VAWKUM Transactions on Computer Sciences
http://vfast.org/journals/index.php/VTCS@ 2021, ISSN(e): 2308-8168, ISSN(p): 2411-6335
Volume 9, Number 1, January-December , 2021 pp:8-22

New Approach to Distributed Flood Prediction Model using Agents based Communication

Naveed Qasim[1]

Department of Computer Sciences , Abdul Wali Khan University Mardan, Pakistan
*Corresponding author email address: naveed91pk@gmail.com

ABSTRACT

Flood is the overflow of water from its normal ways to submerges land which is usually dry. It is clear that floods cannot be stopped, but the timely prediction and management is the only way to combat this hazard. Much work is documented on physical and logical solution to predict on the basis of risk to save people and reduce the destruction of their belongings. In this work we developed a logical model based on client and server agent communication to predict the hazard. In the process, initially a conceptual model is developed, further, an algorithm is constructed. Moreover, for it correctness and verification the algorithm is transformed to formal language Z. Finally the specified formal model is implemented using JAVA programming language.

KEYWORDS

Keywords: Z-Notation, Flood hazard, Logical solutions, Client-Server Strategy, Specification

JOURNAL INFO

HISTORY: Received: August 12, 2021 Accepted: September 30, 2021 Published: November 28, 2021

1. INTRODUCTION

Flooding is the overflow of water from river banks and waterways. It is a natural disaster, occurs in many countries of the world and brings a lot of destruction [1][2][3]. It can occur due to heavy snow or rainfall, high melting ratio, dam, ponds failure or levees failure, high tides, winds, thunderstorms, tsunami, etc. [7]. Unfortunately, the flood cannot be avoided, however, can be predicted in time to save lives and properties of the people [1][7].

In literature, there exists lots of work related to the prediction of upcoming flood with some deficiencies. That includes SPH, which is a simulation based model, uses MAYA, GIS (Geographical Information System), DEM and remote sensing image to model an inundation area. Hydrograph is a technique to show the discharge of water on a daily basis using graph [10] This technique uses in the US and some other countries. Another technique ANFAS is using software with an interactive interface to users. This technique is used in China and France [9]. A combined project for flood forecasting at Mahanandi River basin based on topography information is named as DMS (Disaster Management support) was initiated by GOI (Government of India) and USAID (United States Agency for International Development) [11]. Japan uses a model for the prediction of flood which is collecting data from 700 different critical points; it then alerts the people through cell phones and mobile internet about the flood. This model is known as FRICS (Foundation of River & basin Integrated Communications) [8]. Another model Unsteady Flow River (UFR) used software to calculate the discharge of water by taking some parameters including flow of water and the capacity of the river [12] The above techniques have some limitations; like Hydrograph has no decision power, FRICS has a communication problem in catastrophic conditions, SPH uses only historical data, DMS has insufficient resolution, ANFAS requires heavy bandwidth, and UFR does not predict a flood but only calculates the discharge of water [6][11][12]. To overcome the above problems, we use agent based approach to predict the flood. For the purpose, the model is divided into two parts i.e. server and client agents. The client is located in a suitable place on the river where flood water is passing. It measures speed of water using sensors and calculates the cross-sectional area at that point and sends it to the server. Further, the server calculates the discharge and finds the approximate time of the upcoming flood and makes the appropriate decision based upon the value of discharge and time based on historical database of flood situations.

The paper is organized as follows. Section 2 dealt with the development of the conceptual model. Section 3 presents the design of algorithm based on the developed conceptual model. In section 4, the designed algorithm is specified by using the formal language Z with Z/EVES specification and verification tool to make its verification by removing inconsistencies in the flow of the work. In Section 5, the implementation, using Blue J the Java Programming language is realized, while in section 6, the analysis and comparison of the model is demonstrated. Finally section 7 concludes the paper.

2. CONCEPTUAL MODEL FOR SINGLE CLIENT

To make the system practical, we choose a simulation based model, where the client is installed at point "A" on the river to measure the velocity of flood and the area of the cross-section through which the flood is passes as shown in the Figure 2. Further, we suppose a situation where the velocity of water at point "A" is recorded as $\beta\, km/h$ and the area of the cross-section is calculated as $\lambda\, m^2$. The client sends this data to the server. The server first converts $\beta\, km/h$ to $\phi\, m/s$ as $\left[\beta\, km/h = \beta\, km/h \times (1000/3600) = \phi\, m/s\right]$. Further, the server calculates the total discharge as $\phi\, m/s \times \lambda\, m^2 = \Phi\, m^3/s$. The server now predicts the flood accruing in $\frac{\alpha km}{\beta km/h} = {}^{\alpha}\!/_{\beta}\, Hours$ where E is the location of inundation area and $\alpha\, km$ is the distance between the point "A" and the point "E". The value of discharge and time of prediction depends on the area of cross-section and speed of flood as shown in the Table 3.

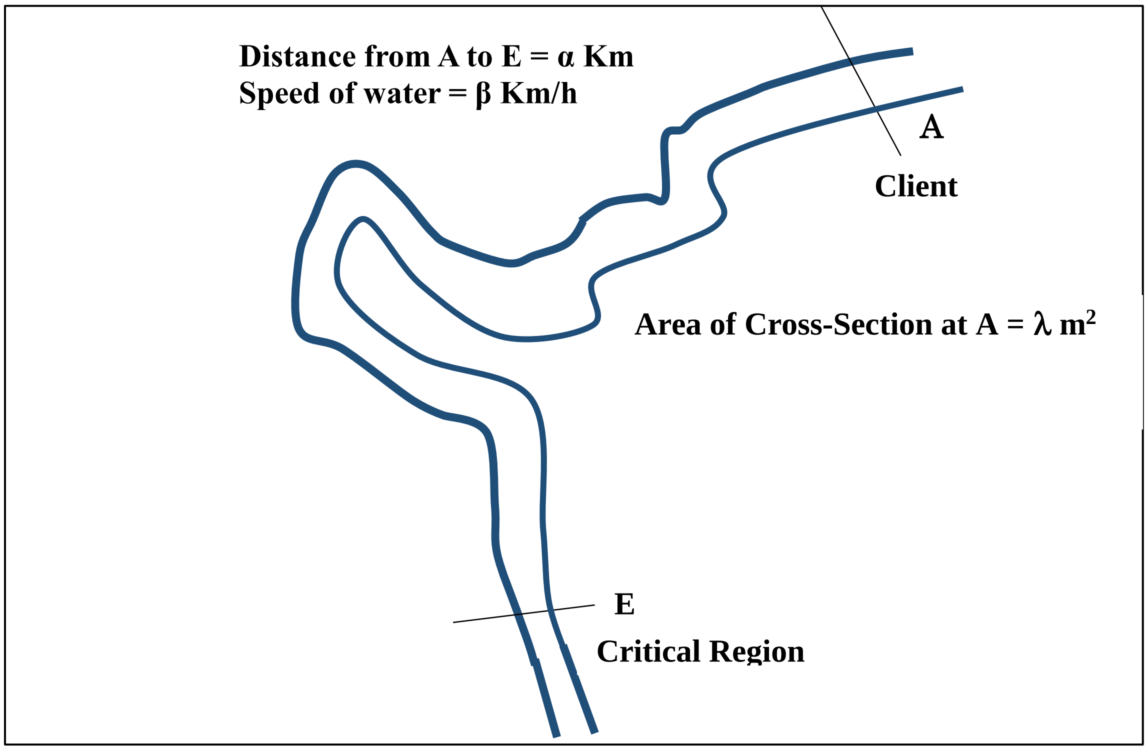

Distance from A to E = α Km
Speed of water = β Km/h
A
Client
Area of Cross-Section at A = λ m²
E
Critical Region

Figure 2. Conceptual Model.

Table 3. Values of Discharge w.r.t to Time.

S.No	Distance b/w A & E (km)	Speed (km/h)	Speed (m/s)	Area (m2)	Discharge (m3/s)	Time (Hour)
1	200	30	8.33	790	6581	6.67
2	200	32	8.89	793	7050	6.25
3	200	34	9.44	796	7514	5.88
4	200	36	10	799	7990	5.56
5	200	38	10.56	802	8469	5.26
6	200	40	11.11	805	8944	5
7	200	42	11.67	808	9429	4.76
8	200	44	12.22	811	9910	4.55
9	200	46	12.78	814	10403	4.35

In this work important thing is to calculate continuously the accurate discharge as long as the flood as passing through the client region. To calculate the discharge, speed and area of the cross-section are the two required values. To calculate the speed, the current meter is used and Simpson rule is used for measuring the area of cross-section of irregular surfaces. For area of the cross-section, the width of irregular bottom is divided into n subintervals with n+1 digital bubble tubes as shown in the Figure 3. The depth of the river is recorded at each bubble tube. The spacing of the intervals is made in such a way that there should be not more than 10 percent (ideally 5 percent) of the total discharge between two subintervals. The cross-section area is calculated as:

$$\textbf{Cross} - \textbf{section Area} \approx \frac{h}{3}\left[LW(0) + 2\sum_{i=1}^{\frac{n}{2}-1} LW(2i) + 4\sum_{i=1}^{\frac{n}{2}} LW(2i-1) + LW(n)\right]$$

Where, ‘h’ is the width of the river between two consecutive bubble tube $LW(n-1)$ to $LW(n)$. The vertical lines in the figure show n+1 bubble tubes, while the dots show the points where the velocity is measured, i.e. velocity is measured at different points of the area of cross-section and then the average velocity is to be taken. The total discharge is calculates as

$$Discharge = Cross - Section\ Area \times Velocity$$

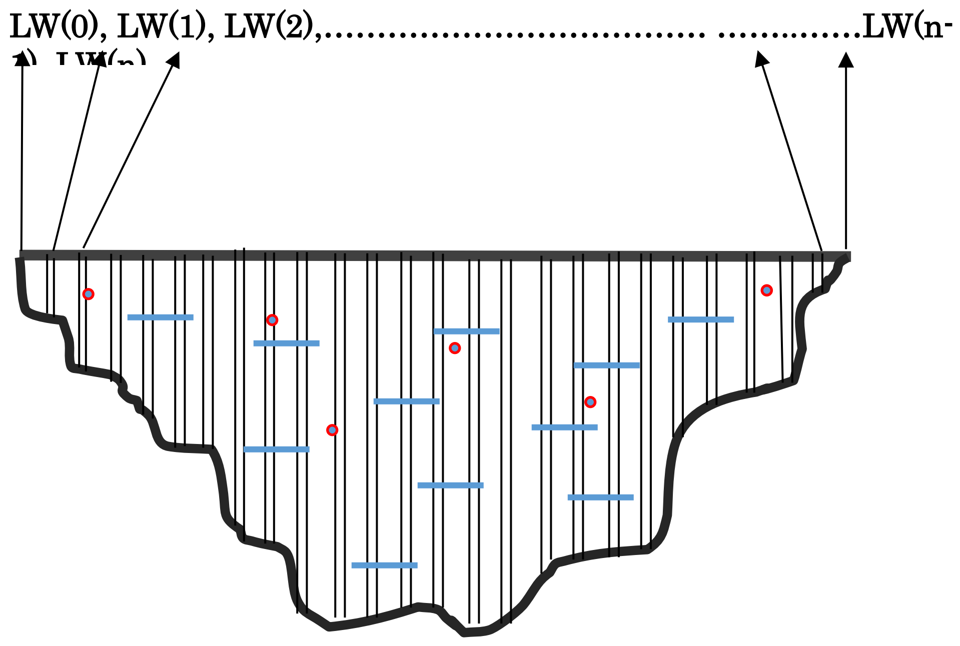

LW(0), LW(1), LW(2),.................................. ....LW(n-1), LW(n)

Figure 3: Area of Cross-Section by Simpson’s 1/3 Rule.

3. ALGORITHMIC APPROACH TOWARD THE CONCEPTUAL MODEL

The algorithm is based on client, server communication, calculation and prediction. Initially, the server establishes connection to the client through “connectAgent ()”. If it connected successfully, the server calls the function “calculateSA ()” of the client, to get the speed and area of the cross-section at that agent. Then the server calculates the discharge value, and connects to the database and retrieves the historical flood data for that area. Further, it compares the calculated discharge value to the historical flood data to determine the flood type so that to predict the approximate time of flood at the critical region.

Server Module

```
Flood-Prediction ( ) {
    boolean agent ← **connectAgent( )**
    while (!agent){
      agent ← connectAgent( )
      wait for some time …….
    }
    struct sa   ← calculateSA( )      // sa contains speed, area and clientNo.
    speed ← sa.speed
    area ← sa.area
    discharge ← speed × area
    boolean db ← connectDB( )
    if (db) {
    retrieve information from database for the specific area i.e $D_{ci1}$ to $D_{ci4}$  values for a specific area.
        if ( $D_{ci4}$<discharge ) {
            alarm ← on
            flood-type ← exceptionally-high
            message ← exceptionally high flood will reach in (distance/speed) time.      // server will know
              the location of the client & so will know the distance.
        } else if ( $D_{ci3}$< discharge   ≤ $D_{ci4}$ ) {
            alarm ← on
            flood-type ← very-high
            message ← very high flood will reach in (distance/speed) time.
        } else if ( $D_{ci2}$< discharge   ≤ $D_{ci3}$ ) {
            alarm ← on
            flood-type ← high
            message ← high flood will reach in (distance/speed) time.
        } else if ( $D_{ci1}$< discharge   ≤ $D_{ci2}$ ) {
            alarm ← off
            flood-type ← medium
            message ← medium flood will reach in (distance/speed) time.
        } else if (discharge   ≤ $D_{ci1}$ ) {
            alarm ← off
            flood-type ← low
            message ← low flood will reach in (distance/speed) time.
        }
    }
}          // Flood-Prediction
```

Client Module

```
calculateSA( ) {
    speed ← read the speed of water through current meter.
    k ← 0 to m // (m+1) is the total number of bubble tubes in the river.
        depth (k) = read the depth of water at k.
      // Calculations of Area through Simpson's 1/3rd rule
    sum ← depth(0) + depth(m)
    k ← 2 to m-1 (step 2)          // that is increment the value of k by 2 in each interval.
        sum ← sum + 2 × depth(k)
    k ← 1 to m-1 (step 2)
        sum ← sum + 4 × depth(k)
    $area \leftarrow sum \times \frac{h}{3}$    // h is the width of each interval
    struct speedArea
    speedArea.speed ← speed
    speedArea.area ← area
    return speedArea
```

}

In the above algorithm, the client agent has the only one component ***calculateSA*(), initially, t**he function reads the speed of water and stores it in “speed”. It then calculates the area of cross-section of the river and stores it in a variable “area”. Further, it stores the “speed” and “area” in a structure “speedArea” and returns it to the calling function. The server agent contains the function **connectAgent( )** uses to connect the server with client to get the stored speed and area “sa”. The function **connectDB( )** provides connection to the database of the historical data. Where Dci_1, Dci_2, Dci_3 and Dci_4 are the values taken from the historical data. Finally the algorithm compares the calculated discharge against the values (Dci_1, Dci_2, Dci_3 and Dci_4) and based on the comparison, it determines the type of flood for the specific area with its prediction.

4. SPECIFICATION OF PROPOSED MODEL IN Z-NOTATION

To check the flow of the algorithm and its verification we transform the algorithm into Z-Notation, which is given as below.

$$Alarm ::= on \mid off$$

$$Flood\_Type ::= low \mid medium \mid high \mid very\_high \mid exceptionally\_high$$

Server

$Dci_1, Dci_2, Dci_3, Dci_4: \mathbb{N}_1$

$discharge: \mathbb{Z}$

$alarm!: Alarm$

$flood\_type!: Flood\_Type$

$discharge \geqslant 0$

Server_init

$Server$

$discharge = 0 \wedge alarm! = off \wedge flood\_type! = low$

Client

$speed: \mathbb{Z}$

$area: \mathbb{N}$

$hci: \mathbb{N}$

$speed \geqslant 0$

Client_init

$Client$

$speed = 0 \wedge area = 0$

CalculateSA

$\Delta Client$
$LW: \mathbb{N} \rightarrow \mathbb{Z}$
$sum: \mathbb{N}$
$max: \mathbb{N}$
$read?: \mathbb{N}$

$sum = 0 \wedge max \geqslant 0$
$sum = LW\ 0 + LW\ max$
$\forall k: 1 \mathinner{..} max - 1 \bullet\ k \bmod 2 = 0 \Rightarrow sum = sum + 2 * LW\ k$
$\forall k: 1 \mathinner{..} max - 1 \bullet\ k \bmod 2 = 1 \Rightarrow sum = sum + 4 * LW\ k$
$area = sum * hci \operatorname{div} 3$
$speed = read?$

Discharge

$\Xi CalculateSA$
$\Delta Server$

$discharge = area * speed$

Time

$\Xi CalculateSA$
$distance: \mathbb{N}$
$approximate\_time!: \mathbb{N}$

$approximate\_time! = distance \operatorname{div} speed$

Message ::= *low_flood_will_reach_in*
| *medium_flood_will_reach_in*
| *high_flood_will_reach_in*
| *very_high_flood_will_reach_in*
| *exceptionally_high_flood_will_reach_in*

Message2 ::= *approximate_time*

In the discussed specification, schema are used which have three parts, name of schema, declaration part and the predicated part. Attributes are declared in the declaration part and their relationships are given in the predicate part. Some attributes are also declared as Attribute-Name ::= attribute-values, outside the schema box. The decoration "?" shows input to the schema while "!" shows output from the schema. Schemas are included in other schemas are denoted as Ξ*Schema-Name,* which means that the attributes of the included schema are used but their values cannot be changed. Schemas are also included in other schema as Δ*Schema-Name,* which means that the attribute values of the included schema can be changed. In the schemas we use the following attributes: **Alarm with only** two possible that is on and off. The attribute **Flood_Type** have five possible values which are low, medium, high, very_high and exceptionally high. **Server** is the base schema of the system which declares four attributes Dci_1, Dci_2, Dci_3 and Dci_4 of the type Natural Numbers N, discharge of type integer, alarm of type Alarm and flood_type of type Flood_Type. **Server_init** is an initial schema of the schema Server, which initializes discharge to zero, alarm to off and flood_type to low. **Client** is another base schema, which represents the static view of the client. In its declaration part, it declares speed of type Integer, area and hci of the type non negative natural numbers where hci represents the width of each interval between bubble tube at client i. In predicate part, it restricts the values of speed to be non-negative. **Client_init** is an initial schema of the base schema **Client**, which initializes the values of speed and area. **CalculateSA** is an operational schema which can change the **Client** schema due to the inclusion of Δ*Clients*. **Discharge** is an operational schema which uses the functionalities of other schema with changing them by using **Ξ*CalculateSA* and Δ*Server***. **Time** is an operational schema in which **Ξ*CalculateSA*schema** is included. **Message** is an attribute which have five possible values which are low_flood_will_reach_in, medium_flood_will_reach_in, high_flood_will_reach_in, very_high_flood_will_reach_in, and exceptionally_high_flood_will_reach_in. The **Message2** is also an attribute which have only one possible value that is an approximate_time. **Alert** is the final operational schema which includes the schemas Δ*Discharge* and Ξ*Time*. It declares two other variables, the message of type Message and message2 of type Message2. In the predicate part, it only compares the value of discharge to historical data Dci_1, Dci_2, Dci_3 and Dci_4. It alerts the flood authorities about the predicted flood based on the comparison of the actual flood with the historical data.

```
Alert
ΔDischarge
ΞTime
message!: Message
message2!: Message2
──────────────
Dci₄< discharge
⇒ flood_type! = exceptionally_high
∧ alarm! = on
∧ message! = exceptionally_high_flood_will_reach_in
∧ message2! = approximate_time
Dci₃< discharge ⩽ Dci₄
⇒ flood_type! = very_high
∧ alarm! = on
∧ message! = very_high_flood_will_reach_in
∧ message2! = approximate_time
Dci₂< discharge ⩽ Dci₃
⇒ flood_type! = high
∧ alarm! = on
∧ message! = high_flood_will_reach_in
∧ message2! = approximate_time
Dci₁< discharge ⩽ Dci₂
⇒ flood_type! = medium
∧ alarm! = off
∧ message! = medium_flood_will_reach_in
∧ message2! = approximate_time
discharge ⩽ Dci₁
⇒ flood_type! = low
∧ alarm! = off
∧ message! = low_flood_will_reach_in
∧ message2! = approximate_time
```

5. IMPLEMENTATION OF THE SYSTEM USING BLUE J

The proposed specified and verified model is for the Single Client is implemented in Java programming language using **BlueJ** IDE as shown in the Figure 4. A brief introduction of each block is given as follows:

The **Server** is the main block of the System represents the Server. Initially it uses the block "**WelCome**" to welcome the user, further, it uses the block "**FloodTypes**" to show the types of flood. Moreover, it sends message to "**Client**" block and get input data from through "**StreamIO**" block. After that it connects to the block "DBClass" to retrieve the historical data. It then calculates the results and sends them to "**StreamOutput**" block. After the execution the welcome screen, flood types, Alert to client by server, Server Client Communication and calculated flood data are shown in the figures 5, 5, 7, 8 and 9.

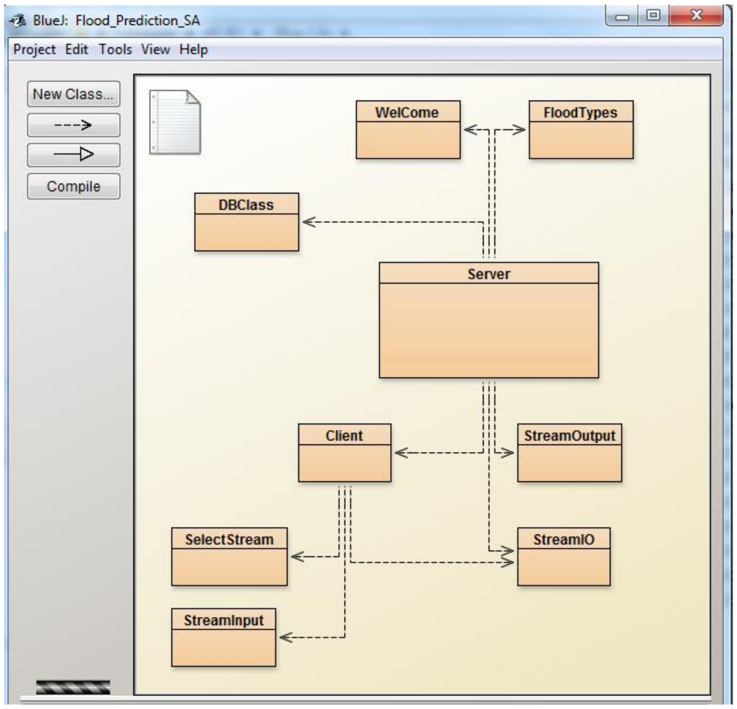
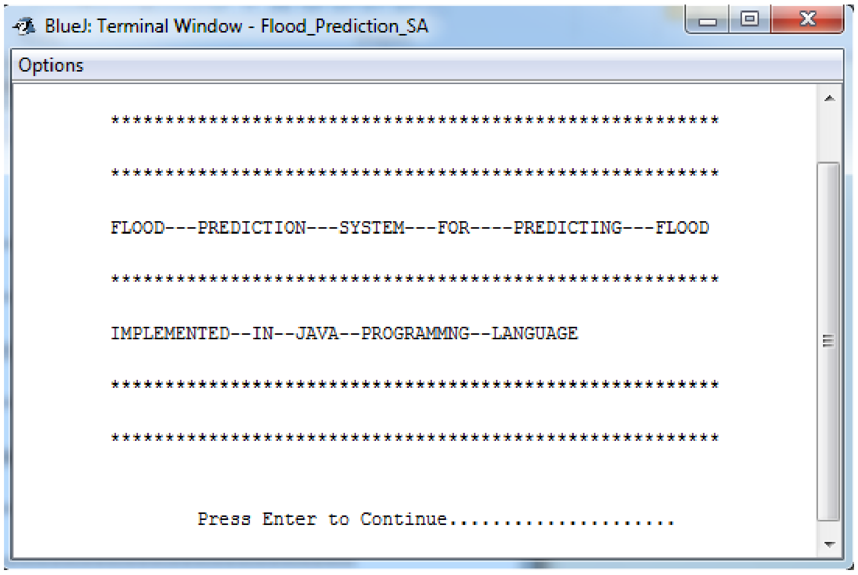
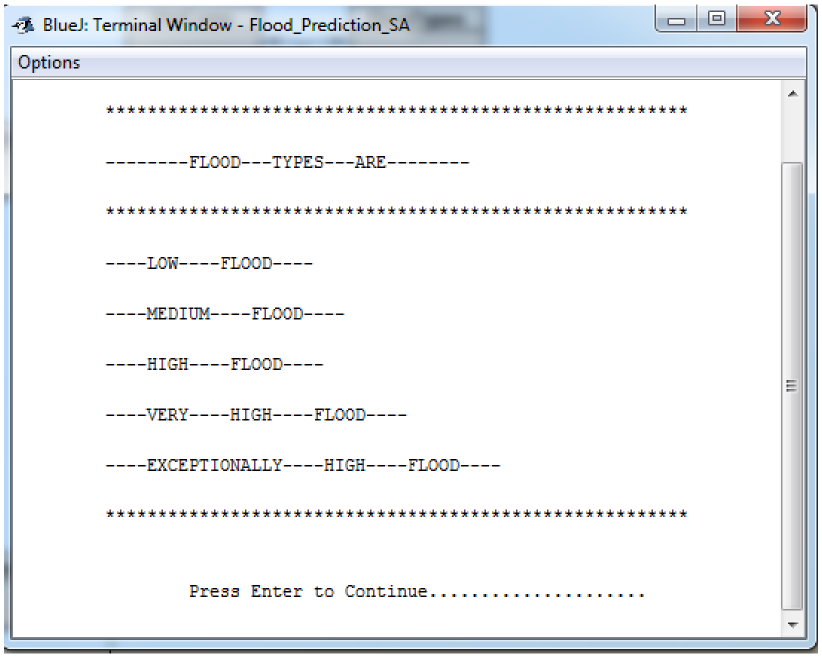


Figure 4: Block Diagram of Single Client.



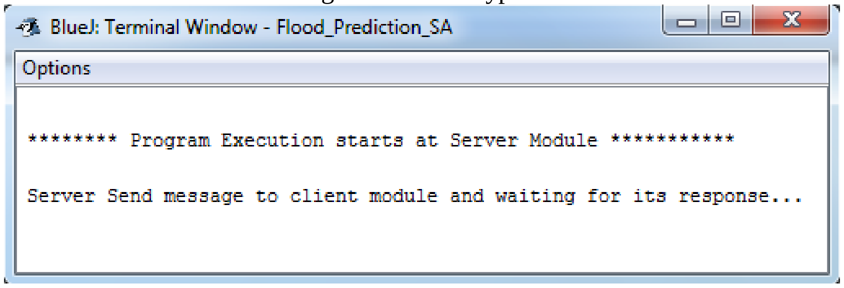
```
*********************************************************

*********************************************************

FLOOD---PREDICTION---SYSTEM---FOR----PREDICTING---FLOOD

*********************************************************

IMPLEMENTED--IN--JAVA--PROGRAMMNG--LANGUAGE

*********************************************************

*********************************************************


        Press Enter to Continue....................
```

Figure 5: Welcome Screen.



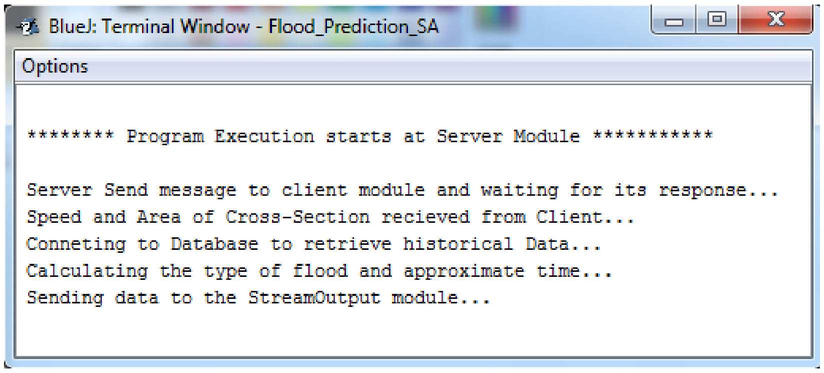
```
********************************************************

--------FLOOD---TYPES---ARE--------

********************************************************

----LOW----FLOOD----

----MEDIUM----FLOOD----

----HIGH----FLOOD----

----VERY----HIGH----FLOOD----

----EXCEPTIONALLY----HIGH----FLOOD----

********************************************************


        Press Enter to Continue....................
```

Figure 6: Flood Types.



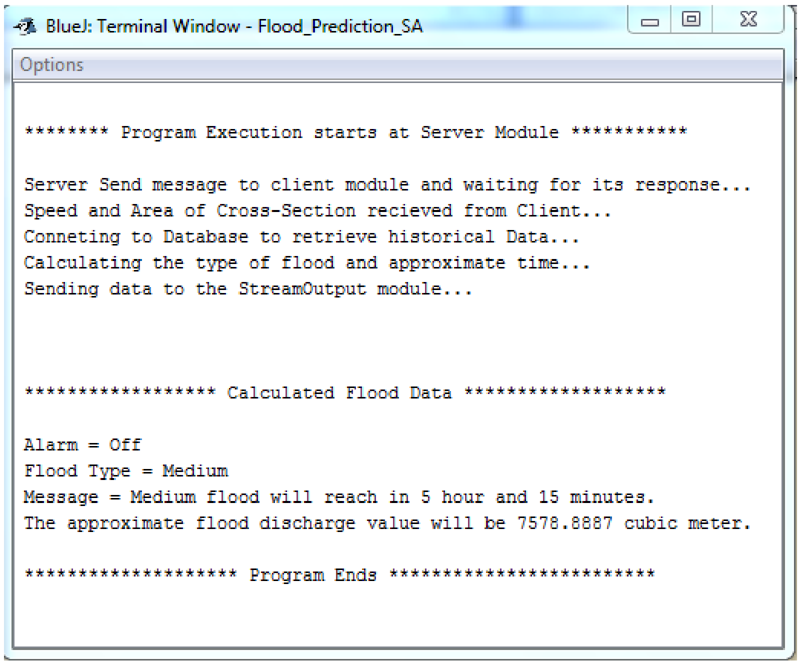
```
******** Program Execution starts at Server Module ***********

Server Send message to client module and waiting for its response...
```

Figure 7: Alert to Client by the Server



```
******** Program Execution starts at Server Module ***********

Server Send message to client module and waiting for its response...
Speed and Area of Cross-Section recieved from Client...
Conneting to Database to retrieve historical Data...
Calculating the type of flood and approximate time...
Sending data to the StreamOutput module...
```

Figure 8: Server Client Communication



```
******** Program Execution starts at Server Module ***********

Server Send message to client module and waiting for its response...
Speed and Area of Cross-Section recieved from Client...
Conneting to Database to retrieve historical Data...
Calculating the type of flood and approximate time...
Sending data to the StreamOutput module...



****************** Calculated Flood Data *******************

Alarm = Off
Flood Type = Medium
Message = Medium flood will reach in 5 hour and 15 minutes.
The approximate flood discharge value will be 7578.8887 cubic meter.

******************** Program Ends *************************
```

Figure 9: Calculated Flood Data.

6. ANALYSIS OF APPROACH:

In this paper, we improve the work of Asif's at al, [4] in four facets as follows:

(a) In the previous model clients calculate the discharge of water while in our model clients only send cross-sectional area and speed of flood to the server, which is an initial fundamental change for accurate decision making

(b) In their model, sequential approach is used while in our model parallel approach which increases the efficiency of the system

(c) In their model server first predicts the type of flood and then connect to the database to find the inundation areas while in our approach the server first connects to the database, find the historical data, compare the existing flood with the past and decide accordingly

(d) In the previous model, working of the system is not shown if the connection to the database failed, while in this approach in case of failure of connection to the database, it tries again and again.

(e) Further, a conceptual general model is developed for a single client which is then developed to an algorithm and then it is specified and checked by using Z and finally transform towards implementation using BlueJ.

7. CONCLUSION

In this study we improved the existing work of flood prediction in term of quality and efficiency. Initially, a conceptual model for a single client agent in real-time distributed systems is developed, and then its algorithm is constructed and further transformed into Z specification by using of the Z/Eves tool. Z/Eves tool provides the environment to make specification and to check its proof. Further, the specified and verified form of model is implemented using BlueJ of Java programming language. The developed model is general in nature and can be applied to any other flooded situation with many clients connected to the server.

REFERENCES

[1]. Africa, E. (2011). Presentation to the Portfolio Committee: Update on the Management of Recent Disaster Incidents. Presentation given the Parliamentary Portfolio Committee on Cooperative Governance and Traditional Affairs. Cape Town: Parliament of South Africa.

[2]. Ahmad, S., & Simonovic, S. P. (2000, 7). Dynamic modeling of flood management policies. (pp. 6-10). Bergen, Norway: In Proceedings of the 18th International Conference of the System Dynamics Society: Sustainability in the Third Millennium.

[3]. Anding, P., Muzhuang, Y., & Bishan, C. (2010). Flood Hazard Evaluation and GIS in Guangzhou. (pp. 1-4). Guangzhou: In Multimedia Technology (ICMT), 2010 International Conference on IEEE.

[4]. Asif, R. S. (2012). Analysis and Design of Flood Prediction Model using Mobile Agents. Lahore: Roa Sohail Iqbal Asif, University of Central Punjab.
[5]. Balica, S. F., Popescu, I., Beevers, L., & Wright, N. G. (2013). Parametric and physically based modelling techniques for flood risk and vulnerability assessment: A comparison. Environmental Modelling & Software, 41, 84–92
[6]. Ghazali, J. N., & Kamisn, A. (2008). A Real Time Simulation and Modeling of Flood Hazard. 12th WSEAS International Conference on System (pp. 438-443). Heraklion: WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering (No. 12). WSEAS.
[7]. Iqbal, R. S., Alshmari, M., Khan, S. A., Zafar, N. A., & Islam, S. (2013, January). A Mobile Agent-Based Algorithm for Prediction of Inundation Area. Research Journal of Recent Sciences, 3(1), 72-77.
[8]. Ministry of lands Japan. (2000). Japan Tokai Heavy Rain. WMO/GWP Associated Program on Flood Management. Tokai: WMO/GWP Associated Programme on Flood Management.
[9]. Monga, O. (2000). Anfas—data fusion for flood analysis and decision support. (pp. 27-29). Europe: In Proceedings of an International European-Asian Workshop. Ecosystem & Flood 2000.
[10]. Nelson, S. A. (2002). River systems and causes of flooding. Tulane: Tulane University.
[11]. Sengupta, S. K., Bales, J. D., Jubach, R., Scott, A. C., & Kane, M. D. (2006, December). Flood Forecasting and Inundation Mapping in the Mahanadi River Basin. A Collaborative Effort between India and the United States. Odisha: A Collaborative Effort between India and the United States.
[12]. Zhao, D. H., Shen, H. W., Tabios III, G. Q., Lai, J. S., & Tan, W. Y. (1994). Finite-volume two-dimensional unsteady-flow model for river basins. Journal of Hydraulic Engineering, 120(7), 863-883.
[13]. Zuma, B. M., Luyt, C. D., Chirenda, T., & Tandlich, R. (2012). Flood Disaster Management in South Africa: Legislative Framework and Current Challenges. Konya: INTECH Open Access Publisher.