Integration of urban runoff and storm sewer models using the OpenMI framework
Ying-Po Liao, Shiu-Shin Lin and Hung-Sung Chou

ABSTRACT
This paper presents an integrated modeling approach that combines the legacy urban runoff and the legacy storm sewer models with data exchange at runtime to simulate sewer flows with real-time rainfalls. A software system called IRES has been developed to demonstrate the proposed approach. IRES uses the RUNOFF and the EXTRAN block of Storm Water Management Model (SWMM) as the legacy urban runoff and storm sewer models, respectively. This study uses the Open Modeling Interface and Environment (OpenMI) framework, which is the standard interface for model integration, as the platform for dynamic data exchange in IRES. The Jhong-Kang drainage system located in Taipei, Taiwan, is taken as the case study to verify the proposed approach. Simulated results show that IRES is valid, and the proposed approach can be successfully applied to legacy model integration with dynamic data exchange.

Key words | OpenMI framework, runoff model, storms sewer model, Storm Water Management Model (SWMM)

INTRODUCTION
Water environments have an integral and undividable nature. Because there has never been ‘a unique model for everything,’ hydrological models typically target a specific subsystem, i.e. the storm sewer, river, rainfall-runoff, and inundation models. However, to accurately model the physical behaviors in an urban storm event, the mass and momentum interchange among such subsystems should be carefully considered. Taking an urban storm sewer system represented by a sewer model as an example, the input discharge of the sewer system is usually caused by the outflows of upper catchments that mostly have been modeled from another rainfall-runoff model. Thus, incorporating sewer model with the rainfall-runoff model is an integrated solution for urban storm simulations.

Model-linking for subsystem models is a straightforward strategy that has been popularly used to resolve practical engineering cases (Argent 2004; Bulatiewicz et al. 2010; Castronova & Goodall 2010). However, in addition to the technical problems of model-linking, its ability to correct physical operations has also been a big challenge. There are two commonly used model-linking strategies. One is the file-based connection, which is done through file transfer as in the case when output data from rainfall-runoff are adapted to input data for the sewer model. The Storm Water Management Model (SWMM) (Huber & Dickinson 1988) is an example of this type. In SWMM 4.0, linking different modules can be done by setting the input file headers. Simulation is independently and sequentially performed by running rainfall-runoff to obtain output flows of the manhole and then imposing the sewer model as input discharges. The other commonly used strategy is the model-based connection, which dynamically interchanges data during system runtime (Argent 2004; Castronova & Goodall 2010). The two approaches imply different operations.

Figure 1 illustrates a conceptual model of storm sewers that can be divided into two components and are respectively modeled from the rainfall-runoff and sewer models. Mass and momentum transfer between the two models are tightly coupled with each other at every time step. This means that once an output discharge is simulated by the
rainfall-runoff model at any time, even by increased amounts of rain in the future, the sewer model is supposed to immediately take the output discharge as its corresponding input flows and consequently perform sewer modeling.

The file-based connection provides static-only linking, which implies that the sewer model can be started only after the rainfall-runoff model is finished. This approach has its limitations in reflecting real physical operations because two systems may feed physical quantities to each other. Moreover, this approach is not applicable to a real-time system in that the static-only linking is always difficult to tackle if dynamical real-time data are used.

A more advanced approach to model linking, such as the model-based connection, is required to take correct physics into account. This means that physical quantities should be allowed to transfer between different models in a time-step-by-time-step manner. In this case, integrating the rainfall-runoff and sewer models into a single runtime-based or runtime process (Tangenbaum 2007) allows interchange flow discharges between the two models coupled with each other while simulating sewer flows.

Figure 1 | Conceptual view of rainfall-runoff and sewer model integration.

A model-based connection is more complicated than in a file-based connection. It tightly couples different models, independent runtime processes, in the same runtime process, holding the simulation in fidelity. Integrating heterogeneous models into a single runtime process usually deals with in-process communications or message-passing within different models, thus involving some complicated issues, such as development of models through different programming languages, as well as different spatial and temporal discretizations, units, numerical iterative schemes, message-passing mechanisms and interfaces, and so on.

Fortunately, the Open Modeling Interface and Environment (OpenMI) framework has provided standard interfaces for both message passing and model linking (Moore & Tindall 2005; Gregersen et al. 2007; www.OpenMI.org). Standard interfaces provided by OpenMI allow in-process data exchange among heterogeneous models on a time-step-by-time-step basis.

The main objective of this paper is to demonstrate the applicability of the integration of the legacy urban runoff and storm sewer models using the OpenMI framework, thereby providing the basis for dynamic message passing. SWMM, developed by the United States Environmental Protection Agency, is adopted in the present work as the legacy software components. In solving kinematic wave equations, the RUNOFF block of SWMM is adopted as the rainfall-runoff model to provide the runoff hydrograph from subcatchments. By solving full de Saint-Venant equations (Chaudhry 1993), the Extended Transport (EXTRAN) block, which is a well-developed and widely used sewer model (Park & Johnson 1998; Zaghloul 1998; Lin et al. 2006; Tsihrintzis et al. 2007; Lin et al. 2010), is adopted to solve the hydrodynamics of sewer flows in which the inflow boundary conditions are based on the output results of RUNOFF. First, two models are respectively modified and wrapped up to independent OpenMI-compliant components. Over each time step, sewer flow dynamics (solved using EXTRAN) is updated according to timely upstream inflows. Such inflows also rely on data containing the timely rainfall input and consequent outflows from subcatchments, which are generated by RUNOFF. Dynamic model-linking simulations are then performed using OpenMI standards and tools.
To demonstrate the applicability of the proposed strategies, a simulation system called IRES is designed and developed for this research. IRES can be used to study a number of practical engineering cases. Results compared with static-linking strategy can be used for verification. The functions of IRES are mainly focused on: (1) conforming SWMM models to OpenMI interfaces through minimized re-engineering, and (2) model linking and integrating the urban runoff and storm sewer legacy models using OpenMI to achieve real-time simulations for urban storm events.

How to select OpenMI and SWMM versions may be worthy of concern. The current release version of OpenMI is 2.0, and the previous release version is 1.4. OpenMI 2.0 renews the standards that mainly focus on flexibility and usability. The main principles and frameworks, however, do not change in both versions (www.OpenMI.org). OpenMI 1.4 is used for the system demonstrations and the detailed descriptions in this paper. However, since the tasks involved in this study are aimed at a generally applicable approach and do not bind to a specific version, later OpenMI versions without significant changes may also be used.

Because the main objective of this paper is to propose an approach for the integration of legacy runoff and storm sewer models using the OpenMI framework, SWMM-RUNOFF and SWMM-EXTRAN are the roles selected to serve as the examples. SWMM 4.0 is developed by Fortran 77 based on traditional procedure calls and performed using simple text input/output files in a command-line mode, which is meant to be the ‘legacy model’ and selected as the examples in this paper. Again, this study demonstrates an applicable approach to integrate legacy models using OpenMI rather than aim for the linking of SWMM-RUNOFF and SWMM-EXTRAN. How to link SWMM itself is not the scope of this research because SWMM 5.0 has well resolved this issue by facilitating the rewriting of all modules and the development of user-friendly graphical user interfaces (GUIs). In addition, in this research, the proposed integration approach is fully dynamical and prepared for a real-time system. It implies some necessary re-engineering for legacy models to make runoff generation and runoff transport able to be exchanged between models along with every time step. The module linking in SWMM 5.0, on the other hand, is not significantly improved and retains static-only strategy, which implies the difficulty of achieving a real-time simulation system. More details are provided below under Methodology.

The rest of this paper is organized as follows. The Methodology section discusses the methodology. The next section provides a brief review of the OpenMI framework. The following section presents details of the design and implementation of IRES. The Applications section describes the performance of the applications of IRES for practical engineering cases and discusses the results. The final section discusses the conclusions.

**METHODOLOGY**

One of the most common applications of model-linking is real-time simulation (Lin et al. 2006). Real-time simulation for urban storm events provides data that can facilitate early warning and decision making. One example of this function is the real-time simulation of sewer flows and inundations according to timely rainfall input, i.e. the preprocessed real-time rainfall data are used as the inputs for the rainfall-runoff model. Sub-catchment outflows (junction discharge) from that model are then computed based on the timely rainfall input as well as the previous initial conditions that rely on catchment flows in previous time steps. The calculated outflows are accordingly streamed as updated boundary conditions into the sewer system through the junctions. The sewer model then simulates the sewer flows by using junction discharge and the initial conditions given by the flow conditions in the last time step. Consequently, the timely junction water stages are calculated.

Figure 2(a) shows the concept of file-based connection using legacy models, such as SWMM. In this example, the SWMM-RUNOFF and SWMM-EXTRAN blocks are employed as the rainfall-runoff and sewer models, respectively. For most of the legacy models, input data are fed by one or more files, thereby accordingly generating the output file. The procedure is sequential, and models are independently executed. The only method for data exchange between models is the piping of output files from the previous model into the subsequent one as inputs, thus ensuring one-way data flow.
Obviously, this approach cannot take over the linking among: (1) rainfall data sources, (2) the rainfall-runoff model, and (3) the sewer model on a timely basis. The independent execution of legacy models implies that all inputs are always pre-defined over simulation time horizon. Taking the SWMM-RUNOFF as an example, the total simulation horizon and the off-line rainfall event is obtained without the flexibility of message passing and the capability of receiving real-time rainfall.

Figure 2(b) shows the approach of model-based connection for model integration proposed in the present work, as well as the approach adopted and implemented in the IRES system. Instead of a file-based, sequential, and one-way data flow approach, the proposed strategy integrates models in a single runtime process rather than as a batch of sequential tasks. Through runtime communications, data exchange and corresponding simulations can be invoked on a timely basis. In this case, SWMM-EXTRAN is run on-line to solve real-time sewer flows, resulting in junction stages at each time step. Each time step invokes the SWMM-RUNOFF model to calculate junction inflows at the corresponding time with real-time rainfall inputs. Flow conditions in previous time steps also are used to simulate the storm sewer flow.

The challenge of using this approach lies in holding both models on standby mode at any time. After resolving the outflows from a real-time rainfall input at time \( t \), SWMM-RUNOFF has to be suspended and must wait for the next request at time \( t + 1 \). SWMM-EXTRAN then simulates the process of piping in sewer flows estimated by SWMM-RUNOFF as the junction inflows at time \( t \). To step forward to time \( t + 1 \), SWMM-EXTRAN has to be suspended and calls SWMM-RUNOFF to acquire the junction inflow at time \( t + 1 \). SWMM-RUNOFF then resumes and computes the corresponding catchment outflows from the new rainfall input at time \( t + 1 \). The resuming-suspending interworking implies that both models should ‘remember’ the inputs and the state variables to restart simulation at any time \( t \). The ability to restart simulation in the said manner is called ‘hot start’ mechanism.

Alternative iterations between models in a single runtime process allow dynamic data exchange on a time-step-by-time-step basis, thus making real-time simulation for storm events possible. The OpenMI framework provides holistic support and tools through which the proposed model-linking approach can be used. However, legacy models, such as SWMM 4.0, are designed as inseparable ‘black boxes’ that run throughout the whole simulation, implying that suspending models to wait for data exchange at anytime is almost impossible unless significant technical changes technical have been made.

OpenMI FRAMEWORK

Introduction to the OpenMI framework

OpenMI defines a set of interfaces that allow hydrological and environmental models to exchange data at runtime, particularly on a time-step-by-time-step basis. OpenMI is funded by the European Commission’s Fifth Framework program under the HarmonIT project. OpenMI provides the generic operational linking mechanisms that enable OpenMI
components to exchange data while running. It also facilitates the creation of OpenMI-compliant components from existing model codes as well as the combination of these components into the integrated modeling system.

The design and implementation of OpenMI follow the object-oriented methodologies, that is, all specifications and technical documents are not only drawn through standard Unified Modeling Language (UML) diagrams and annotations (Rumbaugh et al. 2010), but also conform to the conventions of object-oriented analysis and design. The scope of OpenMI standard interfaces is first given out in a document series, including requirement analysis and Use Cases.

The most significant principle of OpenMI is to link existing models through minimized re-engineering, thereby enabling these models to communicate with each other, i.e. exchanging data or state variables over runtime. A conventional model application shown in Figure 3 possibly provides the user interfaces to facilitate necessary data and parameter inputs for simulations by graphical widgets. Thus, an input file is generated and is fed into the model kernels, called simulation ‘engines’, that are triggered by the user interface and run throughout the simulation time horizon. Afterwards, all output files are generated and visualized through the applications.

The engine is the kernel part of model computation. An ‘engine component’ defined in OpenMI isolates this kernel engine as an independent ‘instance’ along with a set of well-defined interfaces to accept or provide data over runtime. The design and definition of standardized ‘engine interface’, which comprise the most significant factor of seamless data exchange among models, are the major achievements of OpenMI. A ‘linkable component’ defined in OpenMI is a software component for an existing model, which implements the standard interface defined by OpenMI to dynamically exchange data with other engine components, called an OpenMI-compliant engine, that also implement the same interface.

In short, OpenMI is neither a data model specification nor an integrated modeling system – it is actually an interface-based open standard. Instead of sequential linkage, the mechanism of model-linking provided by OpenMI enables models to simultaneously run and exchange data based on a time-step-by-time-step approach, thus representing an accurate physical process interaction.

Model-linking interfaces

Model-linking is a sophisticated procedure that involves the following important issues: (1) designing the model-linking interfaces to accurately trigger actions for model interchange, and (2) designing message-passing interfaces to define the descriptions for data exchange. For the first issue, the solution designed in OpenMI is called a ‘pull mechanism’. A target component, runtime computations of which rely on the other input data, actively inquires another source component to secure necessary data by specifying a particular physical quantity at a particular location and time. If the inquiry request has been invoked, the source component has to compute the necessary values to send back on a real-time basis. This procedure is designed as the ‘GetValue( )’ method in OpenMI. The ‘linear chain’ examples presented in Figure 4 show that models are sequentially linked. Each component calls for its upstream component to complete the computational procedure, thus providing the results to its downstream component. For example, model A is triggered to initiate the simulations. Supposing that, somewhere, this model needs other inputs at runtime, model A suspends all works and requests its upstream model to provide the necessary data by calling the ‘GetValue( )’ method to invoke model B work up. Accordingly, there are two possible scenarios: when model B is able to provide the results with and without other dependences. If, somewhere, model B needs further inputs

![Figure 3](https://iwaponline.com/jh/article-pdf/14/4/884/386822/884.pdf)
provided by upstream model C, it will repeat the similar procedures to fetch necessary data and complete the computations to feed back to model A. Otherwise, if the data has already been calculated, model B would provide the data from its buffer. Once model A has successfully secured enough data to continue operation, it resumes running. A request for other components only take place when necessary to trigger the corresponding computing procedures. This mechanism designed in OpenMI is called ‘pull-based’ or ‘pull-driven’ mechanism.

To realize the ‘pull-driven’ mechanism in OpenMI, ‘Link’ and ‘Linkable Component’ have been conceptualized and designed as the ILink and ILinkablecomponent abstract interfaces, respectively. To enable data exchange, an ILink relates two ILinkablecomponent: one for the source component as the provider and the other for the target component as the receiver.

The ILink interface in OpenMI has a one-way direction from the source component toward target component. Semantically, any ILink refers only to a single physical quantity, meaning that implementing two or more directional message passing, i.e. data exchange between two inter-iterative components, requires the allocation of multiple ILink objects.

Basically, data exchange for all components is done through ILinkablecomponent. Several typical instances of an ILinkablecomponent in this context could include the following: (a) river simulation, surface runoff simulation, or groundwater simulation engines; (b) a measurement device that has to be accessed online on a real-time basis; (c) monitor database consisting of historical data; and (d) data-driven model, i.e. artificial neural networks model.

**Message-passing interfaces**

The design of message-passing interfaces in OpenMI is based on the 4-HW principle of data description (Where, When, What, and How). Describing one value from any data set, which may include identification, geometry definition, period of occurrence, physical quantity, dimension, unit, and conversion operations to other quantities, is required. Several abstract interfaces have been designed
based on the 4-HW principle to describe data exchange and message-passing.

1. **Where.** Identifying geo-information for any data values in OpenMI follows a finite element-like description, which consists of a topologically ordered ‘element set’.

2. **When.** Describing the temporal information for exchange of values in OpenMI is based on two concepts, namely, time stamp and time span. Time stamp represents a single point in a time series, whereas time span indicates the happening period of exchange values from beginning to end.

3. **What.** The physical meaning of an exchanged value is represented by a physical quantity plus a proper unit and dimension. For example, an exchanged value can be described as ‘20 m water level’, where ‘water level’ refers to the physical quantity, the meter (m) refers to the unit, and its dimension is \([L]\).

4. **How.** This concept describes the manner of data exchange among the other components. In model linking, physical quantities are exchanged between two different models. However, even for the same physical quantities, they are probably represented by respective computational element sets in two models. Seamless exchange of data between two models requires the data operations between the different element sets, i.e. interpolations.

## IMPLEMENTATION

**Requirement analysis of IRES**

To achieve the real-time simulation of a storm sewer system in an urban area as well as to develop a runtime environment that allows rainfall data to be inputted on an hourly basis as discussed above in the Introduction and Methodology sections, the rainfall-runoff and the storm sewer models are integrated using SWMM-RUNOFF and SWNN-EXTRAN, respectively. Moreover, the IRES system is developed to demonstrate the works and the availability of the solutions. The requirements are listed as follows:

1. Hourly rainfalls or multiple-hour rainfalls that can be inputted on a timely basis.

2. Sub-catchment outflows that can be calculated on a real-time basis with the latest rainfall and streamed into the sewer junctions as the upper boundary conditions (junction inflows).

3. Sewer flows that can be simulated with the latest junction inflows.

The integration of the SWMM-RUNOFF and SWMM-EXTRAN model with file-based connection is not a novel idea. However, the process of combining legacy models, which are based on a time-step-by-time-step runtime simulation and not on a simple file-based linking, is a big challenge. Considering a 24 h storm event as shown in Figure 5, a typical file-based linking approach is to input a 24 h off-line rainfall event at once and individually and sequentially run the subsequent models. In a real-time decision support system, however, a possible case is to first use the 1st to 12th hour rainfalls as inputs, and then compute the corresponding flow conditions (Figure 6(a)). As soon as the subsequent data are made available, the second part of the 13th to 24th hour rainfalls is then used as inputs (Figure 6(b)). This process implies that linked models must have the abilities to: (1) separate total time horizon to multiple intervals, (2) exclusively run N-hour simulation per calculation task, (3) suspend all tasks and wait for the data that are available for the next interval, and (4) fetch the next N-hour data and resume calculations.
The simulation process can be suspended at the 12th hour and perform only 1 h of forecasting by giving the 13th forecasted rainfall (Figure 6(c)).

Fortunately, OpenMI is a robust framework that features a well-designed, time-step-by-time-step model linking model. The vision of IRES is to implement OpenMI-compliant SWMM-RUNOFF and SWMM-EXTRAN models and then link them using the tools and interfaces supported by OpenMI to eventually fulfill the abovementioned requirements.

System design

Figure 7 shows the IRES procedure based on the ‘pull-driven’ principle. As can be seen, the SWMM-EXTRAN is first triggered to compute the several intervals. By the time the junction inflows are required, the simulation is suspended and a request of the upper junction inflows for the next inputs is sent to the SWMM-RUNOFF via the calling ‘GetValues( )’ method. SWMM-RUNOFF then calculates multiple time steps until it catches up with the SWMM-EXTRAN requests. Note that the time interval of calculation in SWMM-RUNOFF might be different from that in SWMM-EXTRAN. Until the upper boundary conditions are updated, SWMM-EXTRAN accordingly resumes calculations. The procedure of IRES mainly conforms to the steps described in the Methodology section and Figure 7, by incorporating the OpenMI model-linking ‘GetValues( )’ interface described as in Figure 4.

The design framework of IRES, as shown in Figure 8, mainly includes OpenMI foundations and tools, Linkable and Inceptive Components, SWMM core, and Buffer and Calculation Display. OpenMI is the central component of
IRES, which is responsible for mediating all operations of this system, particularly the runtime data exchange. OpenMI initiates the linkable component, which further initiates inceptive components that are in charge of triggering and managing SWMM cores. In IRES, all results of calculations performed using the SWMM models are saved into the buffer region. All ‘GetValues()’ requested by OpenMI-compliant components access data not directly from the results but indirectly from the buffer. By interacting with OpenMI, the calculation display part finally provides the results that are specified by users.

**EXTRANOpenMIClone and RUNOFFOpenMIClone** are the OpenMI linkable components designed in IRES that wrap and manage the SWMM-EXTRAN and SWMM-RUNOFF models, respectively, thus enabling these models to act like OpenMI-compliant components. Taking SWMM-EXTRAN as an example, Figure 9 shows the class hierarchy of SWMM-EXTRAN OpenMI-compliant components in IRES. Classes placed above the dashed line belong to the OpenMI framework. Owing to the OpenMI standard, the abstract interface ILINKABLECOMPONENT defined in package org.OpenMI.Backbone implements most of the responsibilities of the ILINKABLECOMPONENT, such as complicated data exchange with the other components, buffer management, modeling linking...
managements, thread controls, and so on. The simulation engine, however, remains the most important part of this interface. To ensure flexibility and extensibility, this engine is separated from LinkableComponent derived from ILinkableComponent to create another individual abstract interface IRunEngine, which does not really belong to the scope of the OpenMI standards. The operations related to simulation itself in LinkableComponent are carried out by accessing IRunEngine, i.e. setting input values, performing next step simulation, and reading output results. Parts below the dashed line in Figure 9 are the extended works in IRES. EXTRANEngineWrapper is the simulation engine wrapper class that implements the IRunEngine interfaces. This class wraps the SWMM-EXTRAN kernel – the one truly implemented by procedure-based legacy FORTRAN code – as an individual simulation engine and allows the SWMM-EXTRAN kernel to mimic object-oriented-like behaviors, such as the intent of the façade pattern. On the other hand, EXTRANOpenMIComponent represents the SWMM-EXTRAN OpenMI linkable component inherited from LinkableComponent. Accordingly, IRES follows the same procedures to use SWAMM-RUNOFF as the other OpenMI-compliant linkable component.

Figure 10 demonstrates the IRES class diagram. In addition to EXTRANOpenMIComponent and RUNOFFOpenMIComponent, the other classes belong to OpenMI. ILink is the abstract interface defined in OpenMI to represent the linking to a pair of models. SmartLink is a default implemented version. ILink is further inherited by SmartInputLink and SmartOutputLink, which stand for the upper link to source components and the lower link to target component, respectively.

Details of the ‘GetValues ()’ sequence, including the complete UML sequence diagram, are explained in the Appendix.
System implementation

Several significant implementation issues of IRES are worthy of consideration and discussion. As shown in Figure 8, these issues can be itemized as follows:

1. SWMM revision. Similar to most legacy models, SWMM is designed as an ‘all-in-one’ program (Figure 11), which integrates all procedures, including opening files, reading input, time-step loop, and writing out and closing files, in one program. However, this feature hardly fulfills the IRES requirements of data exchange in a time-step-by-time-step manner. First, all SWMM engines should be revised and, along with the execution sequence, divided into three parts (Figure 11), namely, Initialize(), PerformTimeStep(), and Finalize(). A particular function, PerformTimeStep(), must be isolated to make an independent function invocation perform only a numerical iteration rather than do all iterations at once. Therefore, exchanging data with other models at each time step is possible upon the completion of the PerformTimeStep() sequence. For example, SWMM-EXTRAN is capable of receiving up-to-date junction inflows over runtime instead of waiting for SWMM-RUNOFF to end its execution and receive all junction inflow information from the entire urban area at once.
2. Inceptive component. However, identifying the procedure of modifying legacy software components, such as SWMM models, to be OpenMI-compliant components is a tough challenge. As the minimized re-engineering solution proposed in the tasks involved in this study, ‘inceptive component’ is used to realize the implementation of PerformTimeStep() for the SWMM models; this is an independent binary program that not only carries out very low-level management, such as recording hot-start information in binary files, but also tackles SWMM cores and performs numerical simulations. For example, SWMM-RUNOFF keeps the hot-start information, such as restarting time (hour), units, data length for input and rainfall values, in a binary file. Managing this binary file enables SWMM-RUNOFF to restart at any hot-start status. SWMM-EXTRAN can also keep hot-start information, such as restarting time and the number of simulation cycles. The inceptive component is, therefore, equipped to execute SWMM-EXTRAN according to the hot-start information. Two methods are further implemented for SWMM-EXTRAN, namely, startRun() and evaluateRun(). The first method [startRun()] enables first-time-step simulation to initiate a new SWMM-EXTRAN simulation instance, whereas evaluateRun() follows the steps to advance only one step simulation with respect to any hot-file status specified by the binary files. Apart from a number of different settings for arguments, the two methods are very similar.

3. OpenMI-related implementation. OpenMI-related implements are mainly related to the implementation of OpenMI linkable components and their simulation engine wrapper classes, including EXTRANOpenMIComponent and EXTRANEngineWrapper, as shown in Figure 9 for SWMM-EXTRAN and the counterpart for SWMM-RUNOFF. As an individual simulation engine, EXTRANEngineWrapper is delegated by EXTRANOpenMIComponent and instantiated in its constructor. The code can be listed as:

```java
public EXTRANOpenMIComponent()
{
    _engineApiAccess = new EXTRANEngineWrapper();
}
```

where EXTRANEngineWrapper is the real engine executing SWMM-EXTRAN core managing the current status of simulations. Therefore, the most significant implementation is the PerformTimeStep() function, which advances the simulation by a time step and makes the records for the current time steps. Based on the code listed in Figure 12, IRES first performs the one-step simulation of SWMM-EXTRAN, and then advances the counters to update the records of current steps.

4. Buffer. The buffer region is the object that stores and manages calculated results. Our approach is to set up a key-value map in which the type of map key is ‘simulation time’, while the type of map value is a general object to

```java
public bool PerformTimeStep()
{
    ...
    string inf = "S-2-24.dat";
    string outf = "S-2-524-" + Convert.ToString(this._step);
    if (this._step == 0)
    {
        runextranengine.
        startRun(inf, outf, this._timeStep, ..);
    }
    else
    {
        runextranengine.
        evaluateRun(inf, outf, this._timeStep, ..);
    }
    this._step++;
    ...
}
```

Figure 11 | Revision of the engine core.
Figure 12 | Pseudocode of PerformTimeStep().
keep the results stored by a predefined data structure. For example, results, such as junction stages, total inflow, total outflow, and junction surcharges calculated at a particular time step, are stored in a single object with respect to a ‘plain old data’ (POD) structure.

5. Calculation display. Proposing a model-linking approach rather than the software application implies that user interface is never the most significant concern. However, a good interface for calculation display is helpful in checking correctness and numerical agreement. Users can also select the junctions that they want to check, and the results are directly fetched out from buffer regions.

APPLICATIONS

To demonstrate the applicability and to verify the correctness of IRES and the model-linking approach proposed in this paper, the Jhong-Kang drainage system in eastern Taipei, Taiwan, a real sewer system region in an urban area, is selected as the case study. The rainfall input is a design storm with five-year intensity and 24 h duration (Figure 5). To verify correctness, test works are conducted by following the steps below.

1. Benchmark case. This case is performed using typical SWMM models with simple file-based linking. All

![Figure 13](https://iwaponline.com/jh/article-pdf/14/4/884/386822/884.pdf)
2. **IRES simulation with multiple rainfall inputs.** The benchmark case can be regarded as the ‘exact solution’. A similar procedure is used with IRES, but the rainfall inputs are divided into multiple input intervals, and SWMM-RUNOFF and SWMM-EXTRAN are executed to alternatively simulate. This process emulates real-time rainfall input.

3. **Comparison.** Several junctions in the Jhong-Kang drainage system are randomly selected to compare the corresponding junction stages simulated by IRES with the benchmark case. Finally, comparison results are exhibited by charts and tables.

   Four different simulation time intervals (STI), namely, 12, 6, 4, and 2 h, are used for the IRES multiple rainfall inputs. When STI is 12 h, the 24 h duration of the design storm is divided into two equal 12 h sections. It is worth noting that the selection for STIs is not based on any actual real-time system in practical engineering cases. It is only for system demonstrations to verify applicability. The first section included rainfall from the 1st to the 12th hours (Figure 6(a)), whereas the second included that from the 13th to the 24th hours (Figure 6(b)). Similarly, when STI is 6 h, a four-section case is equally divided based on the 24 h design storm. The first section included rainfall from the 1st to the 6th hours (Figure 13(a)); the second from the 7th to the 12th hours (Figure 13(b)); the third from the 13th to the 18th hours (Figure 13(c)); and the fourth from the 19th to the 24th hours (Figure 13(d)). STI = 4 and 2 h continues in the same way.

   Figure 14 shows a simple map of the Jhong-Kang drainage stations. Junctions 4335–818 and 4536–04 are randomly selected for comparison. Results are listed in Tables 1 and 2, respectively, and junction stages are plotted as Figures 15 and 16, respectively. A few results, such as the 2nd, 14th, and 16th hour of STI = 2 h for junction 4335–04 (Table 1), are slightly different with the benchmark case. The majority of the results are exactly equal to the benchmark case, implying that the numerical agreement is excellent, and no ambiguous bias has been produced by IRES. Since this is a simple one-way linking case without two-way alternative iterations, the only possible difference should come from rounding errors.
### Table 1 | The water depth of junction no. 4335–818 (simulation time interval, STI)

<table>
<thead>
<tr>
<th>Time (h)</th>
<th>SWMM</th>
<th>STI – 12 h</th>
<th>STI – 6 h</th>
<th>STI – 4 h</th>
<th>STI – 2 h</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.75</td>
<td>1.75</td>
<td>1.75</td>
<td>1.74</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
</tr>
<tr>
<td>14</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1.74</td>
<td>1.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
<td>1.74</td>
</tr>
</tbody>
</table>

### Table 2 | The water depth of junction no. 4336–04 (simulation time interval, STI)

<table>
<thead>
<tr>
<th>Time (h)</th>
<th>SWMM</th>
<th>STI – 12 h</th>
<th>STI – 6 h</th>
<th>STI – 4 h</th>
<th>STI – 2 h</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.53</td>
<td>0.53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.56</td>
<td>0.56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td>14</td>
<td>0.59</td>
<td>0.59</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.53</td>
<td>0.53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
</tr>
</tbody>
</table>

### Figure 15 | The results of junction stages at no. 4335–818. (a) STI – 12 h. (c) STI – 4 h. (b) STI – 6 h. (d) STI – 2 h.
CONCLUSIONS

This study proposes an approach using the OpenMI framework to integrate the urban rainfall-runoff and the storm sewer models through the SWMM-RUNOFF and SWMM-EXTRAN, respectively, with real-time rainfall inputs and dynamic data exchange for real-time simulation of storm sewer flows. An integrated system, IRES, has been developed to demonstrate the feasibility and the applicability of the proposed approach. The Jhong-Kang pumping system in Taipei is used as the case study to verify the correctness of the IRES. A design storm with 5-year intensity and 24 h duration is pre-designed for the compassion. In comparison with a benchmark case performed using typical SWMM, IRES has shown excellent numerical agreement. Two conclusions can be drawn from this study.

1. The OpenMI framework dynamic data exchange can solve urban hydrological problems. At each time step of the simulations, IRES runtime allows urban hydrological data exchange between the rainfall-runoff and sewer models. The exchanged data have been proven to be spatially and temporally correct.

2. IRES is a software system developed using the OpenMI, suggesting strong abilities for the integration of hydrological models. Incorporating with any legacy software...
models only requires minimized re-engineering works, irrespective of which programming language is used for the development of legacy components. OpenMI can provide those legacy models with good reusability via well encapsulation and greatly supported tools and frameworks.

ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support by the National Science Council under Grants NSC 97-2625-M-033-001 and NSC 98-2625-M-033-001.

REFERENCES


First received 10 April 2011; accepted in revised form 8 December 2011. Available online 9 March 2012

APPENDIX

Figure 17 is the sequence diagram illustrating the details of the ‘GetValues()’ implementation for data exchange between SWMM-EXTRAN and SWMM-RUNOF. The delicate and sophisticated procedures are carried out through the interactions between two OpenMI linkable components (EXTRANOpenMICOMPONENT and RUNOFFOpenMICOMPONENT) inherited from the LinkableComponent that has already implemented most of default procedures. Once again, the real engines are separated. These two linkable components only tackle the tough tasks of model linking and data exchanges.
Figure 17 | IRES sequence diagram for GetValues().