A Three-Tier System Architecture Design and Development for Hurricane Occurrence Simulation

Shu-Ching Chen*, Sneh Gulati†, Shahid Hamid‡, Xin Huang*, Lin Luo*, Nirva Morisseau-Leroy§, Mark D. Powell¶, Chengjun Zhan* and Chengcui Zhang*
*School of Computer Science, Florida International University, Miami, FL 33199, USA
Email: {chens, xhuan001, lluo0001, czhan002, czhang02}@cs.fiu.edu
†Department of Statistics, Florida International University, Miami, FL 33199, USA
Email: gulati@fiu.edu
‡Department of Finance, Florida International University, Miami, FL 33199, USA
Email: hamids@fiu.edu
§Cooperative Institute for Marine and Atmospheric Science, University of Miami, Coral Gables, FL 33124, USA
Email: nirva.morisseau@noaa.gov
¶Hurricane Research Division, NOAA, Miami, FL 33149, USA
Email: Mark.Powell@noaa.gov

***Abstract*— As an environmental phenomenon, hurricanes cause significant property damage and loss of life in coastal areas almost every year. Research concerning hurricanes and their aftermath is gaining more and more attentions nowadays. The potential changeability of hurricane data and hurricane models requires robust, maintainable and easily extensible software system for hurricane simulation. With focuses on the design and the implementation of the components at each layer, this paper describes a hurricane simulation system built on the three-tier architecture to achieve good flexibility, extensibility and resistance to potential changes.**

***Index Terms*— Three-tier architecture, Web-based system, Database, Hurricane**

I. Introduction

An important step in hurricane analysis and prediction is building computer models of a hurricane. Usually, statistical models are built from the historical hurricane data and then the analysis and the prediction can be performed based on these models. Unfortunately , the number of documented hurricanes is limited. For example, in HURDAT database [11], which is maintained by the National Hurricane Center in Miami, Florida and the National Climatic Data Center in Asheville, North Carolina, there are only 1274 hurricanes. One way to supplement for the number of hurricanes is to run simulations based on the statistical models built from historical data. The projection data from the simulation can be integrated with the real hurricane data for further use such as hurricane track modeling and loss and damage estimation. Therefore, the computer system for the purpose of hurricane modeling and projection is of great significance in aiding the hurricane analysis and hurricane hazard prediction.

This paper presents our work in designing and building a web-based distributed software system that can be used for the modeling and projection of hurricane occurrences. Our system can let the user build statistical models of the hurricane occurrence from the historical data stored in the database, which is a part of the system. It also provides simulation and projection functionality so that the user can run simulations and projections based on the statistical models.

Compared with other relevant systems, which are discussed in section II, the proposed system has the following features: 1) It is a large-scale system which can handle the huge amount of hurricane simulation data and the intensive computation required for analysis and projection; 2) It aims to support both professional and general-purpose users in a very convenient way; 3) Our system is built upon an object-relational database management system, Oracle9i, which is one of the core system components to store and manage the historical hurricane data, the hurricane data model and the projection results. 4) Since the hurricane data are constantly being updated and the mathematical models for the hurricane data are also potentially changeable, a three-tier architecture is adopted as our system's fundamental architecture to provide the transparency among the data layer, business logic layer and the user interface layer, thus making our system more flexible, maintainable, robust and resistant to the potential changes in the lifetime of the system.

The rest of this paper is organized as follows. First, the related works are discussed briefly. Next, the system is introduced from the architectural point of view in Section III. Then the design and implementation of the major system components, namely the user interface layer, application logic layer and database components, are described in a comprehensive way in Section IV, Section V and Section VI respectively. Finally, Section VII gives the conclusion.

II. RELATED WORK

There has been a lot of research on modeling natural atmospheric hazards. Ever since 1948 when Charney et al. made the first successful dynamical-numerical forecast, numerous researchers have contributed to weather modeling and prediction and established Numerical Weather Prediction (NWP) models and systems, such as ARPS [1] and RAMS [14]. The Advanced Regional Prediction System (ARPS), developed at the University of Oklahoma, is a complete atmospheric numerical modeling/prediction system designed to explicitly represent convective and cold-season storms. RAMS [14], or the Regional Atmospheric Modeling System, is a highly versatile numerical code produced to simulate and forecast meteorological phenomena. RAMS was developed by researchers at Colorado State University and the ASTER division of Mission Research Corporation. Like most of the meteorological phenomena modeling and projection system, ARPS and RAMS have the following components: 1) A data analysis package to preprocess the data from observation for future computation; 2) An numerical atmospheric model which performs the actual simulation and prediction; 3) A post-processing package that handles the visualization and analysis of results.

Although improvement in the study of NWP systems favors more and more accurate weather forecasting, most of those systems have the following limitations:

1) There are very few database management and warehouse techniques used in those systems. Due to the tremendous amount of data and time-consuming modeling process which are two inherent problems in natural hazard modeling and prediction, it could be very helpful to apply database management or data warehouse techniques to store, retrieve and manage the data and models efficiently. Currently, most of the so-called GIS "Databases", such as Global Ecosystems (GEP) Database [3] and State Soil Geographic (STATSGO) Database [15], are merely collections of data sets instead of being stored in and managed by a real Database Management System (DBMS).
2) Most of these systems are stand-alone systems. Each application is running on one or several machines, and they are totally independent from each other. Thus it is difficult for different users to share and exchange information.

On the other hand, some software products for the hurricane damage and loss assessment already exist. One of the most prominent tools is HAZUS [2] [10]. HAZUS stands for Hazards U.S. and was developed by the Federal Emergency Management Agency (FEMA) as a standardized, national methodology for natural hazards losses assessment. Using Geographic Information Systems (GIS) technology, HAZUS can support estimates of damage and losses that result from various natural disasters such as wind and flood. Some useful databases, such as a national-level basic exposure database, are built into the HAZUS system that allows the user to run a preliminary analysis without having to collect additional local data. It also provides the functionality to allow the user to plug their own data into the databases.

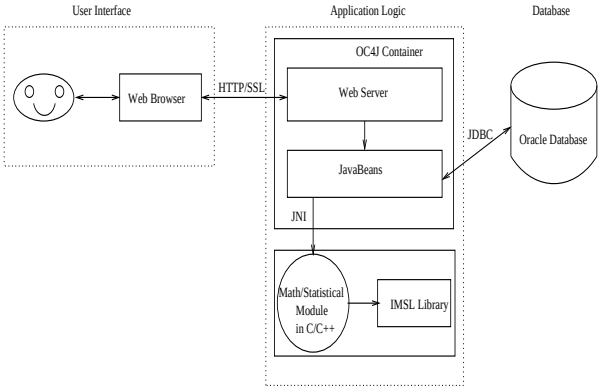

Fig. 1. Detailed architecture of the system

Although HAZUS is powerful and useful, it is not suitable to be used by general-purpose users who may know a little or nothing about the profound mechanisms. And as a stand-alone GIS system, the necessary software such as the commercial GIS package like ArcView, and hardware need to be installed in every machine on which the HAZUS system runs, which in turn increases both the expenses and manual labor.

III. SYSTEM ARCHITECTURE

To achieve the system robustness, flexibility and resistance to potential change, the popular three-tier architecture is deployed in our system. The architecture is composed of three layers: the user interface layer, the application logic layer and the database layer. The three-tier architecture aims to solve a number of recurring design and development problems, hence to make the application development work more easily and efficiently. The interface layer in the three-tier architecture offers the user a friendly and convenient entry to communicate with the system while the application logic layer performs the controlling functionalities and manipulating the underlying logic connection of information flows; finally, the data modeling job is conducted by the database layer, which can store, index, manage and model information needed for this application.

User Interface Tier The first tier is the user interface tier. This tier manages the input/output data and their display. With the intention of offering greater convenience to the user, the system is prototyped on the Internet. The users are allowed to access the system by using any existing web browser software. The user interface tier contains HTML components needed to collect incoming information and to display information received from the application logic tiers. The web visitors communicate with the web server via application protocols, such as HTTP and SSL, sending requests and receiving replies. In our system, the major web-scripting language exploited in

designing the presentation layer is the Java Server Pages (JSP) technique [7] . The detailed design and implementation of this tier will be discussed in detail in Section IV.

Application Logic Tier The application logic tier is the middle tier, which bridges the gap between the user interface and the underlying database, hiding technical details from the users. An Oracle9i Application Server is deployed. Its OC4J container embeds a web server, which responds to events, such as data receiving, translating, dispatching and feed-backing jobs [12] [13]. Components in this tier receive requests coming from the interface tier and interpret the requests into apropos actions controlled by the defined work flow in accordance with certain pre-defined rules. Java Beans perform the appropriate communication and calculation activities, such as getting/pushing information from the database and carrying out the necessary computing work with respect to proper statistical and mathematical models. JDBC [5] is utilized for Java Beans to access the physical database. In the interest of the quick system response, C/C++ language is used to program the computing modules that are integrated into the Java code via JNI [6]. The details on this tier are in Section V.

Database Tier The database tier is responsible for modeling and storing information needed for the system and for optimizing the data access. Data needed by the application logic layer are retrieved from the database, then the computation results produced by the application logic layer are stored back in the database. Since data are one of the most complex aspects of many existing information systems, it is essential in structuring the system. Both the facts and rules captured during data modeling and processing are important to ensure the data integrity. An Oracle9i database is deployed in our system, and the Object Relational Model is applied to facilitate data reuse and standard adherence. Section VI will give more details about it.

IV. User Interface

The intended system is prototyped into Internet; therefore, the design and implementation of the system user interface mainly becomes a job of designing and implementing web pages. The users can gain access to the system through any commonly used commercial browser such as Internet Explorer, Netscape, etc.

Due to its "unlimited" expressive power and natural coherence with the J2EE architecture, JSP web-scripting technology is adopted to implement the web pages [7] [8]. JSP, sitting on top of a Java servlets model, can easily and flexibly generate the dynamic content of a web page. The basic idea of JSP is to allow Java code to be mixed with static HTML or XML templates. The Java logic handles the dynamic content generation while the markup language controls the structuring and the presentation of data.

Figure 2 shows the basic course by which the user interacts with the system. One individual JSP page is implemented for each individual user-system interaction. First, a log-in JSP page is used to take care of the user log-in process. The user needs to provide the user name and password for the system to authenticate the user.

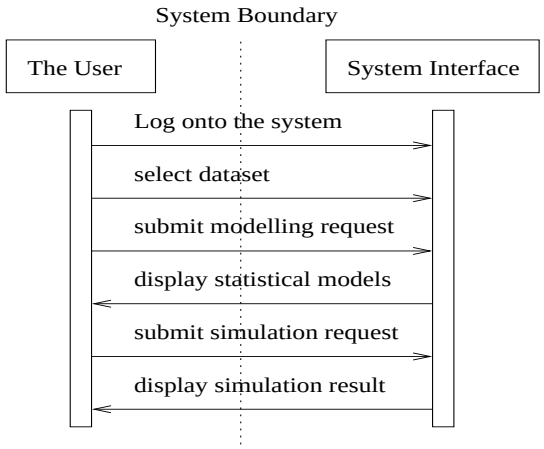

Fig. 2. Interaction flow between the user and the system

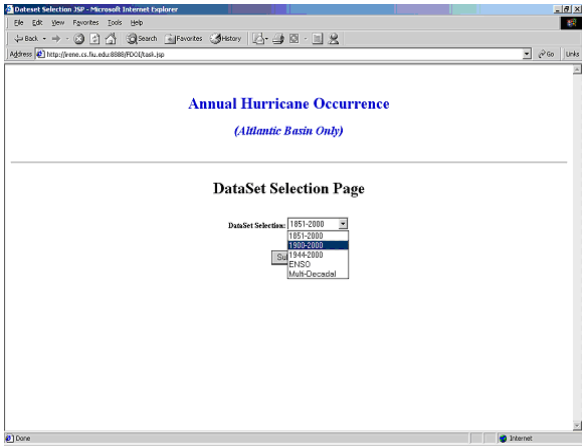

Fig. 3. Data set selection web page

One meteorological fact is that the statistical properties of hurricanes vary with different year ranges. For example, the statistical properties of storms in El-Nino years are quite different from those in non-El-Nino years. Therefore, different statistical models are necessary for different year ranges. In our system, all the historical hurricane records in the database are categorized into five datasets according to meteorologic criteria, which are: 1) 1851-2000, 2) 1900-2000, 3) 1944-2000, 4) ENSO and 5) Multi-Decadal. Different statistical models are built for individual datasets. Therefore, after the user logs onto the system, another JSP page shown as Figure 3, allows the user to select the data set. After a user selects the data set, another JSP page lets the user submit a request to the application logic layer to build the statistical model of the hurricane for the selected data set. Then the application layer builds the model, stores the model into database and sends the model to the user interface layer which displays the

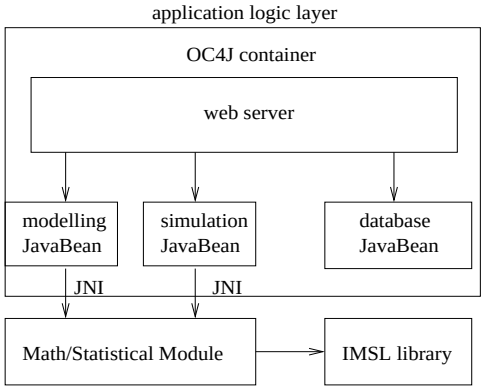

Fig. 4. Basic structure of application logic layer

resultant statistical model to the user by using another JSP page. After that, the user can request the application layer to run simulation (projection) based on the statistical model. The application layer does a simulation as the response, stores the model into the database and feedbacks the simulation results to the user interface layer, which displays them via a JSP page.

V. APPLICATION LOGIC LAYER

The application logic tier is the middle tier which bridges the gap between the user interface and the underlying database, hiding technical details from the user. It communicates with the user interface, performs the statistical modeling and simulation, and interacts with the database layer such as retrieving hurricane data from the database and storing the statistical model and simulation results into the database.

A. Application Logic Layer Overview

Figure 4 shows the basic components in the application logic layer and the relationships among those components.

An Oracle9i Application Server is deployed to supply the fundamental services that allow components to concentrate on business logic without concern for low-level implementation details. It handles networking, authentication, authorization, persistence, and remote object access. Its OC4J container embeds a web server, which responds to events such as data receiving, translating, dispatching and feed-backing job [12] [13]. The Java Beans perform all the actual work in business logic. The "database JavaBean" utilizes JDBC [5] to access the physical oracle database to retrieve and store the hurricane data, statistical models and simulation results. The "modeling JavaBean" is responsible for building the statistical models for the hurricane data from the user's specified data set. The "simulation JavaBean" runs simulations using the statistical models. For the sake of performance, time-consuming computation tasks, namely the statistical model calculation and simulation, are actually achieved by using C/C++ codes in "Math/Statistical Module", which runs on linux platform. The C/C++ code is seamlessly integrated into corresponding Java codes in the Java Beans (the "modeling JavaBean" and the "simulation JavaBean") via the JNI (Java Native Interface) mechanism [6]. The commercial software IMSL [4] provides the high-performance C routines for mathematical and statistical calculation.

B. Annual Hurricane Occurrence modeling

Annual Hurricane Occurrence modeling aims to model the number of hurricanes occurring per year (*AHO*) and the hurricane genesis time (*AHO*). Therefore, for each data set there are actually two statistical models: one for the *SGT* and the other for the *SGT*. After the modeling procedure, the statistical models are stored into the database via the "database JavaBean". The modeling algorithms are implemented in "Math/Statistical Module" which is called by the "modeling JavaBean" and the "simulation JavaBean"

1) AHO Modeling: Since different statistical models are built for different dataset, the user first needs to select one dataset from the five categories through the user interface as aforementioned. Let M data samples in the user-specified dataset retrieved from the database be denoted by $X = \{x_i\}(i = 1, 2, \ldots, M)$, where M is the number of years in the dataset and x_i denotes the number of hurricanes occurring in the i^{th} year in the dataset. The statistical model of *AHO* is built based on the M data samples. According to domain knowledge in meteorology, the best statistical distribution of the number of hurricanes occurring per year is either the Poisson distribution or the Negative Binomial distribution. First, the parameters of both the Poisson distribution and Negative Binomial distribution are estimated from the data samples X. Then the chi-square statistic is calculated to select the final model. The distribution with a higher p-value, namely, the distribution with the better fit is selected as the final statistical model of the *AHO*.

2) SGT modeling: The genesis time of a storm is the first fix data of that storm. *SGT* modeling aims to model the genesis time of the hurricanes. This is achieved by modeling the number of hours between the genesis of a storm in six-hour resolution and the start of its hurricane season rather than directly modeling the storm genesis time. A storm season starts from May 1st of one year and ends at April 30th of the next year. After modeling the number of storms using *AHO* from the historical data, the *SGT* model can be used to predict the time intervals among storms, and thus the storm genesis time of each storm can be predicted as well.

Let N data samples in the user-specified dataset retrieved from the database be denoted by $S = \{s_i\}(i = 1, 2, \ldots, N)$ where N is the number of hurricanes in the dataset and the time associated with the i^{th} hurricane in the selected data set be donoted by s_i. The statistical model of *SGT* is built using the data samples S. Specifically, a nonparametric approach is applied to estimate the Cumulative Distribution Function (CDF) of the time intervals . Let the random variable time denoted by T. First the empirical CDF $F(t)$ for T is calculated from the data samples S. Then the smooth estimator of $F(t)$ is

calculated based on empirical CDF using Epanechnikov kernel which serves as the final statistical model of *SGT*.

C. Simulation/Projection

Based on the statistical model built from the historical hurricane data, the system can run a simulation/projection in response to the user's request based on the desired number of years for the simulation he/she specifies. After the simulation procedure, the simulation results are stored into the database via the "database JavaBean".

Let K denote the number of years the user specifies. First, K random numbers $r_i (i = 1, 2, \ldots, K)$ are generated from the *AHO* model, namely either the estimated Poisson distribution or Negative Binomial distribution. Each random number, r_i, means the number of hurricanes occurred in an individual year. The total number of hurricanes simulated is $U = \sum_{i=1}^{K} r_i$. Then, U random numbers $h_j (j = 1, 2, \ldots, U)$ are generated from the distribution in the *SGT* model. Each random number, h_j, denotes the interval associated with the j^{th} simulated hurricane. Therefore, for the j^{th} simulated hurricane, its genesis time is projected, which is the first day of the hurricane season plus the interval h_j.

VI. Database Component

Data analysis and modeling is a vital aspect of the database component. In our system, an object-relational design pattern is applied to model hurricane data. Object-relational model can assist the reuse of the database objects. The Oracle9i database is incorporated in the system as the information storehouse, which stores data records for all storms occuring in the Atlantic Basin since 1851. An object-relational database schema is designed to facilitate the data reusability and manageability. The major advantage brought by the object-relational concepts is the ability to incorporate higher levels of abstraction into our data models, while current relational databases are usually highly normalized models but with little abstraction. The overall view of the hurricane data schema is depicted in Figure 5 .

VII. Conclusion

In this paper, a web-based distributed system for the projection of hurricane occurrences is presented. It integrates a group of individual applications by combining hurricane data acquisition, storage, retrieval, and analysis functions. The system exhibits a modular, extensible, and scalable architecture that makes it possible to adapt to more complex tasks, such as storm track simulation and wind field generation. The well-established three-tier architecture is exploited to build the system. A variety of advanced techniques such as JSP, JNI and JDBC are used in the design and development of the application. Both the Oracle Database and the Application Server are deployed to integrate the system coherently. The completed implementation is easy and convenient to use. In addition, it is accessible to any user who is able to connect to the Internet and has interest in hurricane prediction information.

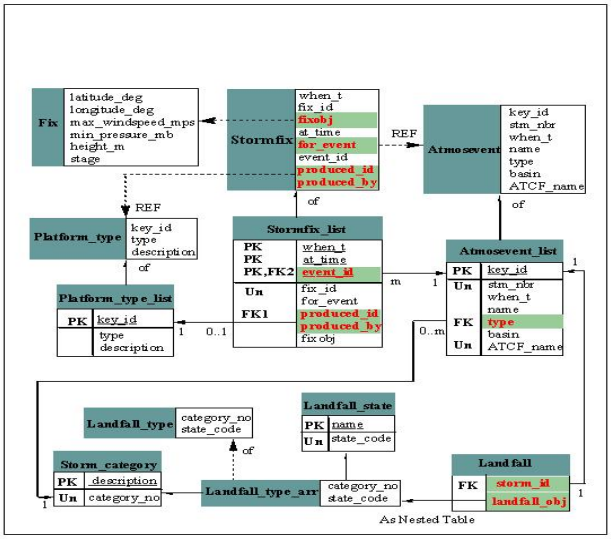

Fig. 5. Database schema

Acknowledgment

This work is partially supported by Florida Department of Insurance under "FIU/IHRC Public Hurricane Risk and Loss Model Project." While the project is funded by the Florida Department of Insurance (DOI), the DOI is not responsible for this paper content.

References

[1] M. Xue, K. Droegemeir, and D. Wang, "The Advanced Regional Prediction System (ARPS) - A Multiscale Nonhydrostatic Atmospheric Simulation and Prediction Tool. Part I: Model Dynamics and Verification," *Meteor. Atmos. Physics*, vol. 75, pp. 161–193, 2000.

[2] HAZUS Home, *http://www.fema.gov/hazus/*

[3] Global Ecosystems Database(GED), *http://www.ngdc.noaa.gov/seg/fliers/se-2006.shtml*

[4] IMSL, *http://www.vni.com/products/imsl/*

[5] The JDBC API Universal Data Access for the Enterprise, *http://java.sun.com/products/jdbc/overview.html*

[6] Java Native Interface, *http://java.sun.com/docs/books/tutorial/native1.1/*

[7] JavaServer Pages(TM) Technology, *http://java.sun.com/products/jsp/*

[8] N. Morisseau-leroy, M.K. Solomon, and J. Basu, *Oracle8i: Java Component Programming with EJB, CORBA, and JSP*, Oracle Press (McGraw-Hill/Osborne), 2000.

[9] National Hurricane Center, *http://www.nhc.noaa.gov/*

[10] HAZUS Overview, *http://www.nibs.org/hazusweb/verview/overview.php*

[11] HURDAT data, *http://www.aoml.noaa.gov/hrd/hurdat/Data_Storm.html*

[12] Oracle9iAS Container for J2EE, *http://technet.oracle.com/tech/java/oc4j/content.html*

[13] D. Panda, "Oracle Container for J2EE (OC4J)," *http://www.onjava.com/pub/a/onjava/2002/01/16/oracle.html*

[14] R.L. Walko, C.J. Tremback, "RAMS: reginal atomospheric modeling system, version 4.3/4.4 - Introdcution to RAMS 4.3/4.4," *http://www.atmet.com/html/docs/rams/ug44-rams-intro.pdf*

[15] NSSC (1994) State Soil Geographic (STATSGO) Database, Miscellaneous Publication Number 1492, National Soil Survey Center, United States Department of Agriculture.