Model integration and the role of data

Alexey Voinov\textsuperscript{a,*}, Carl Cercob\textsuperscript{b}

\textsuperscript{a}International Institute for Geo-information Science and Earth Observation (ITC), The Netherlands
\textsuperscript{b}US Army Engineer Research and Development Center, Vicksburg MS, USA

\textbf{A B S T R A C T}

Model integration is becoming increasingly important as our impacts on the environment become more severe and the systems we analyze become more complex. There are numerous attempts to make different models work in concert. However model integration usually treats models as software components only, ignoring the evolving nature of models and their constant modification and re-calibration to better represent reality. As a result, the changes that used to impact only contained models of subsystems, now propagate throughout the integrated system, across multiple model components. This makes it harder to keep the overall complexity under control and, in a way, defeats the purpose of modularity, where efficiency is supposed to be gained from independent development of modules. We argue that data that are available for module calibration can serve as an intermediate linkage tool, sitting between modules and providing a module-independent baseline, which is then adjusted when scenarios are to be run. In this case, it is not the model output that is directed into the next model. Rather, model output is presented as a variation around the baseline trajectory, and it is this variation that is then fed into the next module down the chain. The Chesapeake Bay Program suite of models is used to illustrate these problems and the possible remedy.

\textit{Published by Elsevier Ltd.}

\section{1. Introduction}

As our impacts on the environment become more dramatic and the systems we analyze become more complex, there is a growing understanding that one model cannot be sufficient to represent all the details needed for decision making and planning (Argent, 2004; Gaber et al., 2008). There are also numerous legacy models that can be reused as building blocks for more complex systems, provided that they can be linked together in a meaningful way matching the variables, scales and resolutions. Currently there is a growing number of efforts to develop the standards and software tools that would provide for this kind of integration (Warner et al., 2008; Barthel et al., 2008; Argent et al., 2009).

The US Environmental Protection Agency (EPA) has been developing the FRAMES (2009a, 2009b) (Framework for Risk Analysis in Multi-media Environmental Systems) system to manage the execution and data flow among multiple science modules. 3MRA (2009) (Multi-media, Multi-pathway, Multi-receptor Risk Analysis) (Babendreier and Castleton, 2005) is a collection of 17 modules that describe the release, fate and transport, exposure, and risk (human and ecological) associated with contaminants deposited in various land-based waste management units (e.g., landfills, waste piles). FRAMES was developed as the framework that would allow these modules to communicate with each other, facilitating the flow of data between them, and helping with simulation of complex environmental processes.

The Object Modeling System (OMS) is developed by the US Department of Agriculture (David et al., 2002; Kralisch et al., 2004; Ahuja et al., 2005). In contrast to FRAMES and some other systems, OMS requires rewriting modules in Java to be then inserted into the system library.

The Open Modeling Interface and Environment (OpenMI, 2009) developed by a consortium of European universities and private companies, is a standard for model linkage in the water domain (Moore et al., 2005). The OpenMI standard defines an interface that allows time-dependent models to exchange data at runtime. When the standard is implemented, existing models can be run in parallel and share information at each time-step. This helps to link models from different domains (hydraulics, hydrology, ecology, water quality, economics etc.), environments (atmospheric, freshwater, marine, terrestrial, urban, rural, etc.), different scales, resolutions, platforms, etc. If models are made OpenMI compliant they can talk to each other at runtime.

The Common Component Architecture (CCA) is a product developed by the Department of Energy and Lawrence Livermore National Lab teams (Bernholdt et al., 2004), which targets high-performance computers and complex sophisticated models. The CCA supports parallel and distributed computing as well as local
high-performance connections between components in a language-independent manner. The design places minimal requirements on components and facilitates the integration of existing legacy code into the CCA environment by means of the Babel (2004) language interoperability tool, which currently supports C, C++, Fortran 77, Fortran 90/95, and Python. The CCA is being applied in a variety of disciplines, including combustion research, global climate simulation, and computational chemistry.

A generic feature of all these frameworks is that they are mostly concerned with the software side of the integration problem. The models are merely treated as software components that are to be made to work together and talk to each other.

Actually, models are more than that. One of the important issues that distinguishes models from mere software is how they relate to data. Data, in fact, should be treated as an intrinsic part of any model, or even as a model in itself. There are a lot of gains that can be made if instead of mechanistically plugging modules together we take into account the specific goals and features of the system, and approach the problem with some creativity.

2. Linking models

An integrated model is made out of two or more independent components. Each of these components can operate on its own and in many cases has been developed independently by separate groups of researchers. The promise of integration in this case is that legacy code and models can be reused to analyze more complex systems, while analysis is simplified since the overall system can be studied and modeled in portions. Suppose we have model A for one sub-system and model B — for another. If a system is a composite of these two sub-systems, then instead of building a whole new model C to represent it, it should be possible to use an integration of the two existing models and model the system with an AB model, where the two components A and B exchange information as they run. For simplicity let us assume that there is only a one-way flow of information from A to B (Fig. 1).

What should be the calibration process in this case? If A and B existed before, they were most likely calibrated previously, based on observations. If not, still it should be much easier to calibrate smaller and simpler components, therefore we may assume that A and B are calibrated separately. However, after integrating A and B we should not expect that the resulting output from the AB model will match data as well as output from B. Note that for our system it is the output from B that matters, since B generates output for the integrated model AB.

The reason that AB will generate different results than the calibrated standalone model is that now the forcings for B come from the calibrated model A, instead of being taken from data. No matter how well A is calibrated, its output is likely to deviate from the data, and will steer AB away from what B previously was generating. The results can be still improved by some further calibration of B as part of the integrated system. In this case this will look like a calibration of the AB model as a whole. This will also include some refinement of overall model performance by model B calibration, in fact, compensating for some deficiencies in the calibration of model A. Moreover, we may be even tempted to further tweak some of the parameters in A to get a better match. This is exactly what is done in integrated models: they need adjustment and re-calibration after the components are put together. However, by doing this we lose much of the advantages of the modular architecture in the integrated AB model. We now need to deal with the calibration of the full model, and whenever any changes are made to any of the components, the other components need to undergo new calibration as well.

Let us consider, for example, one of the best-known success stories of model integration — the modeling suite developed by the Chesapeake Bay Program (Cerco, 2000; Linker et al., 2000; Wang and Johnson, 2000). It consists of three major parts:

- The atmospheric transport model that produces atmospheric deposition predictions for nutrients and other constituents.
- The Airshed Model is based on the Community Multi-Scale Air Quality modeling system (CMAS). The latest CMAS (2009) code runs on a 12 km fine grid in the Chesapeake region, with a 36 km grid used for the continental scale boundary conditions;
- The watershed model, a highly-modified version of HSPF (Linker et al., 2000; Bicknell et al., 1996), that produces loadings that come from the land into the estuaries;
- The estuary model, which is, itself, a combination of three linked models, a hydrodynamic model (Johnson et al., 1993), a eutrophication model (Cerco and Cole, 1993), and a sediment diagenesis model (DiToro, 2001). The Water Quality and Sediment Transport Model (WQSTM) is a three dimensional model of the tidal Bay comprised of 57,000 cells. The current WQSTM represents transport processes, eutrophication processes, and living resources such as submerged aquatic vegetation and benthos. It is further linked to models of trophic networks in the Bay (Cerco et al., 2010).

The modeling system has been developed over the past 25 years and went through many phases. The watershed model is currently in phase 5.1, and phase 5.2 is coming out soon.

The models are linked loosely, so there is no formal software mechanism involved. Output from one model is sent to another model as an input file. Yet still, the models work in concert and are used for decision-making purposes as a suite. Decisions are mostly based on the predictions for the future state of the Chesapeake Bay in terms of such indicators as the area of hypoxia, or suitability of habitat for living resources, while most of the decisions are made for the watershed, where the nutrient and solid loads are generated.

Therefore, the connection between the watershed and the estuary model is crucial. The estuary model is very much dependent upon the loadings that it receives from the watershed model. Whenever the watershed model gets updated, it produces different output (Fig. 2). As a result, every time the watershed model is changed, the estuary model needs to be re-calibrated (Figs. 3 and 4).

In theory, as the watershed model moves from one phase to the next, its output is supposed to become ‘better’ in terms of matching the data; the model is supposed to be better calibrated. Sometimes this is indeed the case; in other cases the upgrades are driven by the need to include new features important for management, or by the need to change the resolution in the model, as happened between

![Fig. 1. The standard way of model integration, when the Output from one model (Module 1) is fed as Input into the other model (Module 2). The Data that is used for calibration of Module 1 is not part of the linkage. Whenever Module 1 is changed and undergoes re-calibration, Module 2 requires re-calibration as well.](image-url)
Phase 4 and 5. The output in this case becomes ‘different’ but not necessarily ‘better’ in terms of a better fit to data.

Again and again the estuary model needs to undergo tedious re-calibration, which is entirely caused by developments that are external to it, and do not necessarily have anything to do with improvements of the estuary model itself. Much effort and time is spent with the only benefit of keeping the components working together.

3. The data comes in

This may be the place where data becomes really important in model integration. The fact that two components are integrated into one model does not have to make the available calibration data no longer relevant. When the downstream model, the estuary model in our example, was first calibrated, it was forced with observed loads, and there is no reason to think that the output from the upstream model, the watershed model, is more accurate than these observations. If the data was still embedded in the integration process, then there would be no need for model re-calibration every time modifications are made ‘up the modeling stream’. Indeed, when new and ‘better’ modules are developed, the data are still invariant, and there is no reason why the same data set that was used for calibration of an upstream model A cannot be used as forcing functions for the ‘down-stream’ model B (Fig. 5).

In case of the Bay model, the data for nutrient loadings should stand between the output of the watershed model and the input of the estuary model. When the watershed model is modified, and the next generation, improved model is produced, the estuary model does not need to be re-calibrated, because still the watershed model is expected to represent the same data as before, and it is that data that feeds into the estuary model.

In a way we are suggesting to employ a version of data assimilation, when data become part of the overall integrated modeling structure. Certainly, in many cases the output of the upstream component is much more information-rich than what the observed data sets offer. For example, in case of the Bay model, the watershed model generates inputs for every tributary, while data sets are available only for some. Yet still that is not a reason to exclude the available observations and replace them entirely by ‘artificial’ data. Wherever and whenever observations are available they should be used and should have precedence over model-generated information. What we suggest is a data-model fusion when integrating components.

But how do the two model components run in concert, if the data component sitting between them is invariant to the changes in the forcing functions that drive the upstream components? After all, most models are built to run scenarios, where certain parameters and forcing functions are modified to answer the ‘what if’ questions or make predictions for the future. How do we treat scenarios that produce results that are not available as part of the existing data sets?

Consider our cascade of models, A and B. In the existing integrated model design, the scenario runs are fed directly from one component into the other one. The output from model A is used as input for the forcing functions in model B. Basically in the integrated model, the components are no longer separated and are assumed to run as one model with results from one component flowing directly into the next one inside the integrated model. There is hardly any place for intermediate data sets that we may wish to employ.

Suppose $S = S(t(x,P,F))$ is the model output. Here $t$ is time, $x$ is the spatial coordinate, $P$ is the vector of parameters, $F$ is the vector of forcing functions.

Let us define the baseline scenario, $S^* = S(t(x,P^*,F^*))$, as the model output that produced the best fit to data during model calibration process. If $D(t,x)$ is the set of observations over which the model is
calibrated, then the baseline scenario, $S_A^*$, is the model output with the observed forcings $F^*$ and with the parameter set, $P^*$, which is the solution to the calibration task:

$$\min |S(t,x,P,F^*) - D(t,x)|$$

where we minimize over the vector of parameters, $P$.

Then $S_A^*$, the baseline scenario for model A, is produced as a result of calibration of model A to the data set $D_A$. Any other scenario that we run with model A can be considered as a perturbation of the baseline scenario caused by changes in the forcings, $F_A^*$, and the parameters $P_A^*$. Suppose we modified the land use in the watershed model, which means that the forcing function and, perhaps, some parameters have been changed. We generated new model output $S_A = S_A(t,x,P_A,F_A)$. The perturbation of the baseline scenario will be $\Delta_A = S_A - S_A^* = S_A(t,x,P_A,F_A) - S_A(t,x,P_A^*,F_A^*)$.

Before the two components were integrated, we used the calibration data set $D_A$ as forcings for the downstream model B. The model output from B was $S_B(t,x,P_B,D_A)$. When model A got integrated with model B, the model A output was taken and fed into model B as its forcings: $F_B = S_A(t,x,P_A,F_A)$. After which model B was run to produce $S_B(t,x,P_B,F_B) = S_B(t,x,P_B,S_A(t,x,P_A,F_A))$.

The data set $D_A$ became irrelevant and no longer used as an intermediate calibration mechanism. It was replaced by the output from model A. We suggest that instead of feeding output from A directly to B, we use A to calculate the perturbation of the baseline scenario and then use this perturbation to modify the existing data set $D_A$ and then use this modified data set as the scenario to drive model B. We use the perturbations of the base run, $\Delta_A$, to perturb the data sets that were previously used to drive the next model down the stream. Under various parameter sets and forcings, $F_A$, the model ‘above’ will be generating scenarios, $S_A^i$, $i = 1,2,\ldots$ for various scenarios. Instead of feeding this output directly into the model ‘below’, we will be calculating the perturbation of the baseline scenario, $\Delta_{A,i} = S_{A,i} - S_A^*$, and feeding $\Delta_{A,i}$ into the next model as forcings. The output from the ‘above’ model is always tied to the data, and it is the same data set that was used in the calibration of the ‘below’ model that now gets modified by the scenarios run with the ‘above’ model. This helps to keep the components independent and avoid additional re-calibration.

Certainly there is no guarantee that the perturbed data sets, $D_A + \Delta_A$, will be meaningful (will have ecological sense). However, there is no more reason to expect that the output from the upstream model will be meaningful when fed directly into the downstream model. When models are loosely coupled and there is some intermediate data processing, there is opportunity to identify model inputs which are physically impossible. But with tight coupling and results from scenario runs directly piped from one model into another, it becomes unlikely that this will be noticed. Therefore, in both cases, with direct coupling and with data based coupling, we need to apply certain checks and balances. For example, check that $D_A + \Delta_A$ are positive, or is within certain limits, etc.

Yet another concern is that by modifying the flow from one component to another we may be creating discontinuities and breaking the mass balance in the model. Indeed, the upstream model may be outputting a flow of water, or material, $Q_A$, which was supposed to be input into the downstream model. Instead a different flow, $D_A + \Delta_A$, calculated based on the calibration data set is fed, into the next component. Most likely, $Q_A$ will not be equal to $D_A + \Delta_A$. It appears that in each particular case we will need to decide, whether the mass balance between components is more important than calibration precision, and module independence. This can become a major concern when components exchange information in both directions: A $\rightarrow$ B, and B $\rightarrow$ A.

In the case of the Bay modeling suite, where the flow of information is unidirectional, from the ‘above’ component to the ‘below’ one, from the watershed into the estuary, with no feedback, this is not an issue. A more accurate and better calibrated output is of higher value than mass balance between components.

As a result, we describe a scenario as relative to the base run, as a perturbation of the base scenario, which is then fed into the next component as an increment to the same data set that was used to calibrate that downstream component before. In a way, at any time step, the upstream component output is corrected based on the data set available, before it is fed into the next component as input. What is most important, we no longer need to undergo the tiresome re-calibration process, when changing the components ‘above’. Only the scenario runs will need to be rerun, since, obviously, a modified upstream component will likely produce different set of perturbations, which will change the performance of the components ‘below’.

Yet another way to look at data sets is to think of them as components or modules as well. In fact, statistical or other models of the data must be developed to fill the gaps between the spatially and temporally discrete observations (e.g. Cohn et al., 1989). In this case data modules will exist along with model modules and when designing integrated systems we will choose which modules to use in which cases.

4. Conclusions

The case study presented by no means devalues the importance and need for module linking software tools, component interfaces and specifications. We are only reminding that models are more than just pieces of software and they require more to make them
work together efficiently. For example, incorporating data sets in appropriate places within the integrated modeling system can help to keep components independent in order to avoid the propagation of perturbations and changes down the stream from one model to another. This can substantially increase the efficiency and accuracy of the integration process. We need data sets to be recognized as components on the same level as models. Such data components can then enter the integrated frameworks at various places, not only at the top, as input to drive the integrated model, and at the bottom, to compare with the output and to calibrate the model. Data components can be also used between components to test, adjust, and correct the data flows inside the integrated model. This will help to keep components independent, and reduce the overall complexity of the calibration task for the integrated model.

References


