METHODS REPORT

Coupling Models by Routing Communication Through a Database

Eric Solano, Robert J. Morris, and Georgiy V. Bobashev

September 2013

RTI Press

RTI INTERNATIONAL

About the Authors

Eric Solano, PhD, who has been with RTI International since 1999, is a data scientist and model analyst with extensive experience in mathematical models, statistical methods, data mining, data-driven projects, database technologies, software engineering, software development, geographic information systems, decision support tools, operations research, and optimization.

Robert J. Morris, BS, who has been with RTI International since 2000, is a programmer and statistician. He performs software development and statistical data analysis as well as programming for quality control, data editing, and data management.

Georgiy V. Bobashev, PhD, who has been at RTI International since 1998, is a senior research statistician and an expert in biostatistics methodology and mathematical modeling. His current research interests cover two major areas: substance-use studies and predictive modeling. In the substance-use research area, he focuses on personalized treatments and a systems approach to addictions. In predictive modeling, he focuses on methods development in forecasting health outcomes.

RTI Press publication MR-0026-1309

Suggested Citation

Solano, E., Morris, R. J., & Bobashev, G. V. (2013). *Coupling models by routing communication through a database* (RTI Press publication No. MR-0026-1309). Research Triangle Park, NC: RTI Press. Retrieved from http://www.rti.org/rtipress

This publication is part of the RTI Methods Report series.

RTI International
3040 East Cornwallis Road
PO Box 12194
Research Triangle Park, NC
27709-2194 USA

Tel: +1.919.541.6000
Fax: +1.919.541.5985
E-mail: rtipress@rti.org
Web site: www.rti.org

doi: 10.3768/rtipress.2013.mr.0026.1309 **www.rti.org/rtipress**

Coupling Models by Routing Communication Through a Database

Eric Solano, Robert J. Morris, and Georgiy V. Bobashev

Abstract

As the number of available large and many-faceted computer models continues to increase, simulating complex systems by coupling existing models of smaller subsystems becomes more attractive because of advantages such as leveraging existing programming. Advances in computational technologies also contribute to the increased feasibility of coupled systems. Although coupled systems may be used to study new problems that their constituent models could not address, the coupling process brings its own challenges. The modeler may face the task of coupling models from a heterogeneous environment of development platforms, programming languages, and model assumptions. Moreover, the modeler may wish to allow constituent models to be replaced or upgraded without significant difficulty. We discuss a model coupling approach that attempts to address these issues. In our approach, the models run as separate executable processes and store data in a database for later retrieval by other models. While the approach does not prescribe any particular database design, we do suggest elements that are likely to appear. We describe a proof-of-concept application of the approach and evaluate how well our approach meets its goals.

Contents

RTI INTERNATIONAL

Introduction

In the past few decades, multiscale modeling (i.e., where separate models examine the same process but at different scales or resolutions) and multicomponent modeling have become increasingly important and widely used in a number of research areas. Examples of multiscale or multicomponent models can be found in biology, hydrology, climatology, warfare simulations, and many other disciplines (Eddy & Schlessinger, 2003; Swain & Wexler, 1996; Warner et al., 2008; Zacharias et al., 2006). As noted, multiscale models often include subsystems representing different scales, be they temporal, spatial, or biological. Similarly, multicomponent models often include subsystems representing different components, such as cities, bodily organs, or climate systems. In cases where standalone models of these subsystems already exist, modelers may wish to link the existing subsystem models together instead of implementing their own versions. This linking process is known as model coupling.

Our work focuses on coupled systems in epidemiology, particularly coupled systems that analyze epidemic phenomena at both global and local scales. Coupling a global component with numerous local components is important for at least two reasons. The first is that even though local events, such as the emergence of a new strain of influenza and ways to contain the outbreak, are often of great interest to public health decision makers, local decisions could depend on the global features of the epidemic, including travel from other regions and supply and delivery of pharmaceutical preparations such as vaccines or antivirals. The second reason is that global disease spread, in turn, depends on the interaction of numerous local events. A pandemic involving millions of people worldwide could be driven by a small number of infected individuals traveling around the world and unknowingly spreading the disease locally.

Epidemiology modelers have many options when choosing an abstraction for disease transmission. These options include patch models, distance models, multigroup models, and network models (Riley, 2007). Another reason for coupling models, then, is to combine abstractions to exploit the advantages offered by each. For example, patch models and other equation-based models (EBMs) are attractive because they are computationally efficient and may be parameterized simply. Network models and other agent-based models (ABMs) are attractive because they can capture individual interactions and represent adaptive behavior. Coupling an ABM for local dynamics with an EBM for global dynamics allows the modeler to represent disease transmission at both local and global scales without either the heavy computational burden of an ABM or the loss of individual variation that is critical during the early stages of an epidemic, which is expected from an EBM (Bobashev, Goedecke, Yu, & Epstein, 2007).

The design of such a coupled system needs to be flexible to accommodate scientific questions asked at different scales. For example, if the questions were to concern disease dynamics at a global scale, such as disease spread in a large region, the details of local spread could be suppressed, and a simplified EBM could be used instead of a detailed ABM for all critical locations. Similarly, as the questions switched from one local area to another, the details of spread in other local areas could be suppressed. At the same time, to accommodate collaboration among scientific communities, the design should enable the coupling of models developed by different research groups using a computational platform convenient to each particular group.

Bulatewicz (2006) reviewed existing approaches to coupling multiple models, grouping them into four categories: monolithic, scheduled, component, and communication. The monolithic approach involves splicing the pieces together into one big model (Guo & Langevin, 2002; Jobson & Harbaugh, 1999; Swain & Wexler, 1996). Such an approach is attractive because it creates a single module. However, it tends to require significant reprogramming that must be repeated whenever a constituent model is added or replaced. In addition, it requires all the models to be implemented in the same programming language.

In the scheduled approach, models are executed as independent programs and do not communicate with each other while they are executing. Instead, each model produces output data sets at the end of

its execution. These output data sets are used as the initial data sets in other models. This approach is not suitable when models need to exchange data while they are executing.

Like the monolithic approach, the component approach combines individual models into a single executable model, but it improves on the monolithic approach by encapsulating each model into its own distinct component. The components communicate with each other through a well-defined interface, typically using functions or libraries that are provided as part of a framework. This approach avoids much of the reprogramming necessary in the monolithic approach when a model is added or replaced. However, it still requires each model to be implemented in the same programming language, and the allowed languages are limited to those supported by the framework. Examples of the component approach are the General Coupling Framework (Ford, Riley, Bane, Armstrong, & Freeman, 2006) and the Model Coupling Toolkit (Larson, Jacob, & Ong, 2005).

Finally, in the communication approach, models are executed as independent programs and communicate with each other during execution using some form of message passing. In this approach, no significant reprogramming is required to add or replace a model, and models need not be implemented in the same programming language, although the allowed languages are limited to those supported by the message-passing library. Frameworks that use the communication approach can be classified by whether or not they include independent applications (couplers) to mediate the execution and communication among the models. Frameworks that do include a coupler have communication libraries that support direct model-to-model communication as well as model-to-coupler communication. Frameworks that do not include couplers are essentially libraries of data transfer and transformation routines customized for the data types (grids, flux, etc.) and communication styles (high-bandwidth, parallel, etc.) needed by models.

Examples of frameworks with couplers are the Potential Coupling Interface (PCI) (Bulatewicz, 2006) and the Standard for Modeling and Simulation (M&S) High-Level Architecture (HLA) (Institute of Electrical and Electronics Engineers [IEEE], 2000). Other frameworks use databases to aid in the process of execution and communication among the models.

The PCI describes the coupling potential of a model, or the aspects of the model that affect how it can be coupled to another model. PCIs are created as model metadata and the variables accessible at each coupling point are specified. When a PCI is created, the original model source code is automatically instrumented with communication code, and after it is compiled, the resulting coupling-ready executable can be used in any coupling.

The HLA defines the format and syntax of HLA object models. The HLA is an integrated approach developed to provide a common architecture for simulation. Model interactions in the HLA are specified through interaction classes in an interaction class structure table, similar to the way in which objects are described in an object class structure table.

The use of databases in communication approach frameworks has been reported in the literature. In one framework (Hoheisel, 2002), control and the data communication among the distributed simulation models are realized by specialized markup languages that employ the XML technology for extension and validation. The input and output data of the models are represented by an XML data model that includes also the physical and mathematical meaning of the transferred data so that different models can be easily coupled by connecting their XML formatted input and output streams, via sockets. The initialization parameters and the relevant output data of the models are stored in an SQL database (MySQL).

Another framework (Brandmeyer et al., 2004) uses the U.S. Environmental Protection Agency's Multimedia Integrated Modeling System (MIMS) to specify parameters and execute multiple environmental models. The MIMS framework provides tools for connecting executable programs, launching programs in sequence, and transferring data between those programs. An Oracle server and databases supply historical data for use with the models. The framework includes connectivity scripts (couplers) that transfer data between the models and the Oracle database.

We have developed a communications-based approach where models run concurrently while exchanging data through an intermediate relational database management system. Although we developed the approach to produce multiscale epidemic models, it can also be applied to other disciplines and to settings in which models are distinguished by factors other than scale. In the next sections, we start with a general description of the approach. Then, we describe a practical example of its application that we developed as a proof of concept. Finally, we evaluate how well the approach meets its goals and discuss its limitations.

Methods

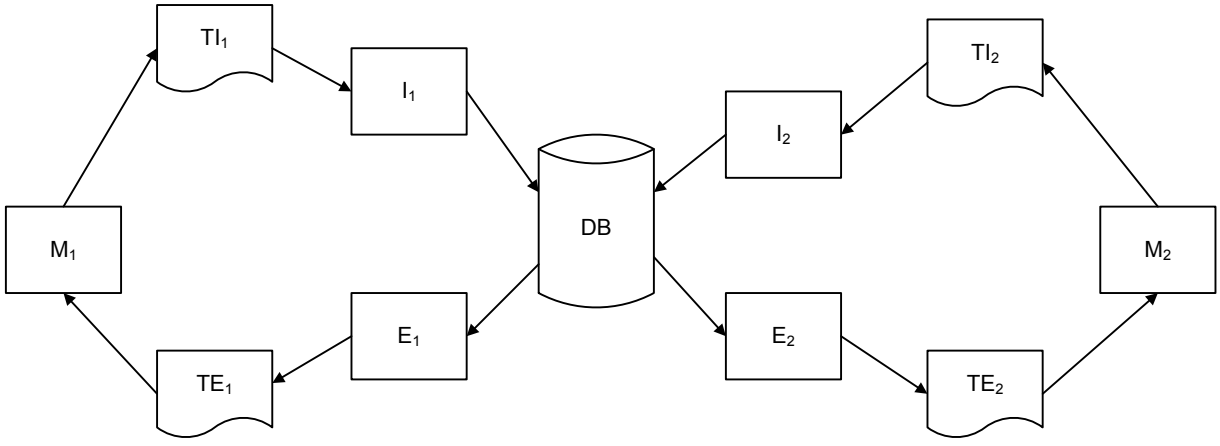
General Approach

We present an approach to model coupling in which the models run concurrently while exchanging data through an intermediate relational database management system. Auxiliary tools tailored to each model assist the models in inserting and extracting data into and out of the database. Logically, the auxiliary tools are known as "inserters" and "extractors." Because the models run concurrently, and because the models may use data obtained from other models in their own procedures, it is important for the models to keep their internal clocks synchronized. In addition, the inserters and extractors need to be scheduled to run at appropriate times. Our approach uses the database for clock synchronization and process scheduling.

Figure 1 displays a schematic illustrating our approach when applied to a situation in which two models are coupled. In the figure, M_1 and M_2 represent the models, DB represents the database, I_1 and I_2 represent the inserters, and E_1 and E_2 represent the extractors. Clock synchronization and process scheduling are omitted for simplicity. When model M_1 needs to send data to model M_2, it exports the data in a format convenient to model M_1. The exported data are represented in the schematic as TI_1. Inserter I_1 (the inserter for model M_1) then imports the TI_1 data and inserts them into the database. Next, extractor E_2 (the extractor for model M_2) extracts the data from the database and exports them into a format convenient to model M_2. The exported data are represented here as TE_2. Finally, model M_2 imports the TE_2 data. A similar path is followed in the reverse direction when model M_2 needs to send data to model M_1.

The inserters and extractors are not essential to the procedure and may be omitted if the models can conveniently write their data directly to the database. However, using them has the advantage of isolating the models from the database. This allows the models to interact with the data in ways that

Figure 1. In the general approach, when applied to a situation in which two models are coupled, the models run concurrently while exchanging data through an intermediate relational database management system.



M_1 and M_2 = models; DB = database; I_1 and I_2 = inserters; E_1 and E_2 = extractors; TI, TI_2, TE_1, and TE_2 = data.

make no assumptions about the database structure. Instead, any such assumptions are implemented in the inserters and extractors. As a result, changing the database structure requires changes only to the inserters and extractors, which are presumably shorter and less complex than the models and therefore easier to modify.

The data model design will likely differ from one coupled system to the next. The design may depend on many factors, with the most obvious being the specific data types to be exchanged or the frequency with which instances of each data type are exchanged. Despite these variable factors, we expect most database designs will share a few common elements. The database design will probably include a table representing all the models; a table representing all model runs, so that other tables can contain data from multiple model runs; a table that will hold the actual data items as they are exchanged; a table defining a standard representation for each common data type to be exchanged; and one or more tables that handle clock synchronization and process scheduling.

Clock synchronization and process scheduling methods may also vary considerably from one coupled system to another. Coupling discrete-time models may call for a different design than coupling discrete-event models, for example. For discrete-time models that exchange data at every time step (i.e., time is broken up into slices or steps), scheduling each process to run in a particular order at every time step may be sufficient to keep the model clocks synchronized. In this case, the database would probably need tables for process scheduling but not for clock synchronization. For discrete-time models that exchange data at irregular or unpredictable time intervals and for discrete-event models that exchange data at arbitrary time points, dedicated clock synchronization tables are probably necessary in addition to the process scheduling tables. In our approach, the models, inserters, and extractors communicate directly with the database to query and update both the clock synchronization and process scheduling tables.

Proof of Concept

To illustrate our general approach to model coupling, we developed a proof of concept using two existing and proven models of flu transmission. The first is the Global Epidemic Model (GEM), a stochastic, equation-based model that examines flu transmission on a global scale (Epstein et al., 2007). The second is a stochastic, agent-based model that examines flu transmission on a local scale in rural Southeast Asia (Longini et al., 2005). We refer to this local model as the SE Asia model.

The two models differ in some fundamental ways. From a technical standpoint, the primary difference is that the GEM is implemented in Java programming language while the SE Asia model is implemented in C programming language. The primary substantive difference is that the two models simulate flu transmission at different scales. The GEM is designed to model flu transmission among hundreds of cities around the world, with relatively simple contact dynamics and population structure within each city. On the other hand, the SE Asia model limits its scope to a single rural region, but the contact dynamics and population structure within that region are more complex than those found in the GEM. For example, the SE Asia model varies contact rates based on a person's age and also assigns personal social networks such as a person's family, neighborhood, classmates, and coworkers. The GEM imposes none of this structure and instead assumes homogeneous mixing within each city.

The GEM models disease transmission among its cities by simulating daily travel from city to city. (We use the term "city" for convenience, but actually the nodes may represent any geographic area.) The GEM bases travel on a transportation network in which each node is a city and pairs of nodes are connected when the daily travel total between those cities is nonzero. Conceptually, coupling the GEM and the SE Asia model means treating the region represented by the SE Asia model as another node in the GEM's transportation network. The difference between the rural Southeast Asia node and the other nodes is that the other nodes are implemented internally by the GEM, while the rural Southeast Asia node is implemented externally by the SE Asia model. As a result, when the GEM directs travelers into its rural

Southeast Asia node, it must recognize that the node is implemented externally and export the travelers to the SE Asia model via the coupling infrastructure. The GEM must also "listen" to the coupling infrastructure to detect travelers who are exiting the rural Southeast Asia node. Similarly, the SE Asia model must interact with the coupling infrastructure to send travelers to the GEM and to receive travelers from the GEM.

While our approach allows coupled models to exchange multiple data types, for the sake of simplicity we chose just one data type for our proof of concept: the traveler. The traveler data type has several associated attributes that are useful to one or both models, most notably disease status. Other data types a modeler may wish to exchange when coupling infectious disease models include vaccine doses and incidence rates.

Database

In this section, we discuss the design of the relational database that supports the models within the proof-of-concept coupled system. While we designed the database specifically to support the GEM and the SE Asia model, a modeler could use relational principles to modify the design to support the data requirements of other models and analysis tools. We implemented the database using PostgreSQL, an open-source relational database management system.

The tables in the database can be split into a few groups: a table listing the models in the coupled system, tables defining the models' representations of the traveler data type, tables describing how to convert between the models' representations of the traveler data type, tables defining individual model runs, a table containing the exchanged traveler data, and a process scheduling table.

The t_model table represents the models available in the coupled system and includes a name assigned to each one. Other models and tools refer to the models by these names.

The traveler data type consists of six attributes relevant to influenza infection. The database contains a table for each of these six attributes, and each table contains the attribute values that are supported by each model. This set of tables thus defines the models' representations of the traveler data type. The six tables and their contents are shown in Table 1.

Table 1. Attributes of the traveler data type related to influenza infection

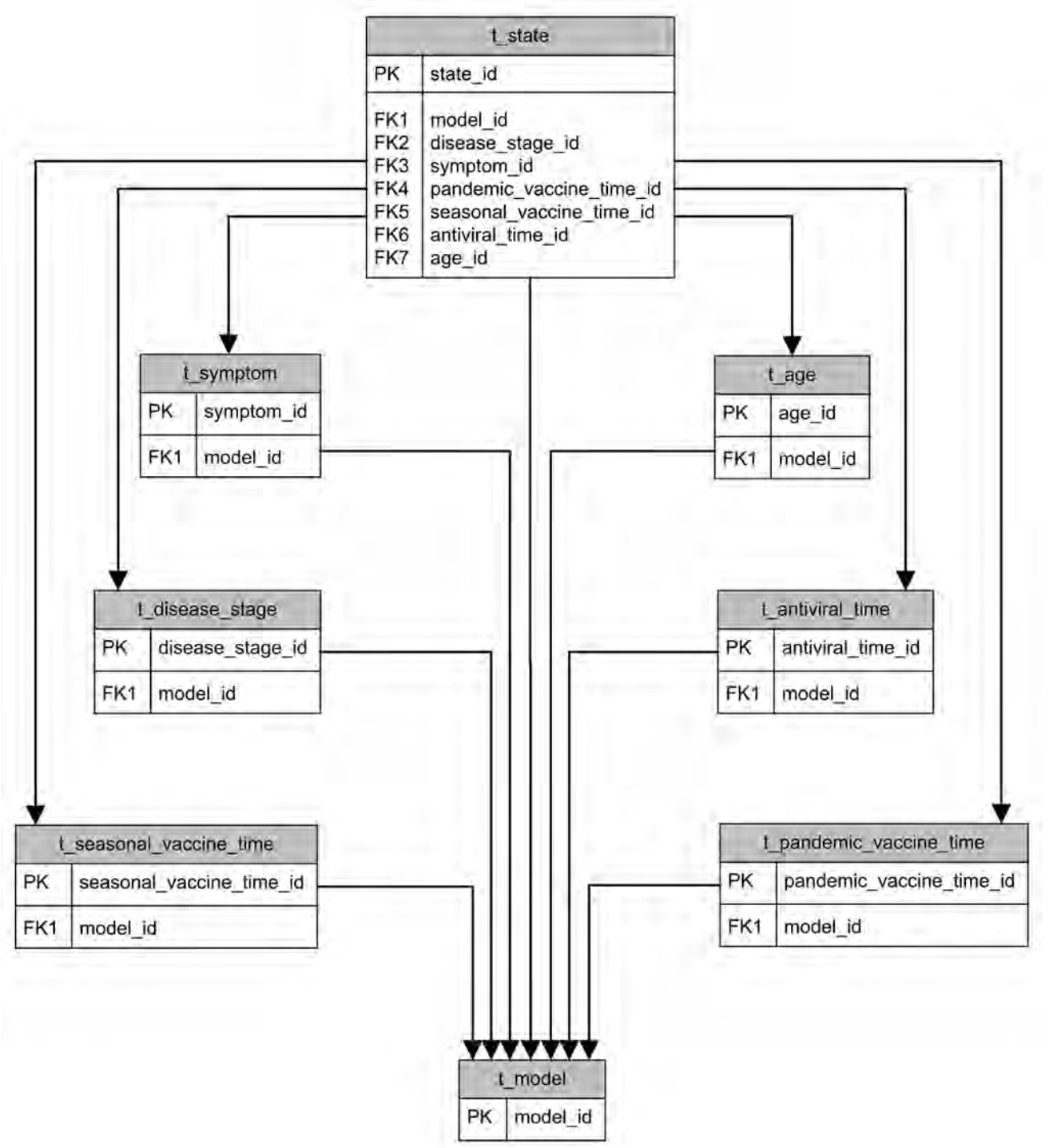
Table Name	Contents
t_disease_stage	Information about the stage of the disease's natural history
t_age	Age groups supported by each model
t_symptom	Information about the symptom severity
t_antiviral_time	Time steps since antiviral treatment began
t_pandemic_ vaccine_time	Information about the time steps since the first pandemic vaccine dose was administered
t_seasonal_ vaccine_time	Information about the time steps since the first seasonal vaccine dose was administered

In addition, the database contains a table called t_state that lists all the unique combinations of attribute values in the six attribute tables. All travelers entering the database are assigned a single state value from t_state representing a specific combination of the six attribute values. This simplified representation of the traveler's attributes is used in other tables so that attributes can be added and removed without requiring fields to be added or removed in the other tables. Figure 2 shows the entity-relationship diagram (ERD) for this set of tables.

The database contains a "crosswalk" table corresponding to each attribute table. The crosswalk table describes how to convert between the GEM's attribute values and the SE Asia model's attribute values. For example, consider the age attribute. The SE Asia model implements eight age groups, so the t_age table includes eight records for the SE Asia model, one record per age group. On the other hand, the GEM does not support age categories, so the t_age table includes a single age group called "n/a" (not applicable) for the GEM. When travelers move from one model to the other, they must be assigned an age group that is compatible with their destination model. Records in the t_age_crosswalk table provide specifications for making these assignments. One record says that 44 percent of the travelers from the

GEM's n/a age group should be placed into the SE Asia model's 18–44 age group. Another record says that 11 percent of the travelers from the GEM's n/a age group should be placed into the SE Asia model's 5–10 age group. Other records handle travel in the opposite direction. One record says that 100 percent of the travelers from the SE Asia model's 18–44 age group should be placed into the GEM's n/a age group. Another record says that 100 percent of the travelers from the SE Asia model's 5–10 age group should

Figure 2. Entity-relationship diagram (ERD) for the t_state and t_model tables in addition to the attribute tables



PK = primary key; FK = foreign key

be placed into the GEM's n/a age group. All other combinations of GEM and SE Asia model age groups are similarly represented in the t_age_crosswalk table.

The database contains two tables for defining individual model runs: t_scenario and t_run. The t_scenario table assigns a name and a unique identifier to experimental scenarios, and the t_run table lists the multiple replicate iterations that are run for each scenario.

The t_travel table contains the travel data that the models exchange at every time step. This table identifies the time step when the travel occurred, the traveler's origin and destination models, and the traveler's state—that is, the unique combination of attribute values assigned to the traveler.

The t_event table is the process scheduling table. This table allows the coupled system to coordinate the execution of the models, inserters, and extractors in sequence by keeping information about the different events that occur as they run. These events represent the actions of the models, inserters, and extractors as they reach points in their execution when they are ready to pass data to the next step in the process. For example, the SE Asia model will update an event's status in the table once it has completed writing an output file with the number of travelers exiting rural Thailand.

Figure 3 shows the ERD for all the database tables except the six attribute tables and their corresponding crosswalk tables. Figure 4 shows the ERD for the entire database.

The data structure includes primary keys in all data tables and foreign keys in most data tables to guarantee referential integrity. These keys reduce data errors by providing quality assurance as data values are added to the database. The relational design allows tables to be linked.

In the ERDs, the fields composing a primary key are labeled PK, and those composing a foreign key are labeled FK1, FK2, etc. The ERDs show a fully interconnected database in which parent and child tables are related by foreign keys and cardinality relationships.

A modeler who decided to add a new model to our proof-of-concept coupled system or to upgrade the GEM or SE Asia model might first need to alter our database design to accommodate features of the new or upgraded model. While we did not attempt to anticipate every possible change, we did design the database with one change in mind: scaling to accommodate additional attributes of the traveler data type. Each traveler attribute is represented by its own table in the database. If a new model introduced into the coupled system were to implement additional

Figure 3. Entity-relationship diagram for the t_model, t_travel, t_scenario, t_state, t_event, and t_run tables

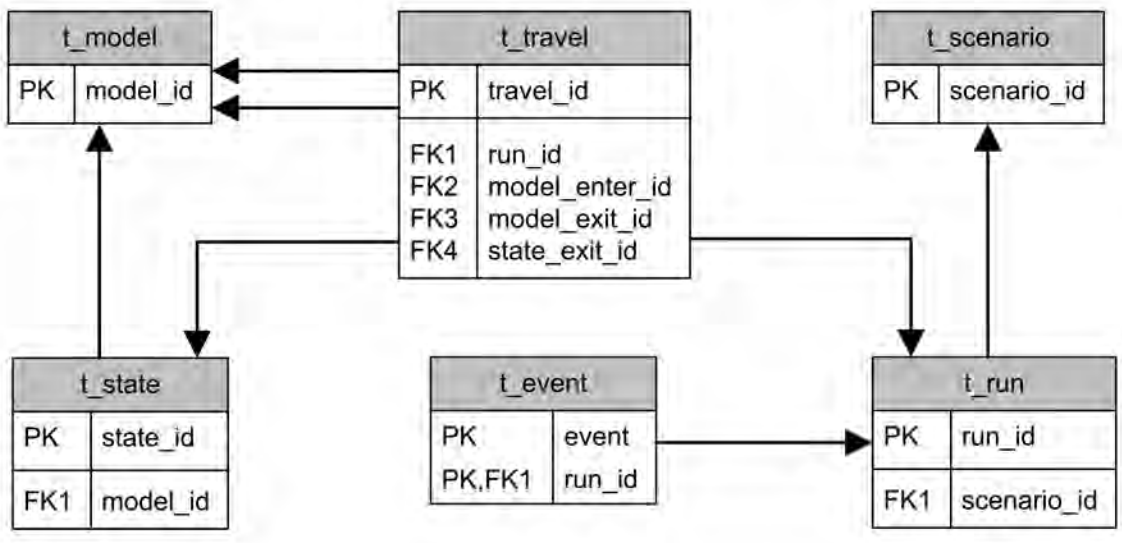
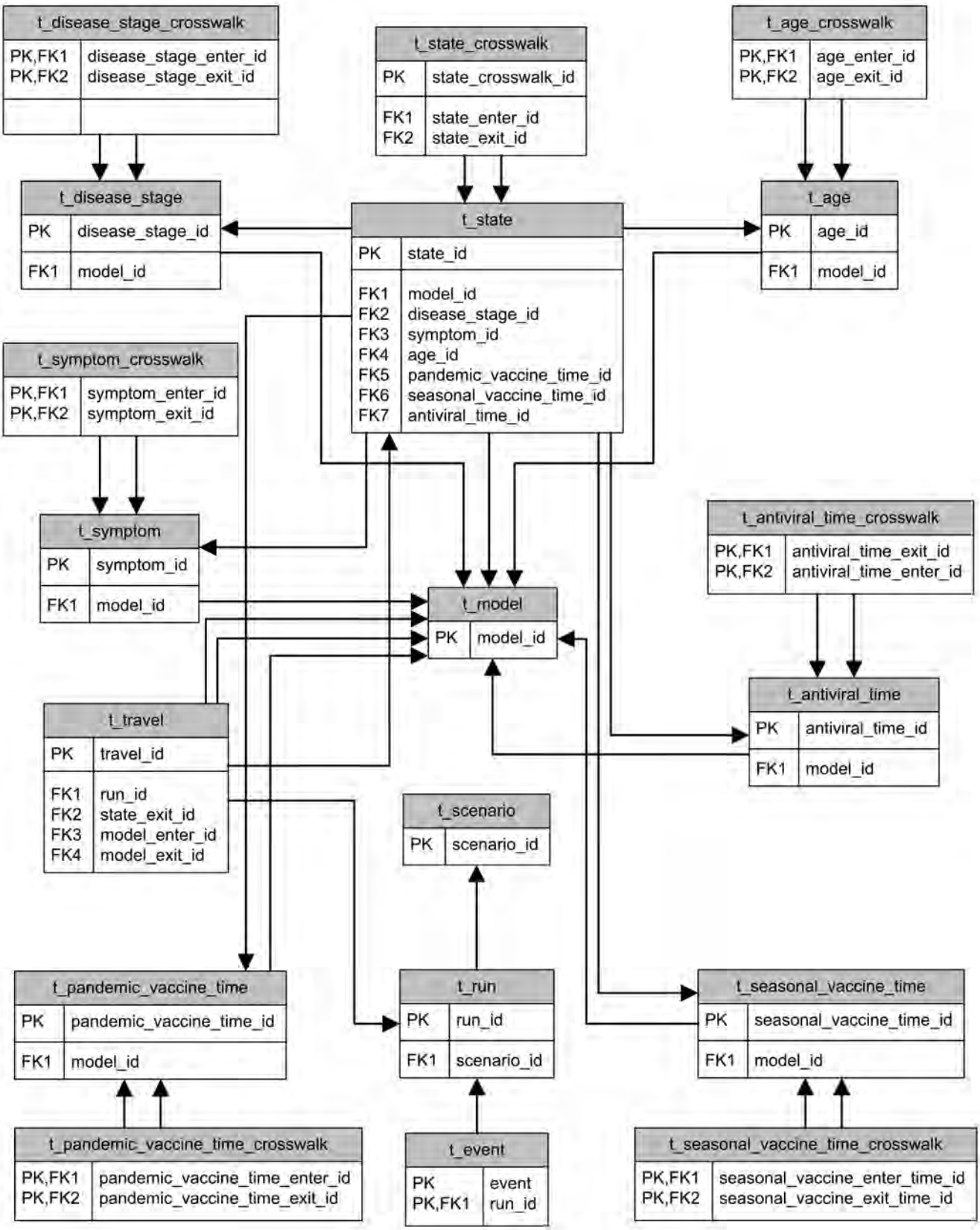


Figure 4. Entity-relationship diagram for the entire database



attributes that the modeler wanted to assign to travelers, the modeler should add a new attribute table to the database and populate it with records listing the attribute values supported by each model. Since this would be a new attribute, an existing model probably would not support any of its values. In this case, a "not applicable" record should be added for the model. Once the attribute table has been created, the modeler should create a corresponding crosswalk table to describe how to convert between each model's values in the new attribute table. In addition to the attribute and crosswalk tables, the modeler should also update the queries that create the t_state and t_state_crosswalk tables so that they account for the new attribute.

A modeler might also decide to modify our proof-of-concept coupled system to exchange other data types besides the traveler. Adding a new data type might require adding new attribute tables, or the modeler might be able to reuse some of the traveler attribute tables. For example, if the new data type were total population size with disease stage as the only attribute, then the traveler disease stage attribute table might be sufficient to describe the new data type. However, if the new data type were vaccine doses with vaccine effectiveness rate as an attribute, then the modeler probably would need to add a new attribute table because vaccine effectiveness is not an attribute of the traveler data type. The modeler should create a crosswalk table for any new attribute tables and then create tables similar to the traveler data type's t_state and t_state_crosswalk tables that list all the unique combinations of the new data type's attribute values. Finally, the modeler should create a table to hold data points of the new data type as they are exchanged by the models, similar to the t_travel table for the traveler data type.

Inserters and Extractors

The models in our proof-of-concept coupled system need to communicate with the database at every time step. Each model records the number of travelers exiting the model, discovers the number of travelers entering the model, and updates the t_event table. We developed auxiliary tools to shift some of the database connectivity tasks out of the models. Specifically, we created auxiliary tools called inserters to handle the task of inserting into the database the number of travelers exiting the models. We created auxiliary tools called extractors to handle the task of extracting from the database the number of travelers entering the models. We did not create auxiliary tools to handle updates to the t_event table, so the models are responsible for performing this task themselves by connecting to the database and communicating with it directly.

We wrote the inserters and extractors in Java, using the open-source Java Database Connectivity drivers to connect to the PostgreSQL database. Each tool provides one well-defined function. For example, one inserter's function is to add records to the t_travel database table indicating the number of people who travel from the GEM to the SE Asia model. Adding records for travel from the SE Asia model to the GEM is a different function and is therefore handled in a different inserter. If we were to later expand the proof of concept by adding a third model—a Mexico City model, for example—then we would write a new inserter for travel from the GEM to the Mexico City model and a new inserter for travel from the Mexico City model to the GEM. If we later were to again expand the proof of concept by allowing the models to exchange another data type in addition to travelers, we would write a new set of inserters for the new data type.

We wrote two inserters for our proof of concept, one associated with the GEM and one associated with the SE Asia model. Similarly, we wrote one extractor associated with the GEM and one extractor associated with the SE Asia model. Since we use the coupled system for exchanging only the traveler data type, each inserter and extractor handles only traveler data.

An inserter in our proof of concept operates by polling the t_event database table, waiting to detect an event indicating that its associated model has written a text file containing outgoing traveler counts. Upon detecting that event, the inserter parses the file, performs any necessary processing on the data it finds, and then inserts records into the t_travel database table. Finally, it updates t_event with an event indicating "new records are available in t_travel." The inserter then resumes polling t_event to repeat the cycle at the next time step. It continues in

this manner until it detects the event indicating "the model run has finished."

Like inserters, an extractor operates by polling the t_event database table. The extractor waits to detect the event indicating that the inserter associated with the other model has added new records to the t_travel database table. Once it detects that event, it uses the database's attribute and crosswalk tables to convert the traveler counts from the other model's representation to its own associated model's representation, performs any necessary processing on the resulting data, and writes the data to a text file. Finally, it updates t_event with an event indicating that it has completed writing its text file. The extractor then resumes polling t_event to repeat the cycle at the next time step. It continues in this manner until it detects the event indicating that the model run is finished.

The primary reason for using inserters and extractors is to isolate the models from the database. This isolation has two benefits. First, it promotes a separation of responsibility in which the models focus on model-related tasks and other tools focus on database-related tasks. Second, it helps reduce the need to update the models when the database structure changes. The database structure may change for many reasons, such as general maintenance and improvement or the addition of data elements to support new or upgraded models. Because the GEM and the SE Asia model are being actively maintained and improved, upgrading the versions currently used in the proof of concept to newer ones is a distinct possibility. If upgrading one model necessitates changes to the database, the model-database isolation means that corresponding changes likely need to be made in the other model's inserter and extractor, not in the model itself.

Models

Broadly speaking, the GEM and the SE Asia model perform the same basic procedure. They step through time in discrete units of 1 day, performing certain tasks during each time step. These tasks include simulating contacts among people in the modeled populations, deciding which susceptible people become infected as a result of those contacts, updating the disease status of all the sick people, and applying any desired interventions, such as vaccinations or quarantines. In addition, the GEM simulates the travel of people from city to city.

A few of the models' tasks enable them to function within the proof-of-concept coupled system. During each time step, the SE Asia model determines which people travel out of its modeled population and writes this information to a text file. It then updates the t_event database table indicating that it has written its file of outgoing travelers. Next, the SE Asia model polls t_event until it detects the event indicating that its extractor has created a text file of incoming travelers from the GEM. Once it detects that event, it parses the file and adds those travelers to its population. From there, it proceeds normally until the next time step, when it repeats the cycle. It continues in this manner until it detects the event indicating that the model run is finished.

The GEM performs similar tasks to work within the coupled system. During each time step, the GEM polls the t_event database table until it detects the event indicating that its extractor has created a text file of incoming travelers from the SE Asia model. Once it detects that event, it parses the file and combines the travelers from the file with the travelers exiting the rest of its modeled cities to determine how many people travel into each city. Next, the GEM writes a text file containing counts of travelers who enter the SE Asia model, after which it updates the t_event table indicating that it has written its file of outgoing travelers. It then proceeds normally until the next time step, when it repeats the cycle. We decided to make the GEM the "controlling model," in that it controls how long the models run. Therefore, when the GEM reaches its terminating condition, it adds an event to t_event indicating that the model run is finished. All other components in the coupled system check for this event to discover when they should terminate.

Unless a model was designed from the beginning to be coupled with other models using our approach, it will certainly require modification before it can function within a coupled system. We made several changes to the GEM and SE Asia model, which we summarize here. This list is provided as an example of

the types of changes a modeler may need to consider when preparing models for coupling. We made the following changes to the GEM:

- We added code to the GEM to implement communication with the coupled system's database. This included connecting to and disconnecting from the database, adding events to the process scheduling table, and polling the process scheduling table to detect the existence of events.
- We added the region represented by the SE Asia model to the GEM's transportation network as a new node. Because many of the SE Asia model's parameters were derived from studies of rural Thailand, for the purposes of the coupled system, we assumed that the region was in rural Thailand. Therefore, we named the node Rural Thailand and incorporated it into the rest of the transportation network by adding a single connection between the new Rural Thailand node and the existing Bangkok node. This network structure implies that all travelers into the SE Asia model come from the GEM's Bangkok node and that all travelers out of the SE Asia model enter the GEM's Bangkok node.
- We added Rural Thailand to the GEM's input file that lists its recognized cities. We added an attribute to this file indicating whether each city is modeled internally by the GEM or externally by another model. If externally, the attribute also indicates whether the city is modeled by an agent-based model or an equation-based model.
- We modified the GEM to distinguish externally modeled cities from internally modeled cities in certain calculations. When calculating the number of travelers exiting an externally modeled city, the GEM waits to discover this information from the coupling infrastructure. When calculating the number of travelers entering an agent-based, externally modeled city, the GEM rounds all travel counts to integers. When updating population totals, the GEM ignores externally modeled cities under the assumption that external models keep track of their own population totals.
- We added code to the GEM that exports the outgoing traveler data to a text file and imports the incoming traveler data from a text file.
- We modified the GEM to allow an initial condition of zero infected people in all of its internally modeled cities. Outside the context of the coupled system, when the GEM was self-contained and did not exchange individuals with another model, this initial condition was not allowed because it generated a scenario in which no one ever became infected, yielding a useless model of disease transmission. However, one scenario we wanted to model within the coupled system was the situation in which the infection arises in the region represented by the SE Asia model and later spreads to the cities represented by the GEM. In this case, an initial condition of zero infected people in all the GEM cities was appropriate.

We made the following changes to the SE Asia model:

- We added code to the SE Asia model to implement communication with the coupled system's database. This included connecting to and disconnecting from the database, adding events to the process scheduling table, and polling the process scheduling table to detect the existence of events.
- The SE Asia model had basic support for people traveling out of its population at every time step. However, it did not support people traveling into its population, so we added code to implement this. After reviewing the SE Asia model source code, we realized that the model assumed a fixed population in many places. Implementing incoming travelers in the intuitive way—creating a new person for each incoming traveler and increasing the population size by one—would require extensive changes to the model, so we developed an alternate method. For each incoming traveler, we chose an existing population member at random and replaced some of that member's attributes, such as disease status, with those of the traveler. For consistency, we modified the existing implementation of outgoing travel using similar logic. Instead of simply removing an outgoing traveler from the population, we replaced the traveler with a new person who retained most of the attributes of the traveler with the exception of a few, such as disease status, that we re-initialized.

- We added code to the SE Asia model that exports the outgoing traveler data to a text file and imports the incoming traveler data from a text file.
- The SE Asia model implemented its own conditions for terminating the model run. Namely, it terminated once it had run for a specified number of days or once its population included zero infected individuals. However, within the coupled system, we wanted it to continue running until the GEM terminated. Therefore, we removed its existing termination conditions and replaced them with the condition that it would run until it detected the model termination event in the t_event database table.
- We modified the SE Asia model so that we could override the file system locations of all its input and output files from the command line.

Results

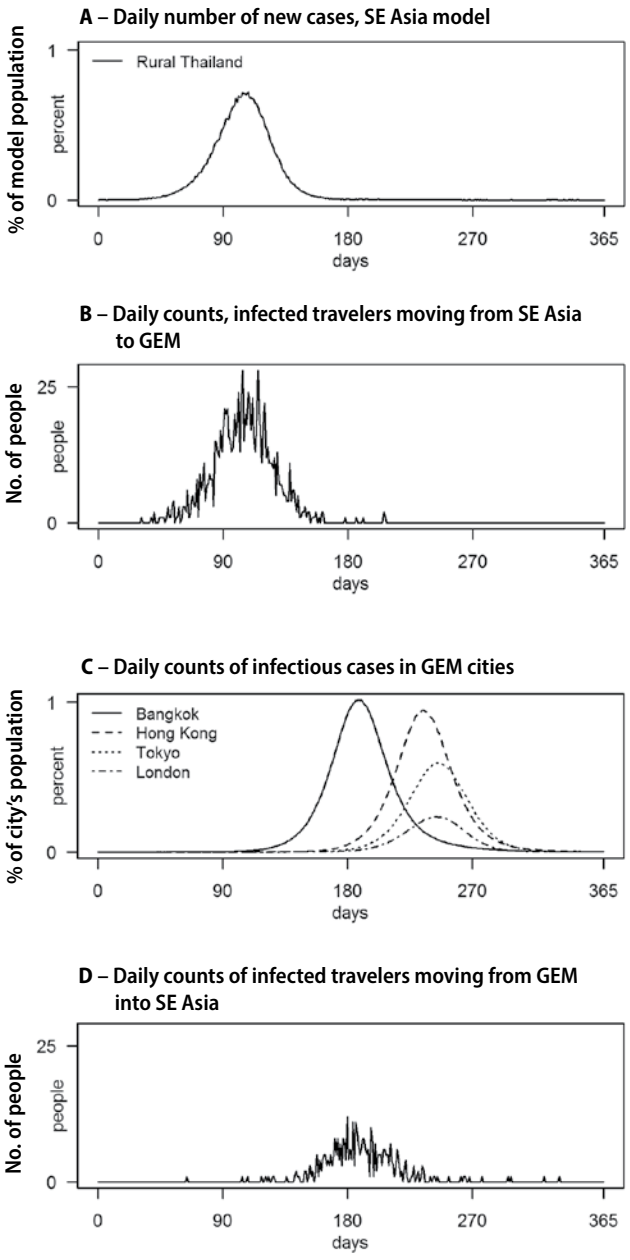
Scenarios

Because the coupled system we developed is a proof of concept only, we did not run it under experimental scenarios with the intent of drawing substantive conclusions. However, we did run it to verify that the coupled models interacted with the coupling infrastructure correctly and that travelers moved between the coupled models appropriately. We present results of the proof-of-concept coupled system from two sample scenarios. We stress that these results are for illustration purposes only.

In the first scenario, we placed the source of the infection in rural Thailand—that is, in the SE Asia model—by setting the SE Asia model to contain 10 initially infected people and every GEM city to contain zero initially infected people. We disabled all interventions (vaccinations, antivirals, travel restrictions, and quarantine) in both models. Figure 5 displays the results of a single iteration of this scenario. Plot A displays the daily number of new infectious cases in the SE Asia model as a percentage of the model's total population size. Plot B displays the daily counts of infected people traveling from the SE Asia model into the GEM. Plot C displays the daily number of new infectious cases in selected GEM cities as a percentage of each city's total population size. Plot D displays the daily counts of infected people traveling from the GEM into the SE Asia model.

From these plots, we see that the infection begins in the SE Asia model and eventually spreads to the GEM as infected travelers enter Bangkok from rural

Figure 5. Results from the first scenario of the proof-of-concept coupled system: Rural Thailand as infection source



GEM = Global Epidemic Model; SE Asia = Southeast Asia Model

Thailand. After Bangkok, the infection later spreads to other GEM cities. While we did not display all 155 GEM cities in the figure, we did confirm that Bangkok was the first GEM city to experience an infected case.

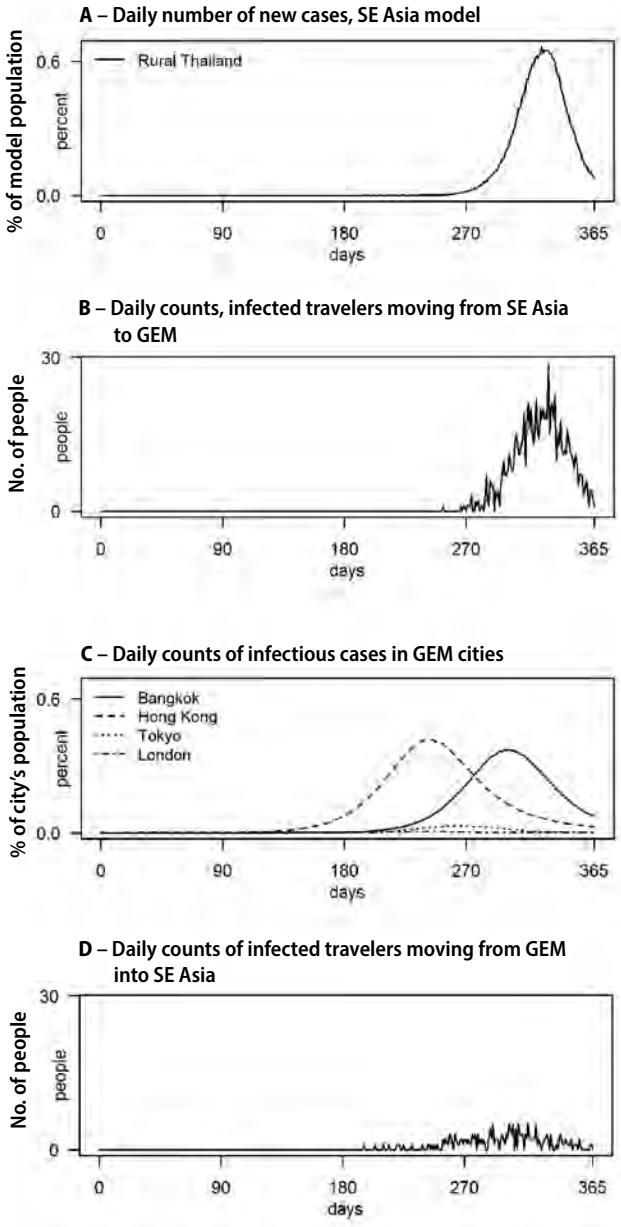
In the second scenario, we placed the source of the infection in Hong Kong, a city modeled by the GEM, by setting Hong Kong to contain 10 initially infected people and every other GEM city as well as the SE Asia model to contain zero initially infected people. We applied vaccinations in the GEM but no other interventions in either model. Figure 6 displays the results of a single iteration of this scenario. The plots in Figure 6 are structured in the same way as those in Figure 5.

From these plots, we see that the infection spreads from Hong Kong to other GEM cities, eventually making its way to Bangkok. Once in Bangkok, the infection crosses over to the SE Asia model as infected travelers enter rural Thailand from Bangkok. In the selected GEM cities as well as in rural Thailand, we see that the peak of the daily number of new infectious cases is roughly 40 percent lower than in the first scenario. We suspect this is because we applied vaccines in the GEM. If this is true (and we would need to perform real experiments with multiple iterations to prove that it was), a notable result is that the reduction in the number of cases in the GEM translated to a similar reduction in the number of cases in the SE Asia model. This implies that even though we applied no interventions in the SE Asia model, it was still affected by the interventions in the GEM, demonstrating an effect of coupling these models.

Areas for Improvement

We developed the proof of concept as a way to illustrate our general model coupling approach. Our goals were to highlight advantages of the approach and to discover potential problems a modeler might encounter when applying the approach. We did not attempt to resolve all the problems we encountered. Here we describe several such unresolved problems as areas where a modeler may want to improve the proof of concept before using it for serious experiments.

Figure 6. Results from the second scenario of the proof-of-concept coupled system: Hong Kong as infection source



GEM = Global Epidemic Model; SE Asia = Southeast Asia Model

- **The total population of the system declined over time.** Our implementation of incoming and outgoing travel in the SE Asia model yielded a constant population size in that model, regardless of any imbalance between total travel into and out of the model. However, because the SE Asia model and the GEM allowed different subsets of people to travel, there was indeed a travel imbalance in which more people entered the SE Asia model than exited it, or, conversely, more people exited the GEM than entered it. As a result, the GEM reported a net decline in population size while the SE Asia reported a constant population size, producing an overall decline. The travel imbalance in our illustrative examples was about 200,000 people, which was only 0.03 percent of the initial GEM population but 40 percent of the SE Asia model population. The SE Asia model should probably be fixed so that its population size is properly adjusted to reflect the travel imbalance.
- **The SE Asia model reported misleading values for statistics that involved the total population size.** This anomaly also occurred because our implementation of incoming and outgoing travel in the SE Asia model failed to properly account for a travel imbalance. While some values collected by the model did not reflect the travel imbalance, other values did. Statistics that are computed using a combination of these types of values are likely to be incorrect.
- **The number of travelers exiting the GEM did not always equal the number of travelers entering the SE Asia model.** In the process of converting the traveler data type from the GEM's representation to the SE Asia model's representation, integer counts of the number of travelers were sometimes split into floating point counts. Because the SE Asia model is agent-based, the floating point counts had to be reconstructed back into integer counts. The reconstruction algorithm did not guarantee that the sum of the final integer counts would equal the sum of the original integer counts. The algorithm should probably be replaced with one that does preserve the number of travelers.
- **Updating the database tables to define the models' data representations was a manual and sometimes cumbersome process.** To complete the limited updates that our test scenarios required, we had to edit the database tables directly, either by using database administration tools or by writing small programs. As it stands, the user would have to execute a sequence of SQL commands to regenerate the t_state and t_state_crosswalk tables after any change to one of the tables from which they were derived. A user-friendly interface could be developed to remove some of the manual steps and to make the remaining necessary manual steps easier.
- **The current database design supported only a single data representation for each model.** This could be a problem if a modeler wished to run multiple scenarios where different scenarios required different data representations. For example, suppose the modeler wanted to run one scenario with age groups defined as [0–5, 6–17, 18–64, 65+] and another scenario with age groups defined as [0–17, 18+]. Since age group is an element of the model's data representation, only one age group definition can exist within the database at a given time. The database design should probably be altered to allow multiple data representations for each model.

Discussion

We have presented a general approach for coupling multiple simulation models and have demonstrated our approach by describing a proof-of-concept coupled multiscale system. The approach enables models to run concurrently while exchanging data with each other and is general enough to be applied to a range of situations. In the approach, models communicate with each other indirectly by using a database as a common access point for exchanged data. Models may also access the database indirectly if the exchanged data are routed through model-specific auxiliary tools. The approach also takes advantage of the database for clock synchronization and process scheduling.

We developed this model coupling approach with the intent of meeting several goals that we considered important to producing a good coupled system.

Below, we describe these goals and evaluate how well our approach meets each of them.

- **Support models written in any programming language.** Models may be written in programming languages such as C, C++, Java, MATLAB, and Fortran. We wanted the flexibility to mix and match any of these models, regardless of the language used. The only requirement our approach places on the programming language is that it must have the ability to read from and write to the database chosen for the coupled system. While this may rule out some combinations of databases and languages, database access libraries exist for most common databases and languages.
- **Be implementable using free and open-source software (FOSS).** Requiring proprietary software might limit adoption of the approach because some researchers have philosophical objections to using proprietary solutions or simply cannot afford them. Our approach involves the implementation of two primary components: the database and the inserters/extractors. Multiple high-quality, FOSS solutions exist for each of these components. At the same time, our approach does not require FOSS solutions, so researchers are also free to use any of the multiple high-quality, proprietary solutions that exist for these components.
- **Support both agent-based and equation-based models.** Similar to the argument for supporting multiple programming languages, we wanted the flexibility to mix and match models regardless of their implementation details. Our approach places no restrictions on whether the coupled models are agent-based or equation-based. However, mixing agent-based and equation-based models in a coupled system may require modelers to take special care when designing the database or when preparing the models for coupling. This is because the continuous nature of equation-based data may not be suitable in the discrete context of agent-based data.
- **Minimize data loss resulting from the exchange of data between models that have varying assumptions and features.** Models developed by different research groups for different purposes are likely to have different assumptions and features. When these varying assumptions and features impact the data exchanged by the models, information may be lost. We did not want to place limitations on the data exchange process that would contribute any additional loss of information beyond what is produced by incompatibilities in the models themselves. Our approach's primary obstacle in this regard is being able to represent the data type of the exchanged data in a set of database tables. This may be more difficult in some cases than in others. However, given that our approach allows any number of tables with any number of fields, it is likely that any data type can be adequately represented in some way.
- **Support the coupling of an unlimited number of models.** While we considered the coupling of just two models a good first step, we wanted an approach that would allow any number of models to be coupled. Coming from our perspective of infectious disease modeling, we envisioned coupling numerous models that examine disease transmission in small geographic areas to produce a single model covering a large geographic area. In our approach, any number of models may communicate with the database, so there is no limit on the number of coupled models.
- **Couple the models loosely.** Loosely coupled models assume little, if anything, about each other. Because a model's code is not affected by the presence or absence of other models, it is easier to swap models into and out of the coupled system. Our approach loosely couples all the models by having them communicate indirectly through the database instead of directly with each other. The advantages of loose coupling become more pronounced as more and more models are introduced into the coupled system. Tight coupling would require each model to include code specific to every other model, while loose coupling avoids, or at least limits, this requirement.
- **Minimize the number of changes required in models when they are being prepared for the coupled system.** The loose coupling between the models helps to eliminate many of the changes that otherwise would be required when a model was being prepared for the coupled system. However, some changes are unavoidable. Because the models communicate directly with the database

for the purposes of process scheduling and clock synchronization, the modeler must insert code at appropriate places to implement this database communication. The modeler most likely needs to insert code at appropriate places so that the model can export data into the coupled system and import data from the coupled system. The specifics of the exchanged data and the model's role in the coupled system may imply additional changes.

- **Scale effectively as more models are coupled.** Because we anticipated coupling many models, we were concerned about the potential for performance degradation as each new model was added. We wanted an approach that would not contribute additional performance losses beyond that imposed by the hardware. Our approach routes all communication through a database, which does highlight the database as a potential bottleneck. Fortunately, improving database performance is a well-developed discipline, and many techniques for improvement are available if database performance becomes a problem. In addition, because the models, inserters, and extractors all run as separate processes in our approach, it should be fairly straightforward to distribute them across a multiprocessor architecture for further performance gains. We wanted to achieve scalability in the context not only of performance, but also of development time. We wanted the effort required to couple each new model to be approximately equal instead of being proportional to the number of models already in the system. Our approach should achieve this through its loose coupling between models and between the models and database.

A potential limitation of our approach is that we have applied it to only one model coupling scenario, and so we may be unaware of deficiencies that broader testing would reveal. For example, we have not attempted to use the approach with discrete-event models or in a distributed computing environment. Despite this limited testing, our experiences with other models give us confidence that the approach can be applied successfully in other contexts, as well.

Another limitation has a much broader reach than the approach itself. While coupling models often seems like a good idea, the task must be undertaken with caution. Salt (2008) argues that it is one of the seven persistent mistakes of simulation modeling. To be meaningfully combined, models should be conceptually compatible. Because different models are usually developed by different research groups for different purposes, any models chosen as a set for coupling likely differ in the features they implement and the simplifying assumptions they make about their modeled phenomena. The utility of a coupled system is limited by the impact of these differences on the compatibility of the models.

The two models we chose for our proof of concept were developed by different research groups and do indeed implement different features and make different assumptions. For example:

- The two models make different subsets of the population eligible to travel to other cities.
- The GEM assumes that all infectious people exhibit symptoms, but the SE Asia model assumes that some portion of the infectious people may be asymptomatic.
- The SE Asia model offers antiviral medication as a possible disease intervention, while the GEM does not.
- Both models offer vaccination as a possible disease intervention, but their implementations differ in the timing, frequency, and effects of vaccination.
- The SE Asia model uses age groups and social networks to structure its population, but the GEM does not.

We have not attempted here to address the issue of whether these models are compatible enough for useful coupling. Instead, we leave that decision to future modelers who wish to move beyond the proof-of-concept stage in coupling the GEM and the SE Asia model. Evaluating the models' compatibility while considering the goals of the proposed coupling will be a critical step in constructing a useful coupled system.

Assuming compatible models can be identified, our model coupling approach could be applied to many simulation fields, including infectious disease, military operations, biological systems, multimedia environmental modeling, and manufacturing processes. The approach could be applied to multiscale modeling, or to models that operate at the

same scale but that examine different locations, such as geographic areas or internal organs.

Our focus so far has been retrospective model coupling—that is, locating existing models and then determining how to couple them. However, our approach could also be applied to prospective model coupling. Here, modelers would reverse the process by first identifying a problem domain that could be simulated by coupling multiple models that do not yet exist, then developing both the models and the coupling infrastructure with interoperability as a requirement from the beginning. Prospective coupling would require more up-front coordination to ensure all the models would work with each other and with the coupling infrastructure, but would have the benefit of helping eliminate the model incompatibilities that might limit retrospective coupling.

References

Bobashev, G. V., Goedecke, D. M., Yu, F., & Epstein, J. M. (2007). A hybrid epidemic model: Combining the advantages of agent-based and equation-based approaches. In S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew, & R. R. Barton (Eds.), *Proceedings of the 2007 Winter Simulation Conference* (pp. 1532–1537). Washington, DC: Institute of Electrical and Electronics Engineers (IEEE) Press. Retrieved June 28, 2013, from http://www.informs-sim.org/wsc07papers/186.pdf

Brandmeyer, J. E., Solano, E., Zerbonia, R. A., Gao, G., Xin, L., Tian, C., & Jiawen, J. (2004, October). *Development of an air quality management decision support system for Beijing, China.* Paper presented at the Community Modeling and Analysis System (CMAS) Models-3 Conference, Chapel Hill, NC.

Bulatewicz, T. (2006). *Support for model coupling: An interface-based approach* (Unpublished doctoral dissertation). University of Oregon, Eugene, OR.

Eddy, D. M., & Schlessinger, L. (2003). Archimedes: A trial-validated model of diabetes. *Diabetes Care, 26*(11): 3093–3101.

Epstein, J. M., Goedecke, D. M., Yu, F., Morris, R. J., Wagener, D. K., & Bobashev, G. V. (2007). Controlling pandemic flu: The value of interna–tional air travel restrictions. *PLoS ONE, 2*(5): e401. doi:10.1371/journal.pone.0000401

Ford, R. W., Riley, G. D., Bane, M. K., Armstrong, C. W., & Freeman, T. L. (2006). GCF: *A general coupling framework. Concurrency and Computation: Practice and Experience, 18*(2): 163–181.

Guo, W., & Langevin, C. D. (2002). *User's guide to SEAWAT: A computer program for simulation of three-dimensional variable-density ground-water flow.* Techniques of Water-Resources Investigations of the United States Geological Survey, Book 6, Chapter A7. Tallahassee, FL: US Geological Survey. Retrieved June 28, 2013, from http://fl.water.usgs.gov/Abstracts/twri_6_A7_guo_langevin.html

Hoheisel, A. (2002). *Model coupling and integration via XML in the M3 simulation.* University Park: CiteSeerx beta website, College of Information Sciences and Technology, University of Pennsylvania. Retrieved June 28, 2013, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.5852

Institute of Electrical and Electronics Engineers (IEEE). (2000). *IEEE standard for modeling and simulation (M&S) high level architecture (HLA)—Framework and rules.* IEEE Standard No. 1516-2000. New York: Author.

Jobson, H. E., & Harbaugh, A. W. (1999). *Modifications to the diffusion analogy surface-water flow model (DAFlow) for coupling to the modular finite difference ground-water flow model (ModFlow).* U.S. Geological Survey Open-File Report 99-217. Reston, VA: US Geological Survey. Retrieved June 28, 2013, from http://water.usgs.gov/nrp/gwsoftware/modflow2000/OFR99-217.pdf

Larson, J., Jacob, R., & Ong, E. (2005). The model coupling toolkit: A new Fortran90 toolkit for building multiphysics parallel coupled models. *International Journal of High Performance Computing Applications, 19*(3): 277–292.

Longini, I. M., Jr., Nizam, A., Xu, S., Ungchusak, K., Hanshaoworakul, W., Cummings, D. A., & Halloran, M. E. (2005). Containing pandemic influenza at the source. *Science, 309:* 1083–1087.

Salt, J. D. (2008). The seven habits of highly defective simulation projects. *Journal of Simulation, 2*: 155–161. doi:10.1057/jos.2008.7

Riley, S. (2007). Large-scale spatial-transmission models of infectious disease. *Science, 316*: 1298–1301.

Swain, E. D., & Wexler, E. J. (1996). *A coupled surface-water and ground-water flow model (MODBRANCH) for simulation of stream-aquifer interaction*. Techniques of Water-Resources Investigations of the United States Geological Survey, Book 6, Chapter A6. Washington, DC: US Geological Survey. Retrieved June 28, 2013, from http://pubs.usgs.gov/twri/twri6a6/html/pdf.html

Warner, J. C., Perlin, N., & Skyllingstad, E. D. (2008). Using the model coupling toolkit to couple earth system models. *Environmental Modelling & Software, 23*: 1240–1249.

Zacharias, G. L., MacMillan, J. , & Van Hemel, S. B. (Eds.). (2008). *Behavioral modeling and simulation: From individuals to societies*. Washington, DC: The National Academies Press.

Acknowledgments

We thank D. Michael Goedecke for providing the Global Epidemic Model's source code and for his helpful insights into the model's implementation details. We also thank Ira M. Longini Jr. for providing the Southeast Asia model's source code. This research was supported in part by the Modeling of Infectious Disease Agents Study (MIDAS), grant 1 U24 GM087704 from the U.S. National Institute of General Medical Sciences (NIGMS).

www.rti.org/rtipress **RTI Press publication MR-0026-1309**