An Effective Flood Forecasting System Based on Web Services*

Ya-Hui Chang, Pei-Shan Wu, Yu-Te Liu, and Shang-Pin Ma

Department of Computer Science and Engineering, National Taiwan Ocean University
yahui@ntou.edu.tw

Abstract. A flood forecasting system usually needs to integrate many hydraulic modules, which may be legacy programs written in FORTRAN, running in heterogeneous environments, and differing in execution time. Besides, the data required for each module should be provided in real time, and the programs also need to be executed in a correct sequence. In this paper, we discuss how to build the flood forecasting system based on Web services to support module integration and data exchange. We also propose several workflow strategies for composing services, and perform experiments to determine which strategy is better.

1 Introduction

The disaster brought by heavy rain has become more and more serious in Taiwan for the past few years due to global warming. In certain areas such as the Linbien city, which is surrounded by the South China Ocean and passed by two rivers, *i.e.*, the Linbien River and the Lili River, the situation is even worse due to the complex interaction of rain, rivers, waves and tides. It is well known that the Morakot typhoon in 2009 seriously damaged that area. Therefore, it is very important to build an effective flood forecasting system for early warning so that the authorities can evacuate residents in time.

A few experts in the hydraulic discipline have developed several program modules with different functionality for that specific area. Particularly, the Hydrol module is constructed to predict the river overbank, the WaveTide module is used to estimate the future overtopping discharge of the ocean, and the TwoFD module is designed to estimate the water level of the whole area. A complete flood forecasting system needs to integrate these modules to provide comprehensive warning messages, but it is not a trivial task based on the following reasons:

– These legacy hydraulic modules are written in the FORTRAN programming language. Moreover, some modules only function well in the LINUX environment, while others are designed to run in the Windows operating system. The flood forecasting system should be able to invoke modules operated in different environments.

* This work was partially supported by the National Science Council under Contract No. NSC 101-2625-M-019-004-.

J.-S. Pan et al. (Eds.): *Advances in Intelligent Systems & Applications*, SIST 21, pp. 681–690.
DOI: 10.1007/978-3-642-35473-1_67

- There exists complex data dependency relationship between these modules. For example, the output of the WaveTide module is needed by both the Hydrol module and the TwoFD module for real-time computation. Therefore, the integrated system should ensure that these modules run in a correct order, and each module gets the data it requires.
- The Hydrol Module takes a few minutes to compute, while the WaveTide and TwoFD Modules may take up to an hour. Since the running time of modules differs a lot, we wish to design a good workflow strategy so that the throughput of the whole system can increase.

There have been many flood forecasting systems seen in literature. They are built based on different architectures to meet their own needs. For example, the European Flood Alert System (FEAS) [10], which intends to provide flood forecasting in trans-national river basins, is a large system and applies the grid architecture to integrate several national hydrological and meteorological services. The Web-based Flood Forecasting System (WFFS) [8] and the Flood Early Warning System (FEWS) [7], which are small to medium sized systems, are component-based and apply the Java technique. On the other hand, the concepts of *services*, particularly *Web services*, attract a lot of attention recently due to their flexibility and reusability. For example, the researchers in [5,6] use services to allow others accessing their water resources represented in data warehouses or databases. The researchers in [4] identify the functions which are frequently required by a flood forecasting system, such as formatting data, and represent them as Web Services, while the researchers in [3] intend to represent the water resource models as Web services.

In this research, we also propose to build the flood forecasting system using Web services, since Web services can be invoked through networks without concerning about the differences of the underlying environment. Besides, there exist softwares to assist in composing Web services into different execution sequence. These properties reduce the cost of maintaining the system. The contributions of this paper are summarized as follows:

- We wrap several legacy hydraulic modules written in FORTRAN as Web services, so that these modules can be easily composed and reused as the basis of a complete flood forecasting system.
- We discuss different workflow strategies to compose these services and design several experiments to examine their performance.
- We implement the whole system based on the .Net solution [2], particularly the Windows Communication Foundation (WCF) for Web services and the Windows Workflow Foundation (WF) for flow control. It is shown that this system can function effectively.

The rest of this paper is organized as follows. In Section 2, we introduce the functionality and the output of the legacy hydraulic modules. In Section 3, we describe the whole architecture for the flood forecasting system. In Section 4, we describe the Web services designed for hydraulic modules, and propose several different workflow strategies. Finally, experimental results and the system prototype are shown in Section 5, and conclusions are given in Section 6.

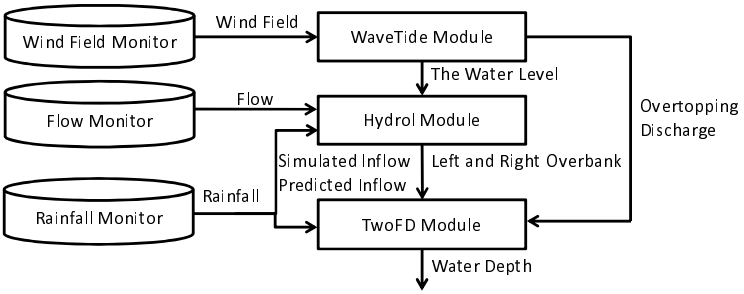


Fig. 1. Data Exchange between Hydraulics Modules

2 Preliminaries

We first introduce the functionality of the existing hydraulic modules. Recall that the flood forecasting system discussed in this paper is designed especially for the Linbien city, which is surrounded by the ocean and rivers.

During a typhoon period, the strong wind usually causes storm surge and swell, and there might also exist tide. *The WaveTide module* first applies the data got from the *wind wield monitor* to calculate the overtopping discharge of the coastal structure, and simulates the water level of the downstream. Second, *the Hydrol module* then evaluates how the heavy rain affects the river flow. It will apply the data got from the *rainfall monitor* and output simulated inflow of the upstream, where there are no flow monitors, and also output predicted inflow for the next three hours. Moreover, it will base on the calculated or observed flow information and the water level of the downstream outputted from the WaveTide module, to estimate the left and right overbank of each river section. Finally, *the TwoFD module* divides the area into around 40,000 small grids. It uses the technique of parallel computing and bases on the output of the other two modules to calculate the water level of each grid, and to predict which area should be warned.

Figure 1 summarizes how these modules exchange data, and their outputs are represented in plain text files as listed in Table 1. For example, the file "HydrolInput.dat" outputted by the WaveTide module provides the water level at a particular location and a particular time. The piece of sample data indicates

Table 1. Output Data of Hydraulic Modules

Module	File name	Sample Data
WaveTide	HydrolInput.dat	(12:00:00, 198738.643, 2480062.259, 0.0098)
	TwoFDInput.dat	(12:00:00, 200588.092, 2478948.852, 6.8, 1)
TwoFD	Output.txt	(12:00:00, 201944.681, 2485393.236, 38.5)
Hydrol	K_RealXX.dat	(12:00:00, 8.63, 2.1)
	K_PredXX.dat	
	SecData_LinbienXX.dat	(12:00:00, 0.098, 0.012, 729.971, 109.793)
	SecData_LiliXX.dat	

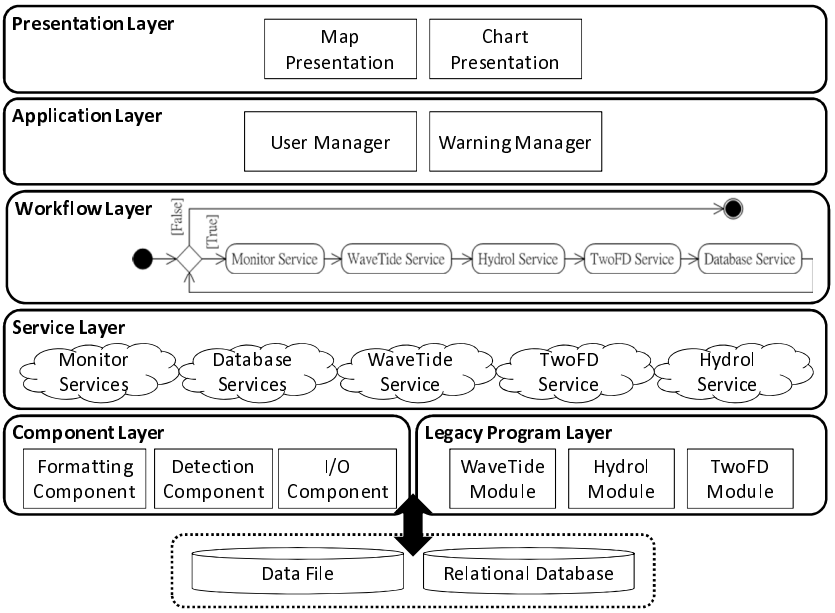


Fig. 2. The Architecture of the Flood Forecasting System

that the water level is 0.0098 m at the coordinates (198738.643, 2480062.259) at the time 12:00:00. The contents of other files are similar and we omit the detailed discussion due to space limitation. One thing to note is that the symbol "XX" represents an identifer for a certain river or a river section.

3 Architecture

We now describe the architecture for the flood forecasting system, which is designed to correctly coordinate those legacy hydraulic modules running in heterogeneous environments, and to output the warning message in real-time.

As shown in the bottom of Figure 2, this system operates two types of data. The first one is the data files outputted by existing hydraulic modules, as described in Section 2. The second one is a relational database additionally maintained to assist in efficient data retrieval, display, and backup. Next, the *Legacy System Layer* consists of the existing hydraulic modules as discussed before, and the *Component Layer* consists of several utilities programs. Particularly, the *formatting components* are used to transform the data acquired from the monitor to the proper format required by the existing hydraulic modules. The *detection components* are used to detect if the monitored data have been transmitted to the local directory. The *I/O components* are responsible for the interaction with the relational database software, including storing and retrieving data.

Table 2. Interfaces of Hydraulic Services

Service	Input Parameter	Output Parameter
WaveTide	NowWind	HydrolInput, TwoFDInput
Hydrol	NowFlow, NowRainFall HydrolInput	K_Real, K_Pred, SecData_LiLi, SecData_LinPien
TwoFD	NowRainFall, TwoFDInput, K_Real, K_Pred, SecData_LiLi, SecData_LinPien	TwoFDResult

On top of the Legacy System Layer and the Component Layer, we implement several Web services. The *WaveTide service*, the *TwoFD service*, and the *Hydrol service*, correspond to the three existing hydraulic modules. By wrapping as Web services, these modules can be invoked from different operating systems, and provide information upon request. The *Monitor services* actually correspond to several services, each of which is designed to get the data from a particular monitor station and provides data in the proper format. The *Database services* refer to those services which involve the interaction with the relational database software, which can be further classified into ToRDB services and FromRDB services. The former type of services is responsible for automatically exporting data calculated by hydraulic modules to relational databases, and the latter type of services is used by the upper *Application Layer*. Note that there is a *Workflow Layer* above the Service Layer. It is used to compose those services in a correct order, and those composite services can be created based on users' demand.

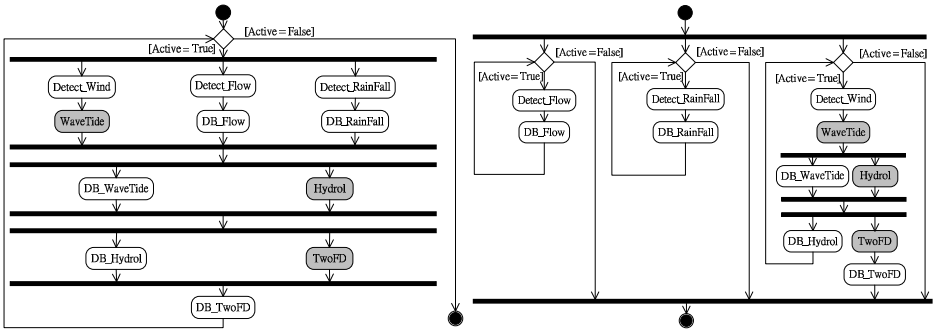
Finally, the *Application Layer* and the *Presentation Layer* complete the construction of the flood forecasting system as a Web-based system. Particularly, *the User Manager* classifies users as administrators or ordinary users and allow them to browse different Web pages. The *Warning Manager* displays warning messages on the Web pages by maps or charts.

4 Hydraulic Services

In this section, we first discuss how to wrap the three existing hydraulic modules as Web services, which are called collectively as *hydraulic services*. We then propose several possible ways of composing those services.

4.1 Interfaces of Hydraulic Services

As described in Section 2, each existing hydraulic module is written in FORTRAN, which operates on and outputs plain text files. For easy correspondence, we let the output parameters of each hydraulic Web service directly correspond to the original output files as listed in Table 1, and design the input parameters based on their data dependency relationship as depicted in Figure 1. The interfaces of the three hydraulic services are summarized in Table 2. In the table, the three parameters *NowWind*, *NowRainFall*, and *NowFlow* represent the data got from the wind field monitor, the rainfall monitor, and the flow monitor, respectively, while the parameters *HydrolInput*, *TwoFDInput*, and *TwoFDResult*



(a) Strategy I (b) Strategy II

Fig. 3. Workflow Strategies I and II

represent the contents of the files "HydrolInput.dat", "TwoFDInput.dat", and "Output.txt", respectively. An exception is the output parameter of the Hydrol service, since the associated module outputs more than 70 files. If we let each file correspond to a parameter, there will be too many parameters to process. Therefore, we let the same type of data represented by the same parameter, and there are only four output parameters for that service as shown in Table 2.

Based on the discussion above, it is obvious that we need the transformation between plain text files and parameters. Therefore, we wrap the following three parts of program fragments in sequence into a hydraulic Web service:

- transforming the input parameters into input text files
- invoking the legacy hydraulic module, which operates on the input text files, and outputs the calculated results as text files
- transforming the output text files into output parameters

4.2 Composing Web Services

As depicted in Figure 1, the hydraulic services need to be invoked in a correct order for continuous calculation. Moreover, the output of each hydraulic service also conveys useful information, which should be represented in the relational database for further retrieval. Since there exist obvious time difference when executing the three hydraulic services individually, we propose several possible strategies of composing these services and discuss the underlying rationale. The process considered starts from gathering the input data till representing the output data in the relational databases, and covers the following services: (1) the three hydraulic services, *i.e.*, *WaveTide*, *Hydrol*, and *TwoFD*, (2) the monitor services, *i.e.*, *Detect_Wind*, *Detect_Flow*, and *Detect_RainFlow*, and (3) the ToRDB services, *i.e.*, *DB_RainFlow*, *DB_Flow*, *DB_WaveTide*, *DB_Hydrol*, and *DB_TwoFD*.

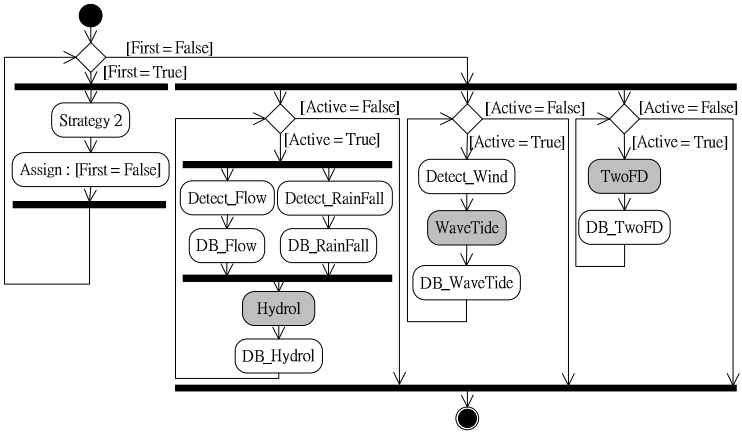


Fig. 4. WorkFlow Strategy III

Each workflow strategy is depicted as a UML activity diagram. In particular, each rounded square represents a Web Service. An exception is the rounded square denoted by the word "Assign", which represents the value assignment to a particular variable. We also let each Web Service associated with one or more variables corresponding to its output parameters, which are omitted in the figure for simplicity. In addition, a diamond represents a conditional statement, a pair of parallel thick dark lines represent parallel processing, and an arrow represents sequential processing.

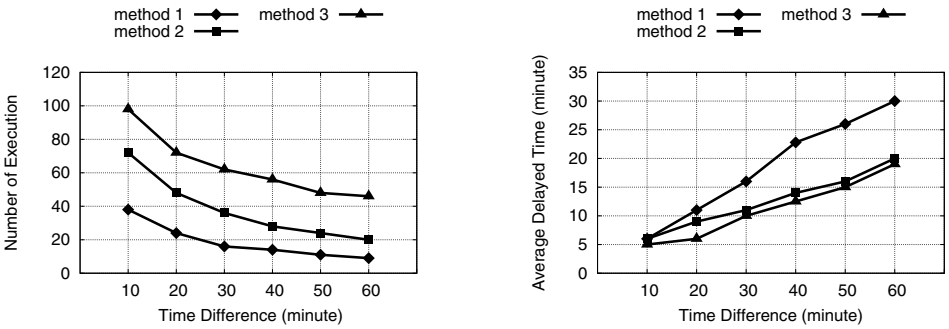
First, Figure 3(a) shows the baseline strategy. That is, a hydraulic service is invoked as soon as all its dependent services have produced new values. In addition, those services which are not related, such as *DB_Flow* and *WaveTide*, is scheduled to run in parallel. A special boolean variable *Active* is used to indicate if the process is currently active or has been terminated by the administrator.

The second strategy, as depicted in Figure 3(b), considers the large time differences between services, and wishes to represent as many as possible data in the relational database. Recall that the Hydrol service only takes minutes to compute, but the WaveTide service and the TwoFD service may take up to an hour. Besides, the flow monitor and the rainflow monitor transmit monitored data every ten minutes. Therefore, we let the monitor services continuously execute without waiting. In addition, after the Hydrol service has executed and transmitted data to the relational database, we immediately start a new execution instance while the TwoFD service is still running. By doing this, we can shorten the time of waiting the slow service to finish.

The final strategy further extends this idea, and makes each service run independently purely based on the current available values of the input parameters. The corresponding activity diagram is depicted in Figure 4. It is divided into two parts. The left part represents the first execution, where there exist no values for the input parameters. It will first execute based on the second strategy, and then assign the variable *First* the value *False*, so the remaining executions will apply the right part of the workflow, where each hydraulic module runs independently.

Table 3. Experiment Setup

WaveTide (min)	Hydrol (min)	TwoFD (min)	Max Difference (min)
1	6	11	10
1	11	21	20
1	16	31	30
1	21	41	40
1	26	51	50
1	31	61	60



(a) (b)

Fig. 5. Evaluations of Different Workflow Strategies

5 Evaluation

In this section, we perform several experiments to evaluate the different workflow strategies, and demonstrate the constructed flood forecasting system.

5.1 Evaluation of Workflow Strategies

In this section, we evaluate the proposed workflow strategies by controlling the execution time of the three hydraulic services. As listed in Table 3, we fix the execution time of the WaveTide service, and increase the execution time of the other services. The maximum time difference of these hydraulic services are denoted in the last column.

In the first experiment, we compare the total number of execution for all hydraulic services within five hours. Since each execution outputs useful information, we prefer the workflow strategy with the highest number. As illustrated in Figure 5(a), we can see that the numbers decrease along with the values of time differences, but strategy III is always the best one.

In the next experiment, we consider the TwoFD service, which requires the outputs of the other two hydraulic services. We calculate the *average delayed time* for this service as follows, where the subscript i represents the i_{th} execution, *start* represents the time when the TwoFD service starts execution, while *input* represents the time associated with its input parameters:

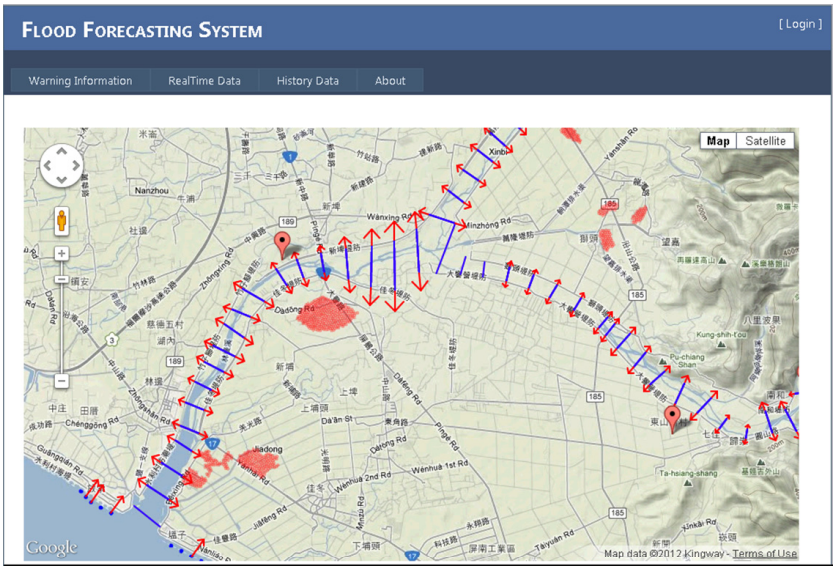


Fig. 6. The Homepage of the Flood Forecasting System

$$\frac{\sum_{i=1}^{n}(start_i - input_i)}{n}$$

If this value is small, the TwoFD service can perform its calculation based on more recent data, and its outputs will be more convincible. As illustrated in Figure 5(b), the average delayed time of the three workflow strategies all increase along with the maximum time difference, but strategy III is still the best.

5.2 System Prototype

The flood forecasting system is built as a Web-based system. Those hydraulic services discussed in Section 4 are implemented in the back end and their computed results are stored in the relational database. FromRDB services are then applied to retrieve required information from the database for display. The whole system is implemented by using the programming languages ASP.net and Javascript. The Web server is IIS and the relational database server is MS SQL Server 2008. As shown in Figure 6, the homepage of the system is mainly the map of the Linbien city annotated with warning messages. Particularly, each river section is noted with a short line, with arrows to indicate river overbank. The arrows depicted by the ocean, as seen in the lower left corner of the map, represent wave overtop. The set of small gray circles show the areas which are inundated. This shows the effectiveness of our approach, and we omit the detailed discussion due to space limitation.

6 Conclusion

In this paper, we demonstrate how to wrap several legacy hydraulic programs as Web services to build a flood forecasting system. We have successfully implemented this system, and currently test it under the real environment. In the future, we will also investigate the open standard such as OpenMI [1,9], so that the proposed services can be more reusable for other systems.

References

1. Open modeling interface, `http://www.openmi.org`
2. Windows .net solution, `http://msdn.microsoft.com`
3. Goodall, J.L., Robinson, B.F., Castronova, A.M.: Modeling water resource systems using a service-oriented computing paradigm. Environmental Modelling & Software 26(5), 573–582 (2011)
4. Diaz, L., Granell, C., Gould, M.: Service-oriented applications for environmental models: Reusable geospatial services. Environmental Modelling & Software 25(2), 182–198 (2010)
5. Dzemydiene, D., Maskeliunas, S., Jacobsen, K.: Sustainable management of water resources based on web services and distributed data warehouses. Technological and Economic Development of Economy 14(1), 38–50 (2008)
6. Goodall, J.L., Horsburgh, J.S., Whiteaker, T.L., Maidment, D.R., Zaslavskye, I.: A first approach to web services for the national water information system. Environmental Modelling & Software 23(4), 404–411 (2008)
7. Hydraulics, D.: Delft-fews, an open shell flood forecasting system, `http://www.wldelft.nl/soft/fews/int/index.html`
8. Li, X.-Y., Chau, K.W., Cheng, C., Li, Y.S.: A web-based flood forecasting system for shuangpai region. Advances in Engineering Software 37(3), 146–158 (2006)
9. Liao, Y.-P., Lin, S.-S., Chou, H.-S.: Integration of urban runoff and storm sewer models by using openmi framework. Journal of Hydroinformatics 14(3) (2012)
10. Thielen, J., Bartholmes, J., Ramos, M.-H., de Roo, A.: The european flood alert system-part 1: Concept and development. Hydrology and Earth System Sciences 13(2) (2009)