared coupling, joined coupling and tool coupling.

The *manual data transfer* method is the most basic level of model coupling, which includes manual extraction, transfer and conversion of output produced by one model to be used as an input for other models. Though this approach requires minimal initial cost and time to apply, it is not attractive when multiple runs and frequent data exchange are required (Brandmeyer and Karimi 2000).

The data exchange between models is automated in *loose coupling*. Models, though, still work independently and the user interacts with each model separately (Wong et al. 2009). Loose coupling also has low initial cost, requires minimal changes to existing codes, and the models still can be developed independently. However, if data structures change in any of the linked models, attention needs to be paid to data conversion.

In *shared coupling,* the models either share the user interface or the data storage. For the first approach, a single user-friendly interface hides the internal coupling method making it less confusing (Berry et al. 1997). In data coupling, the models are kept separate, but share the data storage (van Walsum and Veldhuizen 2011). Shared user interface coupling supports proprietary models and reduces user interaction time. But a user interface update is required in case of model updates. Data coupling makes data maintenance simpler. However, the overall model interface and performance depend on the Database Management System used (Brandmeyer and Karimi 2000).

Joined coupling employs both the common user interface and data storage and may use two structurally different approaches: embedded coupling, when one model contains another (Liu et al. 2014); and integrated coupling, when each model is a peer of every other model (Sudicky et al. 2003). Joined coupling reduce the development costs and promotes code reusability. But it requires access to the models' source codes and a single operating system (OS).

In case of *tool coupling*, the models are coupled using a modeling framework (Babendreier and Castleton 2005, Moore and Tindall 2005). This supports community model development and can be used with both legacy and new models. Though it has higher initial cost due to framework design and development, there are a few such tools developed, such as the Open Modelling Interface (OpenMI) (Gregersen et al. 2007), Model Coupling Toolkit (MCT) (Warner et al. 2008), Community Surface Dynamics Modeling System (CSDMS) (Overeem et al. 2013), Earth System Modeling Framework (ESMF) (Hill et al. 2004), Framework for Risk Analysis Multimedia Environmental Systems (FRAMES) (Shah et al. 2004), PCRaster (Schmitz et al. 2009), O-PALM (Piacentini et al. 2011), OASIS (Valcke 2013) and ICMS (Rahman et al. 2004). However, these tools have their specific requirements on OS, programming languages, data format, access to the source code, licenses, and so on. They often demand changes or rewriting the model codes requiring programming and data/language interoperability expertise.

The order of running the models and the data feedback frequencies are also effecting on coupling choices. Thus, the 'sequential' coupling scheme provides the weakest form of the integration, when the first model runs the required time step/period and provides the output to the second model, which only runs after getting the results of the first model (van Walsum and Veldhuizen 2011). This scheme is often used for manual data transfer or loose model coupling. A drawback of such an

approach is that the stability between the two linked models is determined by the model that gets updated first, which can lead to inconsistencies for the second model. In contrast, the 'fully coupled' scheme supports the full feedback between models within each time step. Though to organize such feedback, fully coupling may require code modification. Moreover, it may result in iterations within iterations and increase its computational load essentially, reducing overall efficiency (van Walsum and Veldhuizen 2011).

Each of the described approaches has its advantages and disadvantages (Brandmeyer and Karimi 2000); and the selection of the method mainly depends on the model requirements, project goals and available resources.

Model Integration through GIS

Considerable efforts have been made to use GIS as an integration tool for different environmental models including soil erosion (Brazier et al. 2005), land-use (Clarke and Gaydos 1998), hydrologic (Devantier and Feldman 1993), water quality (León et al. 2002), pollution and watershed (Basnyat et al. 2000) and other models. In most of such cases, GIS is used for loose coupling of the models, implementing data exchange and visualization. GIS helps to overcome some of the problems related to data interoperability (Goodchild et al. 1997). Moreover, availability of flexible scripting languages in modern GIS packages allow to develop of interactive user interfaces within GIS under which the models can be linked (Tao et al. 1996). Particularly, in this study, the ArcGIS Model Builder was used as a programming environment for Python wrappers linking various independent models and allowing us to capitalize on data management and visualization functionality of ArcGIS package.

Python

Python is an open source object-oriented programming language balancing high-level programming with low-level optimization. Though Python programs usually run slower than FORTRAN, Java or C/C++ programs, they have cleaner syntax and require less time to develop.

From the model integration perspective, Python has specific libraries supporting scientific programming (SciPy), modeling and data analysis (Pandas), visualizations and parallel computing (IPython). Another advantage is its language interoperability often used to glue other programming languages. Thus, Python has libraries supporting function calls from MatLab (MLabWrap), R (RPy), Excel (OpenPyxl), FORTRAN (F2PY, PyFort), Delphi (Python4Delphi), Java (Jyton, JPype, Jepp), Perl (PyPerl), PHP (PiP), C/C++ (Ctypes, Cython, SWIG), etc. Moreover, Python runs natively on Windows, Mac and Linux operation systems. Thus, Python can facilitate interoperating modules implemented in other programming languages (Roberts et al. 2010) and has been successfully used to link such models (Schmitz et al. 2009).

Model Integration Results

As described above, in this application the models are developed in different programming environments; they have various licensing requirements and run sequentially, which makes loose coupling method the most relevant approach.

Wrappers were developed in Python for each of the models, which then were integrated in the ArcGIS Model Builder environment. The process flow diagram paradigm provided by the Model Builder was used to present the models' workflow and linkages. The structures of the wrappers are similar, with a different number of input and output files and parameter sets. Fig. 1 shows a sample Python code used to run the transportation model from ArcGIS.

```
#*********************************************************************
# MSTM model wrapper
# Arguments:
#  0 - MSTM Model executable file
#  1 - Scenario
#  2 - Data exchange folder
#  3 - Output file 1: sovTimePk.csv
#  4 - Output file 2: walkToTransitTimePk.csv
# Created by: Harutyun Shahumyan
#*********************************************************************

# Standard error handling
try:

    import arcpy
    import time
    import os
    import string

    start = time.time()
    arcpy.AddMessage("MSTM model start time: %s" % time.strftime('%X %x %Z'))

    # Get input arguments
    in_Program = arcpy.GetParameterAsText(0)
    in_Scenario = arcpy.GetParameterAsText(1)
    in_ExchangeFolder = arcpy.GetParameterAsText(2)

    # MSTM run parameters
    runParam=in_Scenario+" 6 50 20"
    runCommand = in_Program+" "+runParam

    # Check that the program exist
    if not arcpy.Exists(in_Program):
       raise Exception, "Input program does not exist"

    # Run the model
    arcpy.AddMessage("")
    arcpy.AddMessage("Running %s" % (runCommand))

    desc = arcpy.Describe(in_Program)
    sourceFilePath = desc.path
    os.chdir(sourceFilePath)
    os.system(runCommand)

    # Export shared files
    desc = arcpy.Describe(in_Program)
    sourceFilePath = desc.path + "\\"+ in_Scenario + "\\JavaModule\\input\\"
    file2SILO1=sourceFilePath+"sovTimePk.csv"
    file2SILO2=sourceFilePath+"walkToTransitTimePk.csv"
    arcpy.AddMessage("Exporting exchange data files:")
    arcpy.AddMessage(file2SILO1)
    arcpy.AddMessage(file2SILO2)
    arcpy.SetParameter(3, file2SILO1)
    arcpy.SetParameter(4, file2SILO2)

    elapsed = (time.time() - start)

    arcpy.AddMessage("")
    arcpy.AddMessage("Model end time: %s" % time.strftime('%X %x %Z'))
    arcpy.AddMessage("Processing time in seconds is %s" % str(elapsed ))


except Exception, errMsg:

    if arcpy.GetMessages(2):
        arcpy.AddError(arcpy.GetMessages(2))
    else:
        arcpy.AddError(str(errMsg))
```

Fig. 1 Python wrapper for the transportation model

ESRI Arcpy library was used for getting and setting the model parameters as well as for displaying status messages in ArcGIS geo-processing window. However, those functions can be replaced with standard Python functions if the wrapper shall run without ArcGIS.

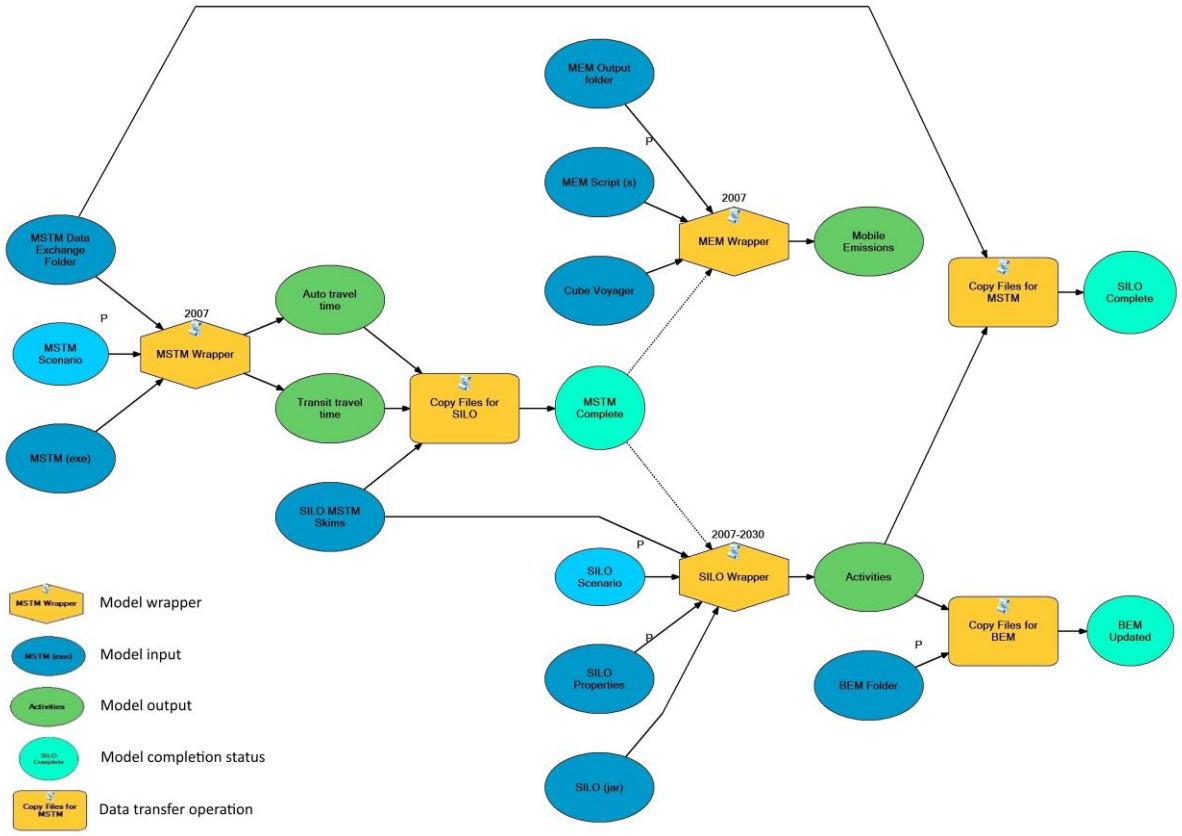


Fig. 2 ArcGIS geo-processing model organizing the models simulation workflow

The `os.system()` standard function was used to call the actual models from the wrapper. In some cases, access to a model source code may be restricted or changing it is not desirable. To avoid any problems with the file paths, the current path is set to the folder containing the model using the `os.chdir()` command. The exchangeable output files are set as wrapper output parameters and can be used by other models.

The geo-processing workflow model developed in ArcGIS Model Builder consists of a set of coupled modules (Fig. 2). Solid lines represent data flow directions between models, while the dotted lines represent the preconditions to run the modules. The direction of the links and the preconditions determine a suitable order for model execution. Execution begins with models that have no incoming links or preconditions and proceeds to models whose preconditions have already been satisfied. All module parameters have their default values. However, they can be changed by double clicking on the relevant module icon. The wrapper codes can be viewed or edited by 'Edit' function available in the context menu, which opens the script in a text editor. The actual model code is not accessible from here, though their file paths are defined in the wrapper and can be used to open the models in their specific environment (e.g. CUBE, Microsoft Visual Studio).

Models are executed for a specific simulation year or period defined by the model's parameters. The same model can be included in the geo-processing model multiple times to represent different simulation periods. Thus, the overall simulation presented in Fig. 2 starts with MSTM for the year 2007. This is followed by MEM and SILO, which are using the output of MSTM for 2007 and run for year 2007 and the period 2007-2030 accordingly. Then, SILO results for 2030 are used by BEM to run for the year 2030. Though it is not presented in the figure, SILO 2030 results can fed back to MSTM to run it for the year 2030, which then can be followed by SILO running for 2030-2040, and so on.

In addition to a standard status log by the ArcGIS geo-processing window, component models have their separate log files delivering detailed information on the specifics of the model run.

With the use of Python wrappers, the implementation of the coupler is separated from the models' source codes. This gives a flexibility, which can help in terms of portability, performance and maintenance of the codes. Though having limitations, this approach supports different types of components and links them under a single user interface without changing their original source codes. The integrated system automatically calls component models developed in Java (SILO), CUBE script (MSTM and MEM) and Excel (BEM) environments (Fig. 2). Works to add the CBLCM model developed in C++ are in process.

Key benefits and limitations of the system are summarized below:

Benefits

- Open Source.
- No need to change the source codes of the models.
- Allows to run models developed in different programming languages and file formats (e.g. exe, dll, bat, jar).
- Can be extended with additional models over time.
- General user interface showing process flows and linkages between the models.
- Availability of wide documentation and support on Python and ArcGIS.
- Rich visualisation and mapping capabilities through integration with ArcGIS.
- Easy to implement in sense of required time, resources and programming experience.

Limitations

- Parallel model runs and dynamic data exchange during simulation time steps is not supported.
- Model processes run independently from one another.
- Data exchanged between modules need to be written to and read from a hard drive. No in-memory data exchange is available.

It is worth mentioning that these limitations are mainly caused by the constraints of licensing and changing the source codes of the models.

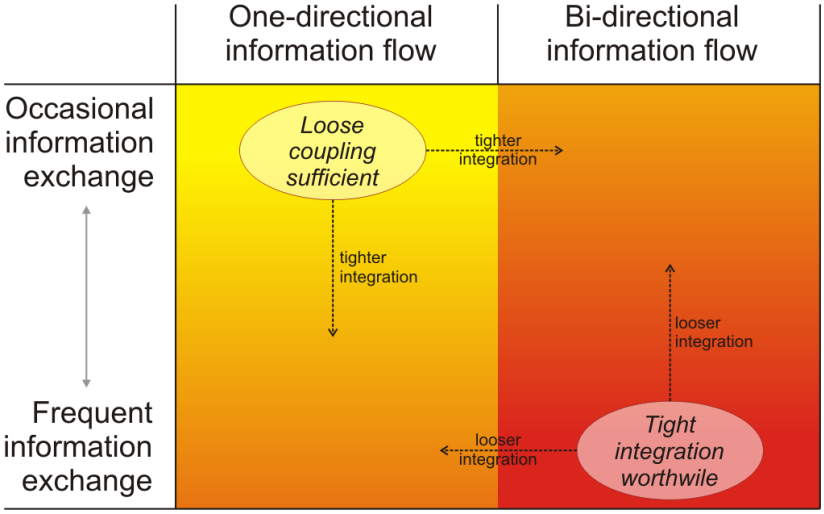
Though the suggested approach focuses on spatial models that integrate transportation, land-use and environmental impact models, the methodology is not limited to this type of systems. It can also be applied to other systems requiring consecutive implementation of standalone components including non-spatial models.

Conclusion

Close model integration has become the mantra among model developers. New tools under development, such as the above-mentioned OpenMI or CSDMS, promote tight integration of different models and ease information transfer between the same. Continuously increasing computational capacities enable ever more comprehensive model integrations. From a technical perspective, the prospects of tight model integration are excellent.

However, the research presented here also exemplified limitations of model integration. Attempts to integrate the models using existing couplers and integration frameworks failed due to a lack of software compatibility and licensing restrictions. Instead, a less sophisticated but robust GIS-based

Fig. 3 Reasons for loose coupling and tight integration



data exchange integration has been implemented, satisfying the project goals. Notwithstanding the ability to run complex model scenarios, the probably most important lesson learned of this research refers to the level of model integration. While the initial attempt of this research was to integrate all models as tightly as possible, the authors developed a much more agile integration approach. A key finding of this research is that model integration should depend on direction of information exchange and frequency of data flows.

Model direction refers to the sender model and receiver model of information. For example, an economic model is used to provide regional control totals of population and employment growth for the entire study area. While the land-use model allocates this growth to individual zones, the overall growth is provided exogenously by this national economic input/output model. In theory, the performance of this study area could be fed back into the economic model, as for example tighter land-use restrictions could push some growth to neighboring regions. In reality, however, the impacts of scenarios for the Baltimore/Washington area on the national economy are minimal. Raising tolls or restricting land development is unlikely to have a notable effect of growth in this study area. Given that economic growth is used as a one-way flow of information, the integration between the economic model and the land-use model is kept offline and solved with a single file transfer covering a 40-year growth forecast. This model linkages is represented by the oval “Loose coupling sufficient” in Fig. 3.

The second aspect of model integration is the frequency of interaction. For example, mobile emissions are calculated every time after the transportation model ran. This is a one-way flow of information: transportation generates emissions and emissions (commonly) do not affect the travel behavior. However, even though this is a one-way flow of information, the exchange of information is frequent enough that the transportation model and the mobile emissions model warrant closer integration. Frequent data flows deserve closer integration to ease information flow, even if the flow is only happening in one-way direction (lower left quadrant of Fig. 3).

The tightest integration should be pursuit for models that exchange information frequently and bi-directionally. In the model presented in this paper, this level of integration applies to the land-use and transportation models. These models exchange information in both directions: the location of households and employment define the origins and destinations in the transportation model, and travel times are converted into accessibilities that affect household relocation decisions. Given the frequency of this bi-directional flow, these two models deserve most attention for simple data

exchange methods to ensure information exchange with little translation loss and limited impact on model runtime.

More integration is not always better. Using the appropriate level of integration improves model stability and runtimes without compromising important linkages between models.

References

Babendreier, J. E. and K. J. Castleton (2005). Investigating uncertainty and sensitivity in integrated, multimedia environmental models: tools for FRAMES-3MRA. *Environmental Modelling & Software* 20(8): 1043-1055.

Basnyat, P., L. D. Teeter, B. G. Lockaby and K. M. Flynn (2000). The use of remote sensing and GIS in watershed level analyses of non-point source pollution problems. *Forest Ecology and Management* 128(1-2): 65-73.

Berry, J., D. Buckley and K. McGarigal (1997). Seamlessly linking ARC/INFO to forest growth and landscape analysis models. *ESRI User Conference*, San Diego, California.

Brandmeyer, J. E. and H. A. Karimi (2000). Coupling methodologies for environmental models. *Environmental Modelling & Software* 15(5): 479-488.

Brazier, R. E., A. L. Heathwaite and S. Liu (2005). Scaling issues relating to phosphorus transfer from land to water in agricultural catchments. *Journal of Hydrology* 301(1-4): 330–342.

Clarke, K. C. and L. J. Gaydos (1998). Loose-coupling a cellular automaton model and GIS: long-term urban growth prediction for San Francisco and Washington/Baltimore. *International Journal of Geographical Information Science* 12(7): 699-714.

Devantier, B. A. and A. D. Feldman (1993). Review of Gis Applications in Hydrologic Modeling. *Journal of Water Resources Planning and Management-Asce* 119(2): 246-261.

FHWA (2014). Maryland State Highway Administration Maryland Statewide Travel Model (MSTM) *Peer Review Report*, US Department of Transportation Federal Highway Administration.

Geertman, S. and J. Stillwell (2009). Planning Support Systems: Content, Issues and Trends. *Planning Support Systems Best Practice and New Methods* 95: 1-26.

Goodchild, M. F., M. J. Egenhofer and R. Fegeas (1997). Interoperating GISs. *Report of a specialist meeting held under the auspices of the Varenius ProjectNCGIA*. Santa Barbara.

Gregersen, J. B., P. J. A. Gijsbers and S. J. P. Westen (2007). OpenMI: Open modelling interface. *Journal of Hydroinformatics* 9(3): 175-191.

Hill, C., C. DeLuca, Balaji, M. Suarez and A. da Silva (2004). *The architecture of the earth system modeling framework.* Computing in Science & Engineering 6(1): 18-28.

Lam, D., L. Leon, S. Hamilton, N. Crookshank, D. Bonin and D. Swayne (2004). Multi-model integration in a decision support system: a technical user interface approach for watershed and lake management scenarios. *Environmental Modelling & Software* 19(3): 317-324.

León, L. F., E. D. Soulis, N. Kouwen and G. J. Farquhar (2002). Modeling diffuse pollution with a distributed approach. *Water Science & Technology* 45(9): 149–156.

Liu, S., R. E. Brazier, A. L. Heathwaite and W. Liu (2014). Fully integrated approach: an alternative solution of coupling a GIS and diffuse pollution models. *Frontiers of Environmental Science & Engineering* 8(4): 616-623.

McFadden D. (1978) Modelling the choice of residential location. In: A Karlqvist, L Lundqvist, F Snickars, J W Weibull (Eds.) *Spatial Interaction Theory and Planning Models*. North-Holland Publishing Company: Amsterdam, New York, Oxford. Pages 75-96.

Moeckel R. (2011). Simulating household budgets for housing and transport. *International Conference on Computers in Urban Planning and Urban Management.* Lake Louise, Canada.

Moeckel, R., S. Mishra, F. Ducca and T. Weidner (2015) Modeling complex Megaregion systems: Horizontal and vertical integration for a Megaregion Model. *International Journal of Transportation*. Forthcoming in April 2015.

Moore, R. V. and C. I. Tindall (2005). An overview of the open modelling interface and environment (the OpenMI). *Environmental Science & Policy* 8(3): 279-286.

Overeem, I., M. M. Berlin and J. P. M. Syvitski (2013). Strategies for integrated modeling: The community surface dynamics modeling system example. *Environmental Modelling & Software* 39: 314-321.

Peckham, S. D., E. W. H. Hutton and B. Norris (2013). A component-based approach to integrated modeling in the geosciences: The design of CSDMS. *Computers & Geosciences* 53: 3-12.

Piacentini, A., T. Morel, A. Thevenin and F. Duchaine (2011). O-Palm: An Open Source Dynamic Parallel Coupler. *Computational Methods for Coupled Problems in Science and Engineering* Iv: 885-895.

Rahman, J. M., S. M. Cuddy and F. G. R. Watson (2004). Tarsier and ICMS: two approaches to framework development. *Mathematics and Computers in Simulation* 64(3-4): 339-350

Roberts, J. J., B. D. Best, D. C. Dunn, E. A. Treml and P. N. Halpin (2010). Marine Geospatial Ecology Tools: An integrated framework for ecological geoprocessing with ArcGIS, Python, R, MATLAB, and C plus. E*nvironmental Modelling & Software* 25(10): 1197-1207.

Schmitz, O., D. Karssenberg, W. P. A. van Deursen and C. G. Wesseling (2009). Linking external components to a spatio-temporal modelling framework: Coupling MODFLOW and PCRaster. *Environmental Modelling & Software* 24(9): 1088-1099.

Shah, A. R., K. J. Castleton and B. L. Hoopes (2004). Framework for risk analysis in multimedia environmental systems: Modeling individual steps of a risk analysis process. *Msv'04 & Amcs'04, Proceedings:* 38-44.

Sudicky, E., J. Vanderkwaak, J. Jones, J. Keizer, R. Mclaren and G. Matanga (2003). Fully-integrated modelling of surface and subsurface water flow and solute transport: Model overview and application. *Developments in Water Science* 50: 313-318.

Tao, C., W. Kainz and R. van Zuidam (1996). Coupling GIS and Environmental Modelling: The Implications for Spatio-Temporal Data Modelling. *International Archives of Photogrammertry and Remote Sensing* 31(B3).

Valcke, S. (2013). The OASIS3 coupler: a European climate modelling community software. *Geoscientific Model Development* 6(2): 373-388.

van Delden, H., R. Seppelt, R. White and A. J. Jakeman (2011). A methodology for the design and development of integrated models for policy support. *Environmental Modelling & Software* 26(3): 266-279.

van Walsum, P. E. V. and A. A. Veldhuizen (2011). Integration of models using shared state variables: Implementation in the regional hydrologic modelling system SIMGRO. *Journal of Hydrology* 409(1-2): 363-370.

Voinov, A. and H. H. Shugart (2013). 'Integronsters', integral and integrated modeling. *Environmental Modelling & Software* 39: 149-158.

Warner, J. C., N. Perlin and E. D. Skyllingstad (2008). Using the Model Coupling Toolkit to couple earth system models. *Environmental Modelling & Software* 23(10-11): 1240-1249.

Welch, T. F. (2013). Climate Action Plans – Fact or Fiction? Evidence from Maryland. *Doctor of Philosophy Thesis*, University of Maryland, College Park.

Wong, I., D. Lam, W. Booty and P. Fong (2009). A Loosely-Coupled Collaborative Integrated Environmental Modelling Framework. *Americas Conference on Information Systems*, AIS Electronic Library.