# AN AUTOMATED FRAMEWORK FOR PRECIPITATION-RELATED DECISION MAKING

A. M. H. D. Adikari

(188080F)

Degree of Master of Science (Major Component Research)

Department of Computer Science & Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

December 2021

# AN AUTOMATED FRAMEWORK FOR PRECIPITATION-RELATED DECISION MAKING

Adikari Mudiyanselage Hasitha Dhananjaya Adikari

(188080F)

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science & Engineering

Department of Computer Science & Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

December 2021

# DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

**Candidate**

2021/12/12

..... .....

*A.M.H.D.Adikari* *Date*

The above candidate has carried out research for the Masters Dissertation under my supervision.

**Supervisors**

2021/12/12

*Dr. H.M.N. Dilum Bandara* *Date*

12/12/2021

..... .....

*Dr. Charith Chitraranjan* *Date*

**ACKNOWLEDGEMENT**

I wish to thank my evaluation panel members who were more than generous with their expertise and precious time.

Also, I owe my deepest gratitude to my supervisors, Dr Dilum Bandar and Dr Charith Chtraranjan of the Department of Computer Science, Faculty of Engineering, University of Moratuwa, for the invaluable support in providing relevant knowledge, advice and supervision throughout the project. Their continuous guidance, inspiring advice and constructive feedback provided this project with great value. This would not have been possible without their expertise and remarkable support.

Nevertheless, I would like to show my gratitude to my supervisor at the research center (CUrW), Dr Srikantha Herath, Team Leader CUrW, Metro Colombo Urban Development Project, Ministry of Mega polis and Western Development Sri Lanka, for his continuous guidance and constructive feedback provided.

I take this opportunity to convey our gratitude towards my wife and my parents who helped me to do my best towards achieving success during the entire project time period. Finally, I would like to thank my colleagues at CUrW for the domain knowledge and feedback throughout this research. Without them, this project would not have been possible.

Also, I would like to thank the entire academic and non-academic staff of the Department of Computer Science and Engineering for their kindness extended to me in every aspect.

Last but not least, I am grateful for all the people who supported me throughout this research in various means.

## ABSTRACT

With the effects of rapid urbanization and climate change, weather forecasting plays a vital role in disaster risk controlling and mitigation activities. Generating weather forecasts using numerical weather prediction methods is a tedious process as it requires multiple combinations of weather model workflows. Due to the weather's chaotic nature, field experts frequently modify these static workflows to cater to their decision-making requirements. Moreover, infrequent weather events make these workflows more complicated and challenging to handle manually. There is a need for a decision support system (DSS) to build and update workflows dynamically with these circumstances. After studying the architectures of existing DSSs from different fields, we understand that they cannot handle all weather-related decision-making requirements. Therefore, we present a generic decision support system framework to create and control complex and dynamic weather model workflows. The proposed framework supports three types of decision-making conditions: accuracy-based, infrequent-event, and pump/gate control. The framework can terminate or dynamically update weather model workflow paths in accuracy-based decisions. In infrequent-event decisions, the framework identifies the weather events. In pump-control decisions, the framework attempts to find an optimized control strategy that minimizes the flood risk in the given catchment area. In addition to the above decisions, the system provides relevant workflow strategies for handling unexpected weather conditions. To demonstrate the utility of accuracy-based decision-making, we executed four workflow runs over six months (on randomly selected days for each month). We were able to achieve a 100% accuracy level with manual verification. Pump/gate control strategies were tested using the data from the 2010 Flood event in the Kelani basin area in Sri Lanka. Pump strategy decisions also had 100% accuracy on logic evaluation and model selection. The proposed framework is deployed in a Google cloud platform of the Center for Urban Water, Sri Lanka, for flood-related forecasts.

Keywords — decision support system, dynamic workflow management, weather forecasting

**TABLE OF CONTENTS**

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| CPU | Central Processing Unit |
| CUrW | Center for Urban Water |
| DAG | Directed Acyclic Graph |
| DSS | Decision Support System |
| FTP | File Transfer Protocol |
| GCP | Google Cloud Platform |
| GIS | Geographic Information System |
| KLB | Kelani Lower Basin |
| KUB | Kelani Upper Basin |
| NWP | Numerical weather prediction |
| RAM | Random Access Memory |
| RMSE | Root Mean Squared Error |
| SF | Statistical Forecasting |
| VM | Virtual Machine |
| WRF | Weather Research Forecasting |

# 1 INTRODUCTION

## 1.1 Background

Rapid urbanization and climate change have increased the vulnerability of many metropolitan areas to catastrophic floods. Weather forecasting plays a vital role in disaster risk mitigation and managing activities in such situations. Generally, weather forecasting has two main categories:

- Numerical weather prediction (NWP) - Uses current weather observations and processes these data with computer models to forecast the weather's future state. Knowing the weather's current state is just as important as the numerical computer models processing the data. Current weather observations serve as input to the numerical computer models through a process known as data assimilation to produce outputs of temperature, precipitation, and hundreds of other meteorological elements from the oceans to the top of the atmosphere [1].

- Statistical Forecasting (SF) - Estimating the likelihood of an event taking place in the future, based on available data. Statistical forecasting concentrates on using the past to predict the future by identifying trends, patterns [2].

When forecasting weather conditions for disaster controlling purposes or agricultural needs, we need to integrate several modeling systems for weather monitoring, forecasting, and risk estimation activities. Center for Urban Waters (CUrW), Sri Lanka, which has been set up for flood control and water management in the Metro Colombo area, uses a set of numerical models like WRF [3], HEC-HMS [4], and FLO-2D [5] for forecast generation on different weather parameters. Each model has pre- and post-processing units to generate inputs and output extraction.

WRF model execution flow in Figure 1.1 illustrates that most of these weather models are built as an integration of several applications, data preprocessing unit (input), data formatting/scaling systems (WPS), actual weather model (WRF), and post-processing unit (output). Therefore, the execution of weather models is a tedious process due to their complex structures.

Moreover, there is a cascade relationship between these model workflows where the output of one model workflow is transferred to another model workflow as the input. As Figure 1.2 depicts, we should run the WRF model workflow first, the HEC-HMS model workflow, and finally, the FLO-2D model workflow. For each model workflow except WRF model workflow, we need observed data and previous model workflows' output to input the descendent model workflows.

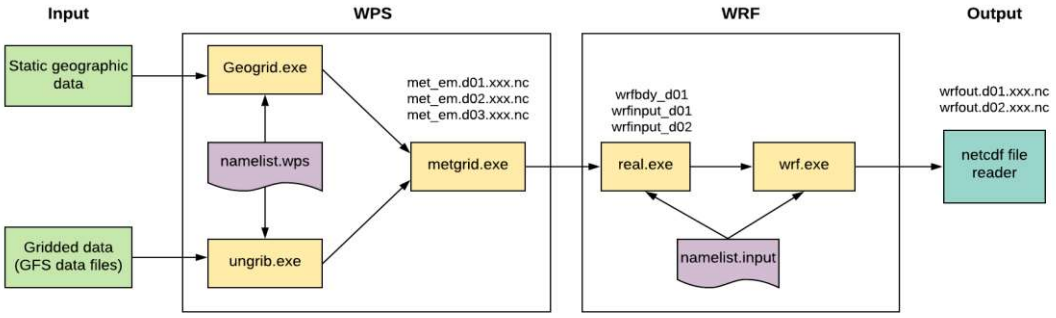


Figure 1.1 WRF model execution flow

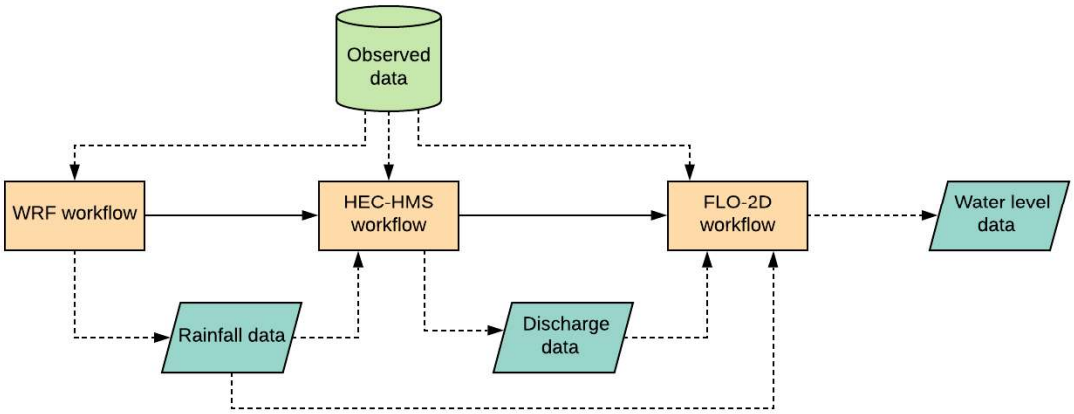


Figure 1.2 Cascade relationship between models.

Therefore, field experts need to run all three models in a defined order to forecast the water levels of a particular area. Further, domain experts need to update these models' configurations, as they are optimized for diverse weather conditions. For example, the WRF model used to forecast rainfall – can have multiple configurations optimized for monsoon seasons or cyclonic conditions. Prevailing weather conditions and the purpose of previous weather model execution impact the next model execution decision. For example, for hydrodynamic models like FLO-2D, field experts use different configurations with different spatial resolutions according to the requirements of the water level forecast (e.g., resolution level of 150 m or 250 m), flood map generation (e.g., resolution level of 150 m), and risk estimation (e.g., resolution level of 10 m or 30 m).

## 1.2 Motivation

In CUrW, field experts regularly monitor and control weather model execution workflows and analyze model results for better forecasts. However, the weather is inherently chaotic (i.e., extreme dynamism and high sensitivity to initial conditions).

Moreover, infrequent events such as flash floods, canal water pumping systems breakdowns, or reservoir dam breakages are inevitable. To cater, all these requirements, field experts should:

- Execute multiple models parallel
- Analyze model results
- Terminate unfitted execution
- Change complete or segments of model workflows.

However, running a particular weather model is not a straightforward task due to input data requirements (with the cascade relationship between models). Furthermore, some model runs like WRF and FLO-2D are costly in execution time and CPU usage. See Table 1.1 with the number of forecast days (output time series length), CPU utilization time will vary in each model execution. Model input preparation (pre-processing) also varies significantly with the output time series length (as both input and output time series lengths should be equal). Furthermore, FLO-2D models have different completion times with the defined grid's resolution as when grid count increases, execution time increases.

Table 1.1 Resource consumption of weather model execution

| Model | CPU Time | CPU Usage | Forecast Days | Machine Configurations |
|---|---|---|---|---|
| WRF | 2-3 hours | 25 % | 3 days (1 day observed) | 12 physical cores<br>16 GB memory |
| FLO-2D (150m resolution) | 3-4 hours | 100% | 8 days (with 5 days observed) | 8 cores<br>12 GB memory |

Hence, the decision to trigger a model execution and terminate a model execution should be critically evaluated before making it. Also, the parallel execution of the same model with different configurations affects system resources. When making the above decisions, it needs to evaluate each model's results for their accuracy requirements and current catchment area conditions with the observed data. Therefore, field experts must always be available for these analyses and making decisions.

Using an intelligent Decision Support System (DSS), we can monitor and control these models' execute on and data flow without human support or intervention. Continuously monitoring workflows, analyzing model results, and using them as an input to the successive subsequent iterations or next following levels (closed-loop architecture) [6] will be advantageous for system resource utilizations. Building workflows dynamically will help handle infrequent events and allow field experts to enforce their decision critically in system operation. Several related works on this topic have limitations on being specific to the geographical area [7], the purpose of the system [8], or the field system is related [9]. Hence this proposed framework will address the void of a dynamic, adaptable, and feedback-driven decision support system.

## 1.3 Problem statement

Precipitation uses as the main parameter for weather forecasting. Based on the precipitation, other relevant parameters like water discharge and water levels are derived using weather models.

The proposed system should be adaptable to handle infrequent rare weather conditions, such as the breakdown of pumps and gates, flash floods, or reservoir damage. With static workflows and static logic, we cannot handle these exceptional cases. Therefore, it needs to be an automated system that can make relevant changes to the decision flows dynamically.

We refer to the previous workflow status and the current weather condition as feedback in this context. The predecessor model directly affects its successor models' accuracy when considering the cascade relationship between models. Thus, the predecessor model's execution status is essential in deciding which model to run next.

Moreover, the current weather status is inevitable in weather forecasting and related decision-making activities. For example, the current rainfall intensity of a particular area and current water levels of the canal system in that area are crucial for decisions related to flood risk mitigation, which should also provide feedback to the model execution decision.

Therefore, the research problem to be addressed by this research can be formulated as:

*How to develop a precipitation-related DSS that is adaptable and feedback-driven?*

## 1.4 Objectives

As a solution to the above-discussed research problem, the following objectives are to be achieved:

- To build a framework to analyze, monitor, and control complex weather workflows automatically. While minimizing the manual interaction system should analyze current weather/workflow conditions and make changes to workflows as necessary
- To dynamically update these complex workflows to specific weather conditions. With the chaotic nature of weather, it is almost impossible to stick to simple static workflows. Therefore, features like branching, decision-making, and accuracy checking need to support adapting to the diverse weather conditions
- To use current observed weather data (rainfall and water level) for decision making in workflows
- To develop a rule engine with dynamic rule variables which allow users to create different rules using dynamic rule variables

## 1.5 Outline

The rest of the thesis is organized as follows: Chapter 2 presents an overview of existing various decision support systems. In that, we are discussing their architectures and limitations. Moreover, it contains a brief description of weather models used in the proposed framework and the current production system of CUrW. Also, it covers Apache Airflow, Docker, and their reason for the selection. Chapter 3 presents the new framework that we propose by explaining the high-level architecture of our approach. Also, we have discussed the main system components and how they are integrated to achieve the research objectives. Performance analysis is presented in Chapter 4, where we discuss implementation details and its performance. We have discussed the performance of the three major significant decisions made by the proposed framework. Finally, Chapter 5 contains the conclusion, research limitations, and future work.

# 2 LITERATURE REVIEW

Decision Support Systems (DSS) represent a concept of computers' role within the decision-making process [10]. Another definition would be DSS is a set of related computer programs and data required to assist with analysis and decision-making within an organization [11].

This literature review was conducted to understand both pros and cons of existing DSS architectures from diverse backgrounds and developed for different purposes. Section 2.1 presents existing Decision support systems architecture from diverse fields. Weather models selected for the proposed solution and their features are discussed in Section 2.2. The workflows management system selected for the proposed solution and justification for the selection is presented in Section 2.3. Section 2.3 describes how the existing production system works and its limitations. Finally, in Section 2.4, we discuss another software tool used in implementing the proposed system.

## 2.1 Architectural aspects of existing decision support system implementations

We have gathered related work from different backgrounds like weather management, reservoir water control, and medicine. Our primary focus is to understand the main design features and functionality of existing DSSs' architectures.

### 2.1.1 Collaborative Adaptive Sensing of Atmosphere (CASA) [6]

McLaughlin et al. stated that Collaborative Adaptive Sensing of the Atmosphere (CASA) is a new paradigm for detecting and predicting hazardous weather using a dense network of short-range, low-powered radars to sense the lowest few kilometers of the earth's atmosphere. The authors have shown this can achieve through a Distributed Collaborative Adaptive Sensing (DCAS) architecture [6].

Distributed and collaborative approach distributed refers to a set of large number radars which is used to collect data in better sensitivity, and collaborative refers to the coordination of beams of these radars to avoid blind spots.

In the workflow management sense, an important aspect is the whole system is built on closed closed-loop architecture. Figure 2.1 shows steps of the data flow which follows closed closed-loop architecture,

- Gathered data through their dense short-range radar network.
- Pushed data to Meteorological Command and Control (MC&C) unit through data ingestion unit
- MC&C will identify meteorological features in the data set and generate a meteorological task list.
- Finally, the resource allocation unit will plan the next scan cycle of the radars to comply with end users' policies.

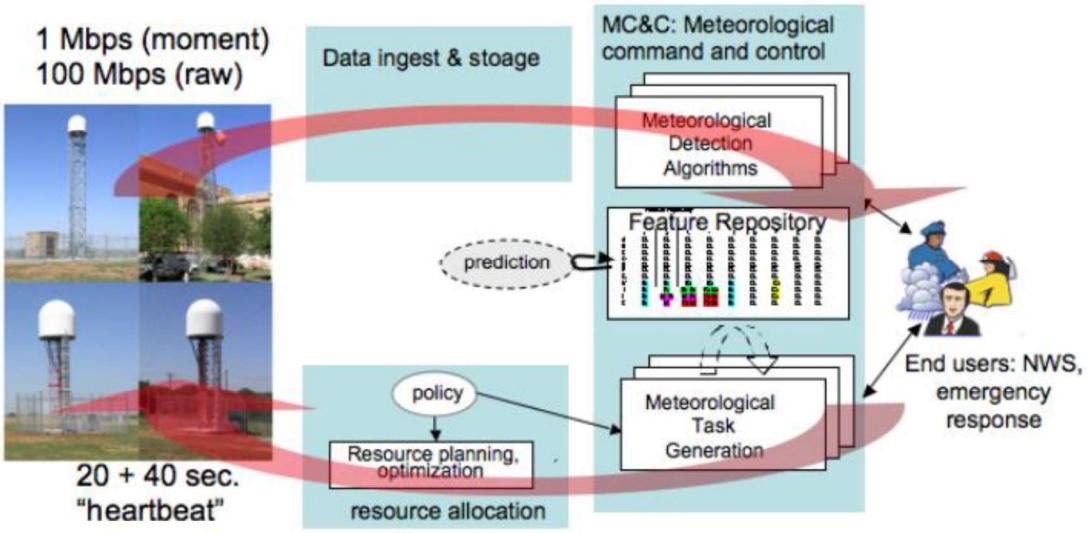


Figure 2.1 Closed-loop architecture of DCAS. Source: [6]

DCAS is a modular architecture-based system (data ingest and storage system, MC&C unit, and resource allocation unit), making it easier to modify and add new functionalities. MC&C unit, the central control part that determines the next scan strategy based on 3 three different information sets [6] on system heartbeat intervals:

- Detections from data obtained in the present heartbeat
- Historical detections from earlier heartbeats
- End-user policies.

Authors have taken the maximum benefit of the latest available data without waiting for the arrival of full complete data for the cycle. Also, unlike existing architectures, like NEXRAD, in which the radars execute predefined scanning patterns and data are pushed to the end end-users independent of their preferences, a DCAS system is a pull system in which the sensing resources dynamically adapt to end-user needs and preferences [6]. Since they have a diverse set of end end-users who have conflicting demands [6] for the system, they have used a preference order for mediating demand conflict policies.

### 2.1.2 Decision Support System for Diagnosing Anemia [9]

This system's primary goal is to reduce the time of admission (of the Iron deficiency anemia -IDA patients) and improve the treatment's effectiveness by ensuring a patient's diagnosis and choosing a treatment method [9].

The proposed solution has two steps; first, identifying patients' symptoms and selecting patients who suffer from anemia; second, categorizing patients' stage of anemia. The final step is to decide on the patients' diagnosis.

For identification and selection, they have built a knowledge base that contains the following data:

- Identification data of the patient
- The social status of the patient
- History data
- Sensations and complaints of the patient
- Medical indicators (analyzes)
- information on the patient's medical history
- Statistical information on the disease

To add to the knowledge base, they get medical experts' direct assistance [9] because the system's decisions will depend on this. It is necessary to develop a knowledge base for an expert system to diagnose anemia, which will support the doctor's decision-making [14]. For the initial screening process, they have identified a list of symptoms shown by patients in IDA development and a set of parameters from biochemical analysis of blood data. Then they have categorized parameters related to each symptom.

Symptoms of IDA:

```
SI (x) = a1x1 + x2 + ... + xm are the symptoms of stage I;
SII (x) = xm + 1 + ... + xn are the symptoms of stage II;
SIII (x) = xn + 1 + ... + xl are the symptoms of stage III.
```
[9]

When users enter each presence of the patient's symptom, the system will generate a score with the above symptom parameter relation. Using this score patient's stage will be decided. The authors have employed a fuzzy-based logical rule set that has helped them handle the symptoms and parameters' vagueness.

For categorization, the authors have built a decision tree that helps to identify the stage of iron deficiency anemia. To determine the diagnosis, the authors have built a static group rule set. With this system, authors have claimed that diagnosing a patient and choosing a treatment method can reduce admission time and improve the effectiveness of treatment [9].

### 2.1.3 A Decision Support System for the Management of the Sacca di Goro (Italy) [7]

This system tries to fulfill the need for a common and flexible framework to ease integrating the outputs of different models and analyses and deal with the diversity of socio-economic and environmental characteristics of several application sites [7]. They had to find the right balance between environmental safety with social and economic improvements when dealing with socio-economic factors. They integrate different kinds of mathematical and analytical models that fit the lagoon ecosystem, like biogeochemical, hydrodynamic, ecological, and socio-economic models.

Therefore, the authors have identified and structured the decision problem in the following terms:

- Control options – the pool of alternatives that change the behavior of the system
- Criteria – an indicator that measures the performance of the system under each alternative control options
- Objectives – target value of the performance indicators
- Constraints – threshold values

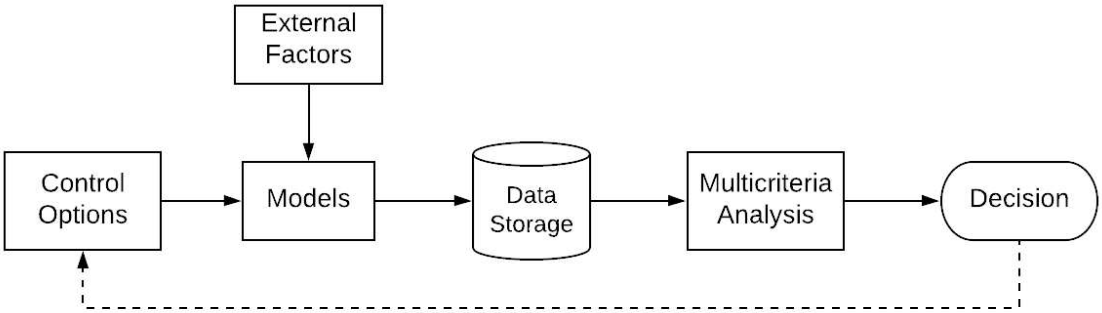


Figure 2.2 Feedback model of the management DSS. Source: [7]

As shown in Figure 2.2, Decision output feds to the control options following feedback model allow users to adapt control options set according to their preferences. The authors have employed a multi-criteria analysis method Analytical Hierarchy Process (AHP), to select one alternative control option. AHP considers a set of evaluation criteria and a set of alternative options, among which the best decision is to be made. It is important to recall that because some of the criteria could be contrasting, it is not valid in general that the best option is the one that optimizes every single criterion, rather the one achieving the most suitable trade-off among the different criteria [7].

### 2.1.4 Dynamic Decision Support System Based on Bayesian Networks [12]

A medical decision support system helps physicians understand and prevent Nosocomial Infections (NI). A nosocomial infection contracts because of an infection or toxin in a specific location, such as a hospital. People now use nosocomial infections interchangeably with the terms health-care care-associated infections (HAIs) and hospital-acquired infections [13]. Generally, NI occurs within 48 48-hours of hospital admission, three days of discharge, or 30 30-days of an operation. They affect one in ten patients admitted to the hospital [14].

Authors have used Knowledge Discovery Databases (KDD) as the knowledge base of this DSS. The knowledge required is in the patients' records [12]. Using a traditional tool like dashboard or Enterprise resource planning (ERP) cannot handle this data's dynamic and temporal nature. Also, they cannot discover and maintain the association between the data. KDD is also known as Data mining, refers to the nontrivial extraction of implicit, previously unknown, and potentially useful information from data stored in databases [15].

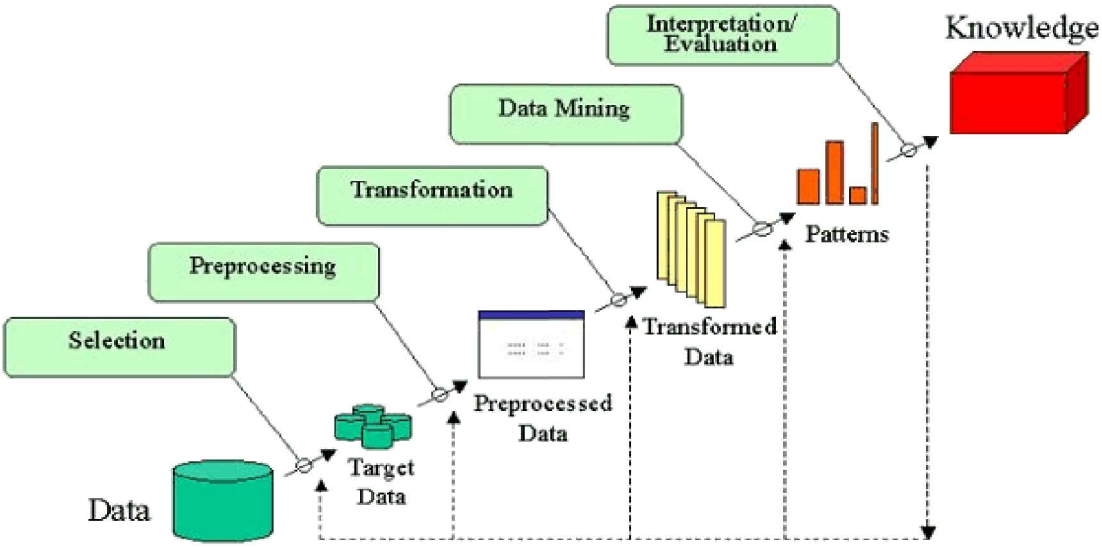

Figure 2.3 Knowledge discovery databases process. Source: [16]

As shown in Figure 2.3, the KDD process includes these steps:

- Selection – After understanding user goals and application domain, select the data subset focusing on discovery.
- Preprocessing – Removal of noise or outliers, Strategies for handling missing data fields [16]
- Transformation – Process of transforming data into appropriate form required by mining procedure [15]
- Data mining – Selecting method(s) to be used for searching for patterns in the data. Deciding the model and parameters may be appropriate. Matching a particular data mining method with the overall criteria of the KDD process [16].
- Interpretation/Evaluation

Authors have identified factors supporting the appearance of the infections and classified them into two main categories: static data (e.g., - age, gender, and weight) and temporal data (e.g., - intubation and Central Venous Catheter). This temporal data is vital that today's values affect tomorrow's values, the characteristic of conditional probability. For that, the authors have used Bayesian networks (BN). As shown in Figure 2.4, the causal graph of the dynamic BNs are used to model the conditional dependencies between temporal data themselves and between static and temporal data in directed graphs.

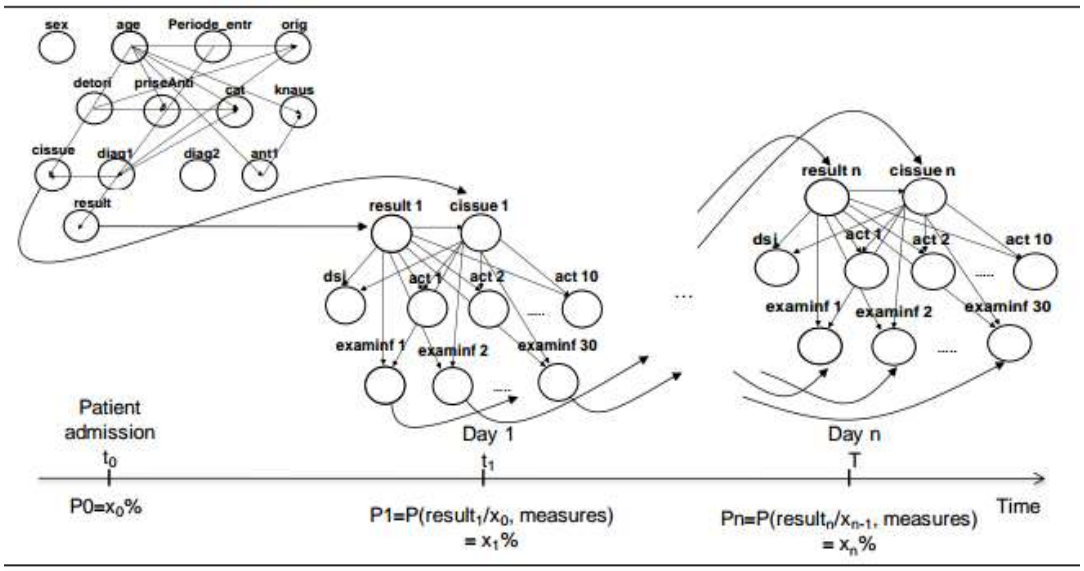


Figure 2.4 The Causal graph of the Dynamic Bayesian Network. Source: [12]

### 2.1.5 An ontology-based interpretable fuzzy decision support system for diabetes diagnosis [17]

Diabetes Mellitus (DM) is a chronic disease associated with an increased risk of morbidity and mortality [17]. The DM's asymptomatic nature requires experts in this area to identify it and examine the patient's complete profile. Physicians must first gather signs of disease, including a patient's past medical history, symptoms, family history, related complications, and physical examinations.

However, these procedures increase patient examination duration, especially to detail study patients' medical history. An automated Clinical Decision Support System (CDSS) can help physicians analyze electronic health care records (EHR), which can derive different hospitals and analyze patient information to make accurate and timely diagnosis decisions.

In this paper, the authors have proposed a new semantically interpretable Fuzzy rule-based system (FRBS) framework for diagnosing diabetes. Figure 2.5 indicates that the proposed system has two layers. The first layer has six sub-FRBSs to determine the patient's risk level according to specific dimensions. According to the medical expert opinions, medical literature, and diabetes clinical practice guidelines (CPGs), the data are grouped according to their medical relationships, where dissonant data are not in the same category [17].

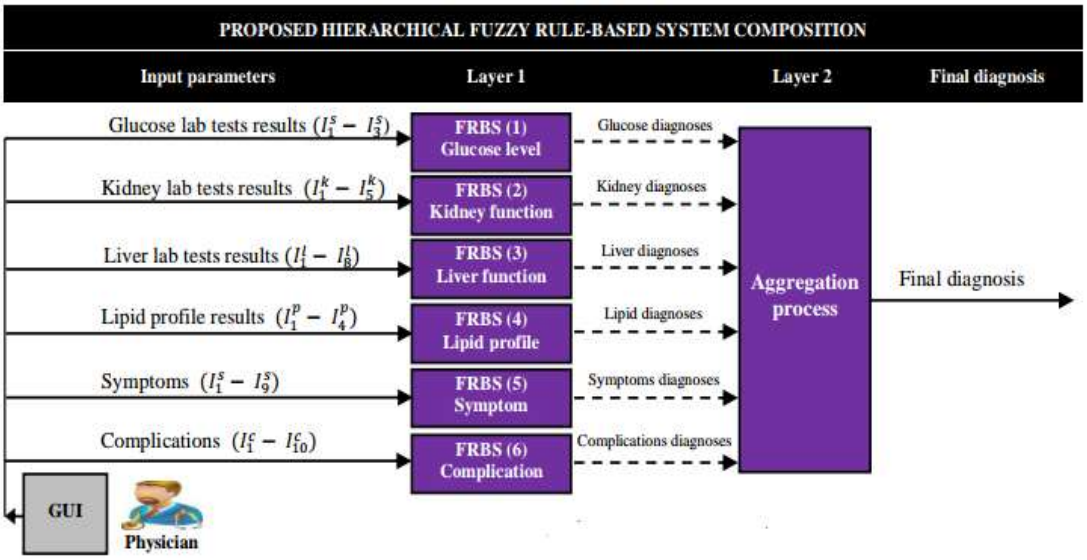


Figure 2.5 Hierarchical FRBS system. Source: [17]

In layer 1, there are six FRBSs. Each FRBS will calculate the risk level of developing diabetes according to the relevant lab tests or input features. FBRS 1 - 4 have numerical parameters only, FBRS 5 has numerical and categorical features, and FBRS 6 has categorical features only. Most of the existing systems concentrate on one type of data to diagnose DM, namely numerical or continuous data [18] [19]. Using only numerical features makes it hard to diagnose DM since it depends on a varied data set.

Authors have defined weights for each FBRS since some feature sets can be more critical than others though every feature set is essential to the system. To calculate related weights, they first requested three-domain experts' opinions and then applied a well-known multi-criteria decision-making (MCDM) technique called the fuzzy AHP technique [19].

The second layer is based on the Mamdani min-max inference mechanism [17], which has the advantages of being intuitive, well-suited to human input, a more interpretable rule base, and widespread acceptance [20]. To give a much broader sense and avoid the system limiting to a defined symptom set, authors have employed a semantic similarity engine to extract data from various social media and EHR.

### 2.1.6 DSS for Reservoirs Operation Based on Execution of Formal Models [8]

Controlling and maintaining reservoirs' water levels is critical in flood seasons, especially when concentration-time is very short. In this paper, the authors have presented a DSS based on the execution of formal models using model checking for this purpose [8]. Existing dam operating systems use a catalog of predefined rules created using parameters like reservoir level, forecast, and current downstream capacity. Unexpected situations like breakdowns also should be handled using these rules [8].

The system has two main operating modes. The first is simulation mode, which carries out a predefined maneuver, and the second one is the optimization mode, which selects the best maneuver from all possible maneuvers. In optimization mode, users can maintain the dam level under some value. Further, the system will provide optimized outlet controlling methods to achieve the target. Also, using simulation mode, users can understand the dam behavior for a given discharge policy.

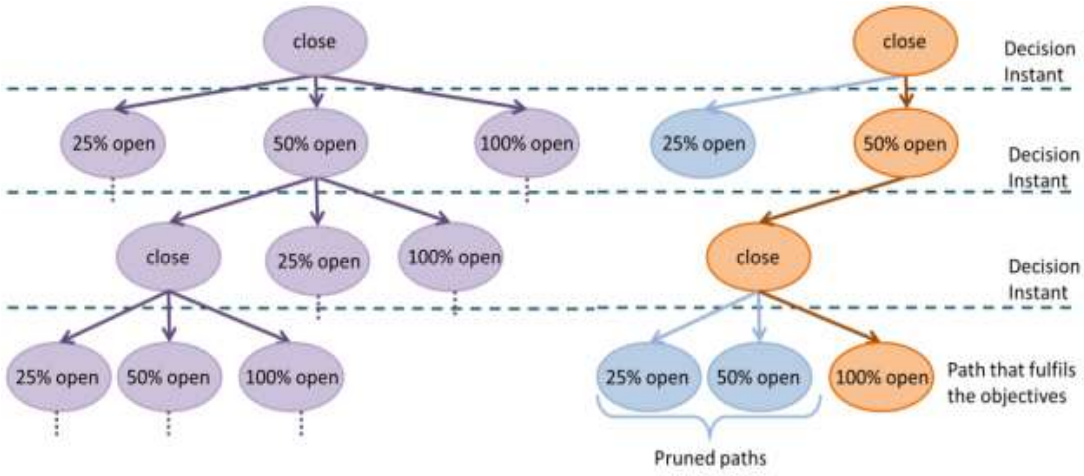


Figure 2.6 Decision flow in the optimization mode. Source: [8]

A basic model checking algorithm performs an exhaustive search of all the system's possible execution paths. Reachability graph shown in Figure 2.6 shows these execution paths, checks whether the user-defined objectives are fulfilled or not [8].

## 2.2 Weather models

Weather models are software systems used to simulate and forecast various weather parameters which are listed in Table 2.1. CUrW uses flowing weather models in different categories for its operations, makes predictions, generates flood maps and flood risk analysis.

Table 2.1 Weather models

| Category | Parameter | Weather model |
|---|---|---|
| Numerical | Precipitation | WRF |
| Hydrology | Water Discharge | HEC-HMS, SHER |
| Hydrodynamic | Water Level (depth/height) | FLO-2D, MIKE 11 |
| Risk profiling | - | GIS |

With the different configurations, one model type can have multiple derived models used for various requirements. Table 2.2 listed weather model execution requirements for each model types. By changing the WRF model's physics parameters, we can create new models optimized for different weather conditions.

In HEC-HMS, we use a single catchment model and multiple catchment model to increase the model results' accuracy. Also, we can build different FLO-2D models with different resolution levels as 250m models are used for primary decision making, 150m models for pump operations, and other high-resolution models for risk assessment purposes.

Table 2.2 Weather model execution requirements

| Model | Configuration | Requirement (Purpose) |
|---|---|---|
| WRF | C - Cyclonic, A - Anticyclonic, E - East, SE - South-East | To adapt to different weather conditions |
| HEC-HMS | Single Catchment, Multiple catchments (Distributed mode) | Increase the accuracy levels |
| FLO-2D | 250m, 150m, 30m, 10m | • 250m - Decision making<br>• 150m - Pump controlling (With different pump configurations and flood plain conditions, there are multiple models with this same resolution level)<br>• 30m, 10m - Flood map generation, risk assessment |

### 2.2.1 Weather Research and Forecasting (WRF) Model

WRF is a next-generation mesoscale numerical weather prediction system designed for atmospheric research and operational forecasting applications [21]. Before using the model for forecasting in a certain geographical area, it should be calibrated for the selected area and extreme events. As seen Figure 1.1, WRF consists of WPS and WRF. Both are required configuration files which include grid size, forecast duration, and set of physics parameters that define the model.

1. WPS (WRF Preprocessing System) - configured by namelist.wps
    - geogrid.exe - static domain information
    - ungrib.exe - convert Grib files to a specific binary file (intermediate file format)
    - metgrid.exe - horizontal interpolator of FILEs to domain resolution
2. WRF model - configured by namelist.input
    - real.exe - vertical interpolator
    - wrf.exe - meteorological model

The forecast generated in this model will be used as input data to the next following models (HEC-HMS and FLO-2D).

### 2.2.2 Hydrological Modeling System (HEC-HMS)

The Hydrologic Modeling System is a lumped model designed to simulate the complete hydrologic processes of dendritic watershed systems. Model software consists of data pre/post-processing systems known as HEC-DSSVue.

In CUrW, this model is used to find the forecasted discharge of the river in a defined location. (See inflow point location in Figure 2.9). This model has been developed and calibrated for the Kelani river upper catchment area. CUrW has developed two HEC-HMS models with catchment definition variances as single and multiple catchment models as shown in Figure 2.7. Using the model output, engineers will extract the water discharge time series for the lower catchment area's inlet as Inflow.

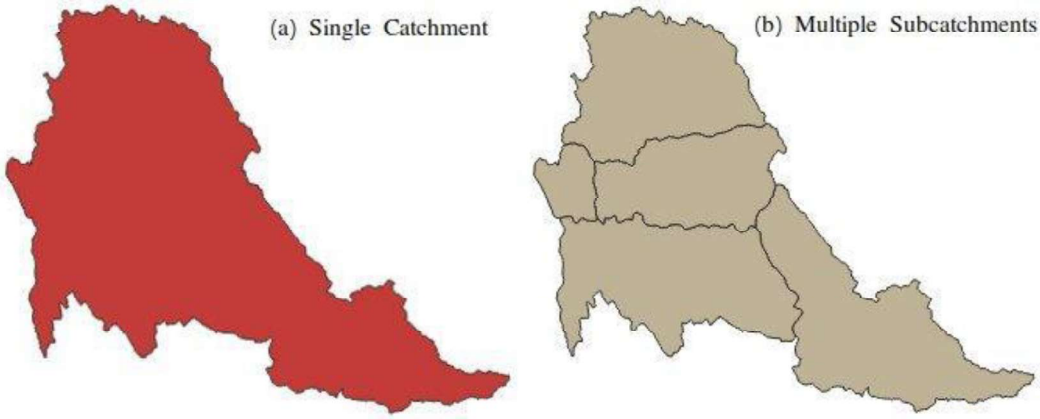


Figure 2.7 HEC-HMS model catchments

### 2.2.3 FLO-2D model

FLO-2D is a 2D hydrological-hydraulic dynamic flood model that simulates channel flow, unconfined overland flow, and street flow over complex topography [22]. To generate forecast water levels using FLO-2D needs rainfall as the primary input. According to the required level of resolution, the catchment area will be divided into grids of 250m, 150m, 30m, or 10m, as shown in Figure 2.8. While increasing the resolution level of the catchment grid, we can get more accurate water level predictions from FLO-2D, but this will increase the application's execution time considerably.

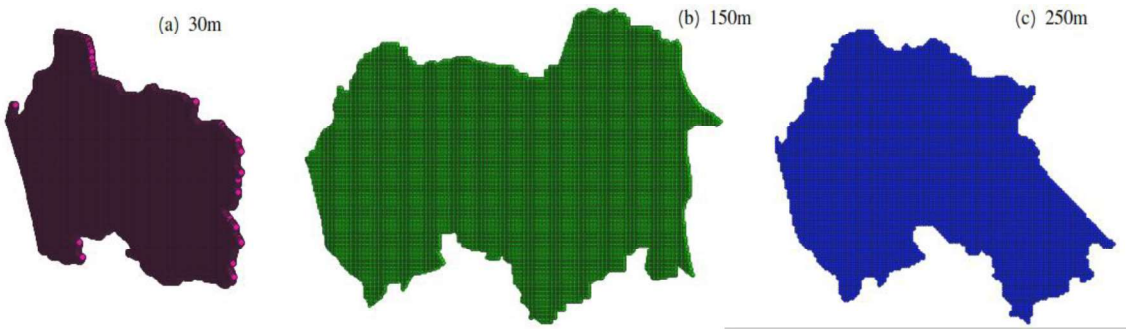


Figure 2.8 FlO-2D grids, (a) 30m grid, (b) 150m grid, (c) 250m grid

Rain input files should contain rain for each grid point for the simulation period. Other than the rain input file, two other data files are used which defines initial and boundary conditions. Figure 2.9 shows the grid point locations for inflow and outflow. Inflow is for initial conditions, and outflow is for boundary conditions.

Inflow represents the entering water levels from upper catchments to the water levels. Following is the list of main input files required for flood forecasting using FLO-2D.:

- RAINCELL.DAT - Main input file contains rainfall for each grid points with the input time steps.
- CHAN.DAT - Channel details are listed against channel cell numbers (initial water levels of the channels in the defined basin area).
- INFLOW.DAT - Discharge time series of inflow location
- OUTFLOW.DAT - Outflow cells and time series of water levels at downstream water level controlling cells (tidal water levels)

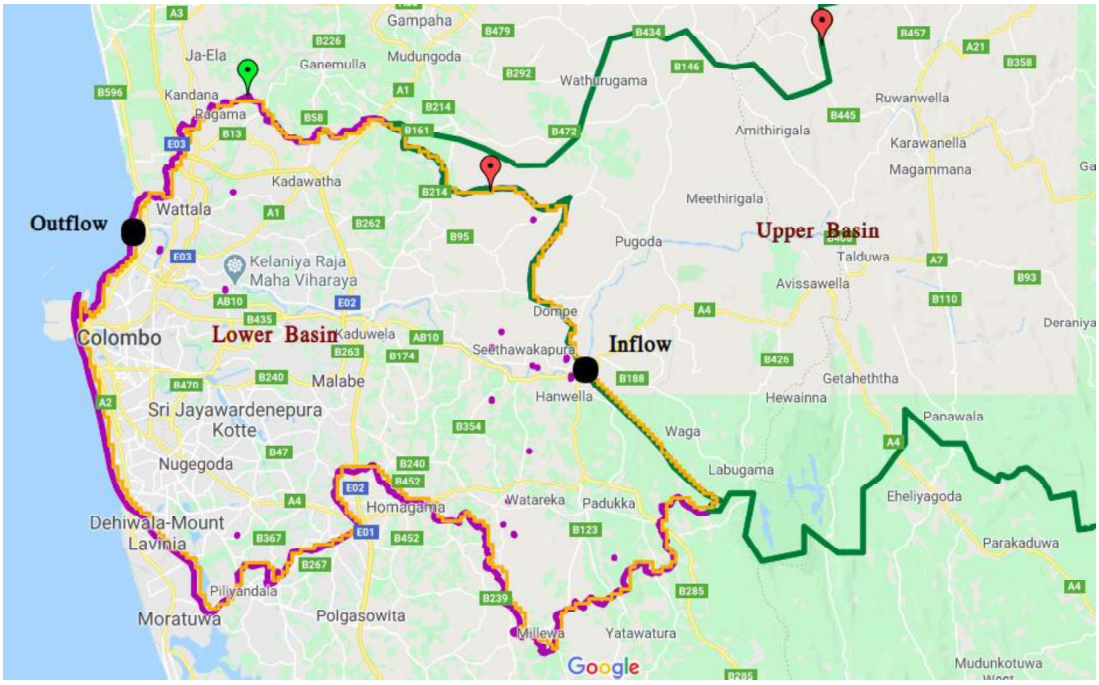

Figure 2.9 Initial and boundary conditions of the catchment

After executing the model using above mentioned input files and model template files, FLO-2D will generate two main output files,

- HYCHAN.OUT - Include water level forecast of the channels
- TIMDEP.OUT - Include water level forecast of the floodplain

## 2.3 Current CUrW production system

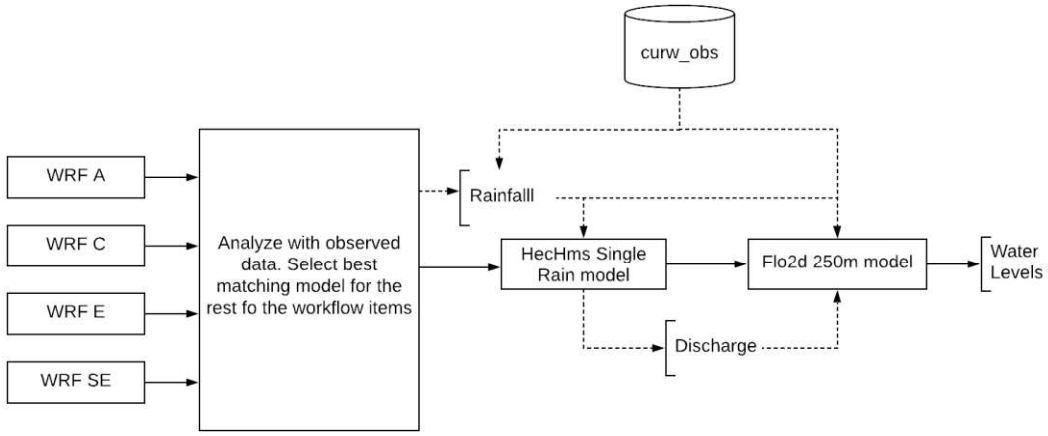


Figure 2.10 Current production system at CUrW

Figure 2.10 illustrates the existing production system used in CUrW for forecast generation by integrating ion of a set of static sub-workflows (weather model workflows) into static parent workflow. Sub-workflows implement using Bash scripts and Cron jobs. Further, Fixed Apache Airflow DAGs use as a single static workflows.

### 2.3.1 WRF sub- workflow

The execution flow of this application set is shown in Figure 1.1. WRF model execution flow. We have containerized separate programming units and static data sets to eliminate the model's installation and execution complexities. That helps us cut down complex workflow to a simple for automation. See Figure 2.11 for WRF sub-workflow structure.

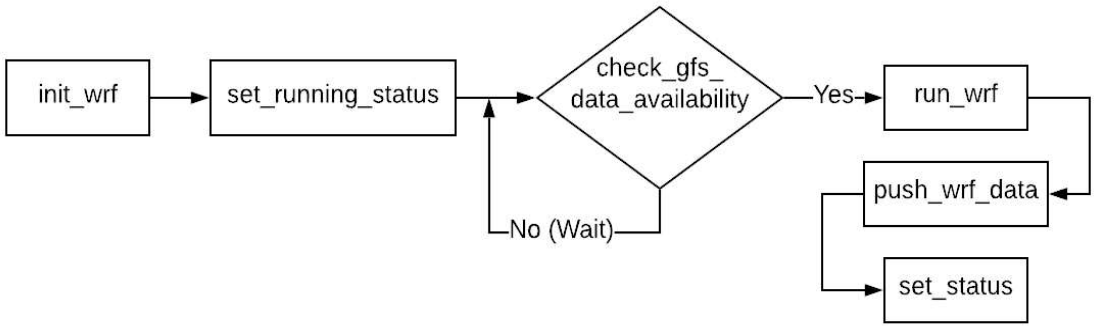


Figure 2.11 WRF sub-workflow structure

### 2.3.2 HEC-HMS sub- workflow

This application has a command-line mode that makes way for automating the execution. The input data file that the HEC-HMS application uses and its output data file are not human-readable (a file with ".dss" file extension). Also, the input file contains rainfall data, and the output file contains discharge data.

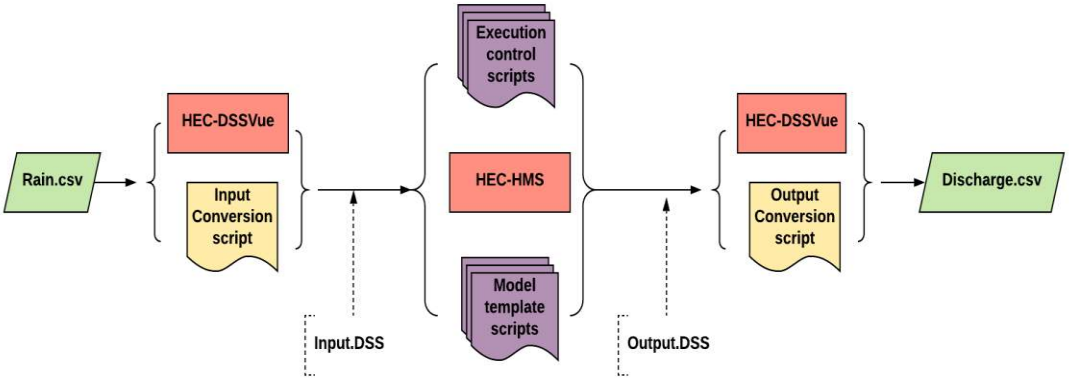


Figure 2.12 HEC-HMS execution flow

As shown in Figure 2.12, HEC-HMS execution flow has following steps,

1. Create Rain.csv file from WRF model out (forecast data) and observed data
2. Using HEC-DSSVue application and input conversion script produce the Input.DSS file
3. Copy the model template scripts and execution control scripts to the execution directory
4. Update model execution scripts with forecast duration, time step length, execution date, and start state information
5. Execute the HEC-HMS model
6. Using HEC-DSSVue application and output conversion script, convert the Output.DSS file into Discharge.csv file

Since WRF models have multiple configurations for various weather conditions and GFS data hours, multiple WRF results are available. Choosing which model result should be used to generate Rain.csv will be a decision made by the user, and that decision can change this parameter value. Also, there is a rainfall time series resulting from Multi Multi-Model Ensemble (MME) as HEC-HMS input. Rain.csv contains mean rainfall for each sub-catchment area which primary catchment (upper Kelani basin-KUB) area is divided into five, as shown in Figure 2.13 (a).

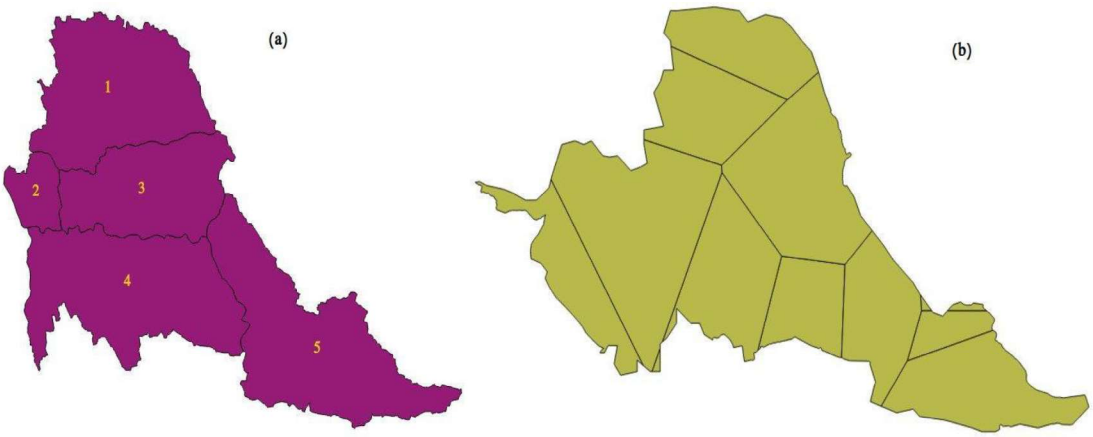


Figure 2.13 (a) KUB Sub-catchments (b) Thiessen Polygon

Rain gauges are installed covering the whole Kelani basin area. So, there can be one or many rain gauges for each sub-catchment. Drawing Thiessen polygon as illustrated in Figure 2.13 (b) for the catchment, we can get effective mean rainfall for each catchment.

However, due to various reasons and faulty conditions, all the rain gauges are not available, so we cannot stick to a static polygon. Therefore, in each run, we are checking available rain gauges and generating Thiessen polygons dynamically, which provides a more accurate mean rainfall calculation.

### 2.3.3 FLO-2D sub- workflow

FLO-2D is proprietary software that has been developed only for the Windows platform. Both input and output files for FLO-2D are human-readable but are in diverse structural formats. Figure 2.14 shows main input files required for running the model:

- INFLOW.DAT – Contains inflow time series and initial water levels of water retention areas. Observed discharge data is calculated from the water level of the given location. Forecast data is generated in two methods,
    - discharge forecasted by HEC-HMS simulation as forecast data
    - Statistical forecast from the observed data
- OUTFLOW.DAT – Outflow cells and time series of water levels at downstream water level controlling cells. Observed data will be collected from water level gauges in the coastal region of the grid. Forecast data is generated in two methods,
    - downloaded from the globally produced data set
    - Statistical forecast from the observed data
- RAINCELL.DAT – Rainfall values corresponding to each time step is listed against cell numbers
- CHAN.DAT – Channel details are listed against channel cell numbers. Contains only the observed data of the channel, which are collected using the water level gauges

From the set of all output files, water levels extracted from the following output files:

- HYCHAN.OUT – Include water level forecast of the channels
- TIMDEP.OUT – Include water level forecast of the floodplain

Figure 2.15 shows FLO-2D model execution steps:

1. Prepare input files
2. Copy input files to model run directory
3. Copy model template scripts to the model run directory
4. Execute FLO-2D model
5. Extract channel water levels and floodplain water levels

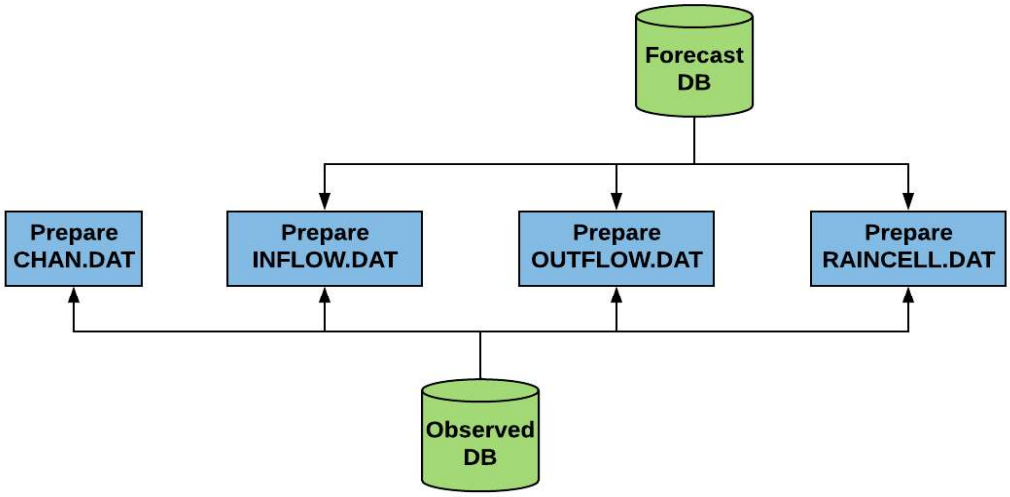


Figure 2.14 FLO-2D input files

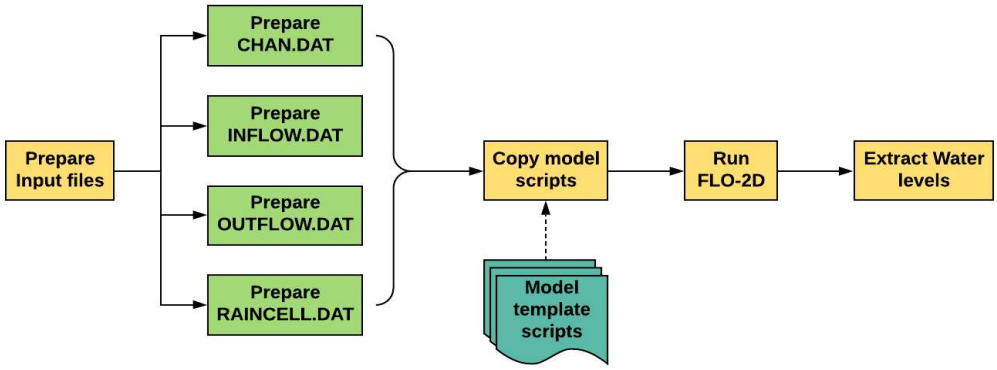


Figure 2.15 FLO-2D model execution workflow

## 2.4 Apache Airflow

Apache Airflow is a platform to programmatically author, schedule, and monitor workflows [23]. Creating a workflow using Airflow should be defined as a Direct Acyclic Graph (DAG). All the tasks in the workflow should be able to execute independently. Importantly they should have the ability to be applied multiple times without producing unintended consequences (idempotent). After creating a workflow, users can either schedule their run using a cron-like scheduling method or execute the workflow immediately as external triggering. Airflow is not a data streaming solution. Tasks do not move data from one to the other (though tasks can exchange metadata). Airflow was selected because of the following features:

- Dynamic nature – Airflow workflows are built as python scripts, which allow room to create workflows dynamically
- Extensible – Contains many in-built units (operators, plugins, and executors) to generate workflows. It also allows users to build their operators, plugins and extend the library so that it fits the level of abstraction that suits the environment
- Elegant – Allow parameterization in scripts using Jinja templating

- Scalable – Allow to scale extensively. Has modular architecture and use message queue to distribute messages among any number of worker nodes
- DAGs – Workflow execution defines in python scripts using tasks (operators and plugins). Define the lists of operators and plugins as tasks and define their dependency as in Figure 2.16.

```
with DAG('my_dag', start_date=datetime(2016, 1, 1)) as dag:
    task_1 = DummyOperator('task_1')
    task_2 = DummyOperator('task_2')
    task_1 >> task_2 # Define dependencies
```

Figure 2.16 Dag definition in Apache Airflow. Source: [23]

- Tasks – Defined using python, Implementation of operators. Trigger operators to start the job.
- Operators – Define what each task is doing, execute bash script, execute SQL query and execute python method. One of the critical features for the selection is users can define any required operators with their definitions.
- Sensors – an Operator that waits (polls) for a specific time
- Plugins – Can be used as an easy way to write, share and activate new sets of features as user-defined required plugins by himself.
- Executors – Executors are the mechanism by which task instances get to run. Airflow supports various executors such as local, Celery, and Kubernetes. Local executors as used to manage the locally deployed workers. Using Celery-executors, airflow workers can work in a distributed environment using message passing as the communication method. Kubernetes executors can be used to manage workers deployed in Docker containers.

## 2.5 Docker

Using Docker, we can create, deploy and run applications inside the containers. A container packages up code and dependencies, so it runs quickly and reliably from one computing environment to another [24] as illustrated in Figure 2.17. Application containerization provides a secure independent environment for the application execution as the container is a self-containment unit.

We are using Docker to utilize the capabilities of containerization on installing weather model software. Most weather model software' requires a set of dependencies and pre/post-processing tools for their execution. We need multiple instances of the same application because we have requirements to execute the same weather model type with different model templates or configurations. As the installation and configuration process is cumbersome, we have adapted the weather model installation to Docker containers. Therefore, we can conveniently move applications to different environments (cloud VMs or high-end workstations).

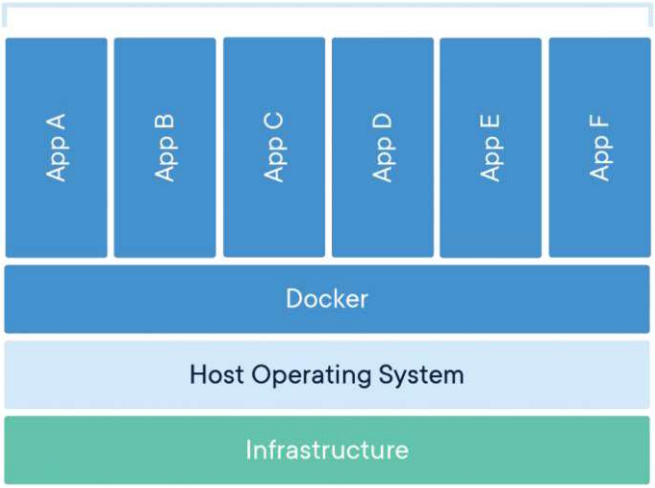

Figure 2.17 Application containerization. Source: [24]

## 2.6 Summary

In the literature review, we have studied existing Decision Support Systems architectures from different fields. Though their built industries are not related, every system contains several main components and features. As components, a knowledge base and a rule engine are common in all systems. A standard feature feedback mechanism should be highlighted. Maintaining a knowledge base on the information needed for decision making and its past decisions is essential to every DSSs. As we cannot handle most weather-related events as individual events, a feedback mechanism is essential in decision-making. Also, they have cascade relationships when it comes to the weather models. So, they should operate in specifically ordered flows for generating valid forecasts.

Also, we can highlight some drawbacks of the existing system. They are tightly attached to the field they were implemented, so it is almost impossible to use them in separate domains. Also, they are limited to single purposes, like detecting cyclones, some diseases, or controlling reservoir water levels. Almost all the systems have defined a static flow for the final decision-making process. So, it is harder to use them in an environment that needs dynamic decision-making workflows.

# 3 PROPOSED SOLUTION

This section discusses the design of the proposed solution, an automated framework for precipitation-related decision-making in detail, and the approaches followed to achieve the stated research objectives through this solution. Section 3.1 provides an overview of the complete system. The main components of the framework that are needed for dynamic workflow generation are discussed in Section 3.2. Section 3.3 discusses rule types, creates rule logics for decision-making using and rule variables used to build logics. Section 3.4 explains how to create dynamic workflows, while Section 3.5 explains their execution.

## 3.1 System overview

This framework was developed following modular architecture; therefore, as shown in Figure 3.1 it integrates multiple components, generating dynamic workflows collectively. As a result, workflows built using this system have no required or static tasks; each item is easily removable and pluggable, improving the dynamic nature further. Therefore, we can build workflows that can adapt to handle unexpected scenarios, feedback-driven for saving system resources, and generate valid and timely results using this system.

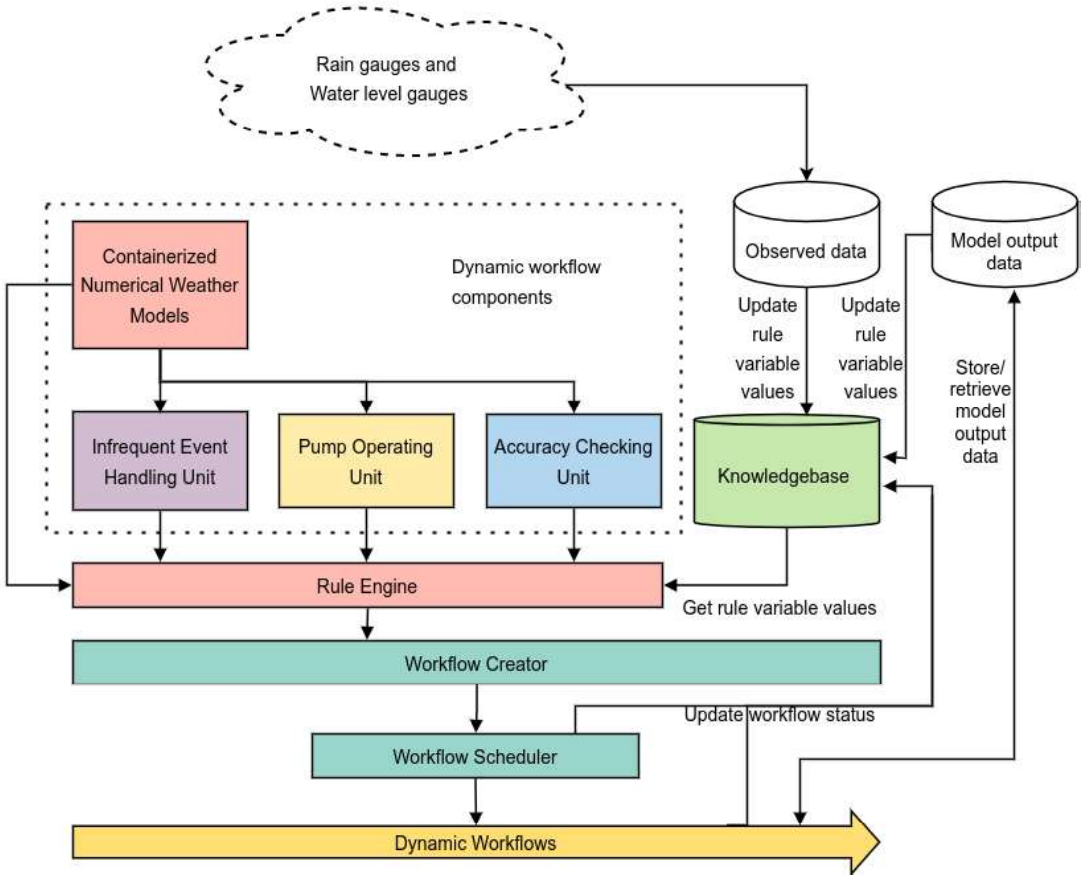


Figure 3.1 System overview

As illustrated in Figure 3.1, the containerized numerical weather models component consists of all the automated weather models which are implemented as sub-workflows. Unexpected Event Handling Unit, Pump Operation Unit, and Accuracy Checking Unit use these sub-workflows with various configurations for each operation. Workflow Creator is the platform to assemble these units and create dynamic workflows while knowledgebase stores rule parameters' values and user-defined rules. Moreover, using Workflow Scheduler, users can schedule workflows using a cron-like scheduling mechanism..

Workflow scheduler (parent workflow that monitors the dynamic workflows) updates current status (ready/running/aborted/failed/error/success) of each dynamic workflow in the knowledgebase, and dynamic workflows themselves update their current status in the knowledgebase as well. Further, current weather status (data from rain and water level gauges) updates the rule variables in configured time intervals (each data type receiving frequencies). Using rule variables like "current/previous workflow status," "current rainfall intensity," and "current water level," users can provide feedback to the workflow execution in real-time.

## 3.2 Dynamic workflow components

Generated dynamic workflows are a collection of one or many of the following workflow components. Users can combine them in their preferred but meaningful order for the execution. Each component can execute independently without any dependency, but the predecessor task's failure may invalidate the successor task's execution due to the logical or theoretical dependency between them.

### 3.2.1 Weather model workflows (sub-workflows)

Each weather model has its execution flow(s). Some models needed specific pre- and post-processing units for input and output formatting before executing the main application. Moreover, some models needed multiple sets of input files with different formatting for the execution. So, each weather model type requires a separate sub-workflow to cater to its execution conditions. These sub-workflows are built as static workflows with a permanent set of tasks. When introducing a new weather model to the system, developers need to create the model's sub-workflow.

Following the standard sub-workflow structure shown in Figure 3.2, we grouped every sub-workflow task into a standard structure using Docker containers and Python scripting. As we can group all the tasks that should be completed before model execution to "pre-processing" tasks and that should be completed after the model execution to "post-processing," every sub-workflow has a standard structure for their weather models. These sub-workflows are built as Airflow dags as they are static flows.


Figure 3.2 Common sub-workflow structure

### 3.2.2 Accuracy checking unit

This unit will evaluate the result of a model that is accurate enough for the use of the next model as input and to check whether running the next iteration of the same model (without changing model parameters) is acceptable or not. Users should customize the accuracy logic condition and what action to take when it evaluates ***TRUE*** or ***FALSE***. Action can be the execution of another workflow, bash script, python script, or MySQL query. See Figure 3.15 Accuracy Checking Unit Structure.

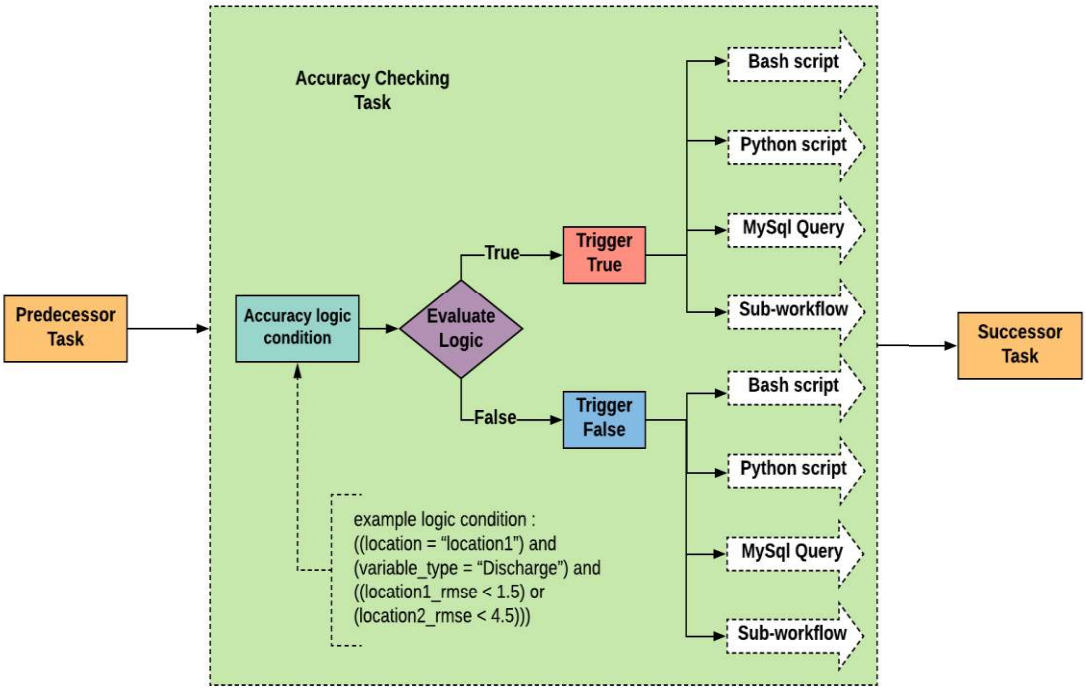


Figure 3.3 The Accuracy checking unit structure

#### 3.2.2.1 WRF rainfall forecast accuracy

WRF is the primary rainfall forecast source for descending weather models (HEC-HMS and FLO-2D) accuracy of WRF model results is crucial for the accuracy of the descending models. The WRF model has multiple results for a particular day due to different input GFS data hours and model parameter variations.

To generate the required forecast rain time series for later models, we should select the highest accurate WRF model result from the available results. We should compare the rainfall forecast generated with the rain gauges' observed rainfall to select the best-performing model.

A WRF run generates three days (72-hours) time series forecast (but changing configuration and execution parameters can be changed).

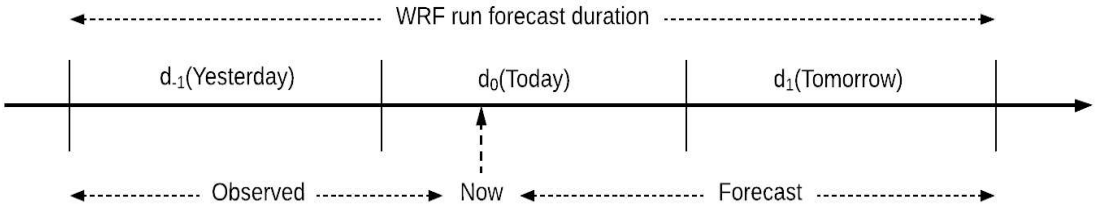


Figure 3.4 WRF forecast time series

As shown in Figure 3.4, part of the forecast time series overlaps with the observed time series. Using this overlapping region, we can calculate the accuracy levels of the model results. Figure 3.5 shows cumulative rainfall plots for WRF model forecasts with observed rainfall for required locations and calculate Root Mean Squared Error (RMSE) for each graph. WRF forecast with the slightest error percentage will be the best fitting graph, and that forecast will apply as the descendent models' rainfall inputs.

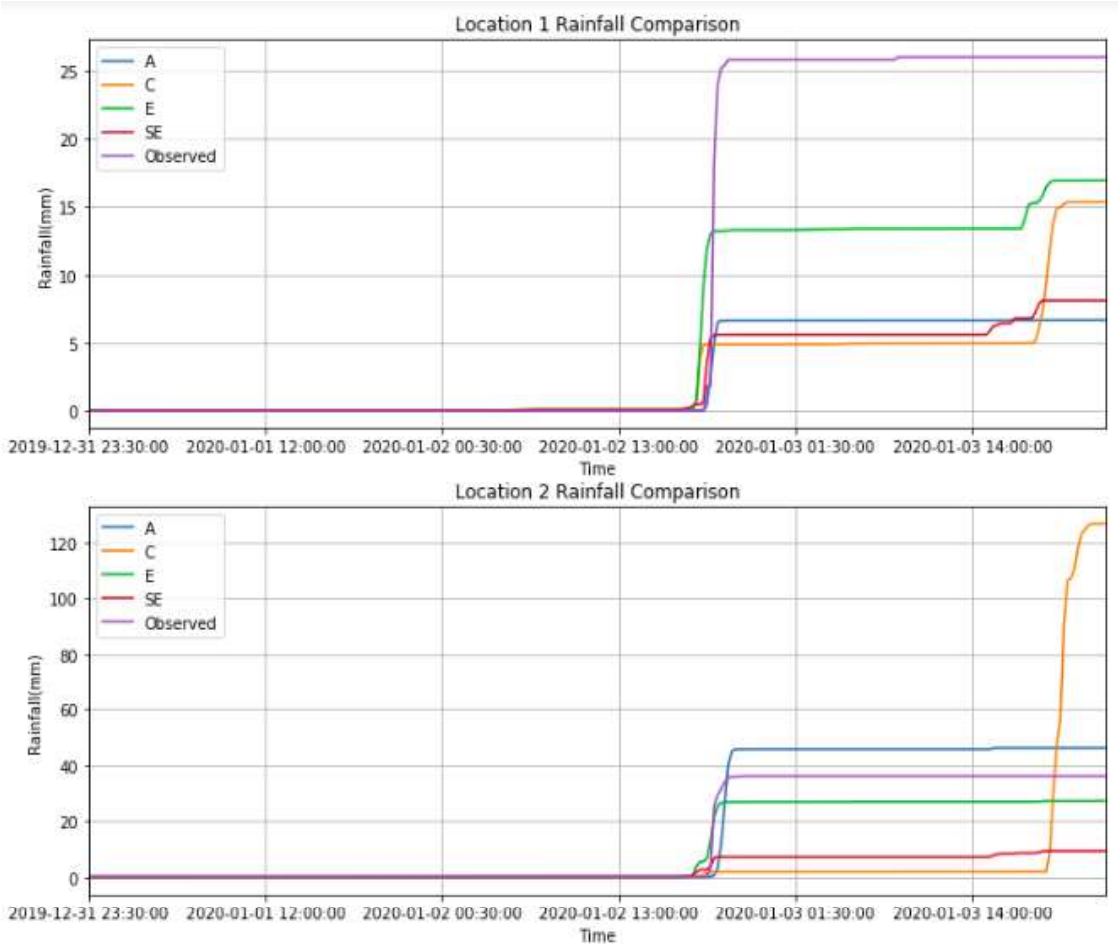


Figure 3.5 WRF forecast comparison

#### 3.2.2.2 HEC-HMS water discharge forecast accuracy

HEC-HMS has a significant impact on the water level forecast of the FLO-2D model because it provides forecast time series for the inlet boundary condition (INFLOW) of the FLO-2D model. We only extract time series from the model output in a single point (for FLO-2D grid inlet). The HEC-HMS model is also designed in a way that forecast duration has an overlap with the observed. Figure 3.6 shows both forecast and observed time series comparison, due to this behavior we cannot use RMSE calculation for accuracy measuring.

HEC-HMS model is a lumped model which does not have any boundary conditions except initial conditions. CUrW uses a model which is developed for the Kelani Basin area. Because of the sea, Kelani river water levels vary with the tidal effect. Since the HEC-HMS model ignored these boundary conditions, its forecast does not show the tidal effect's impact. However, when there is a significant amount of rainfall, tidal conditions have an insignificant effect on discharge amounts.

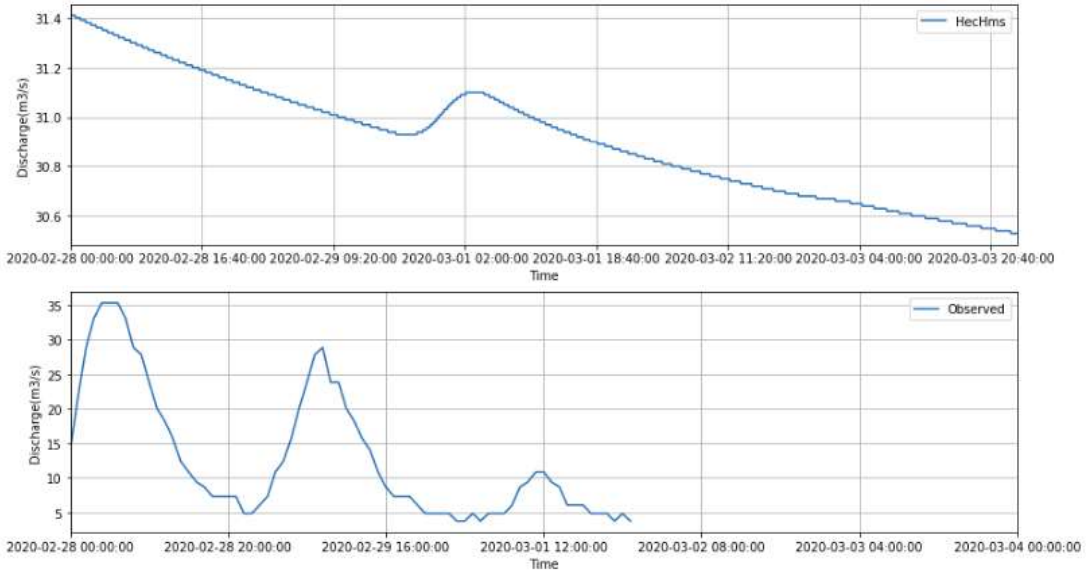


Figure 3.6 HEC-HMS forecast comparison

We can make a statistical forecast (SF) for future discharge values using observed discharges (calculated from water level gauge value using H-Q curve), we can make a statistical forecast (SF) for future discharge values. See Figure 3.7 for statistical forecast of discharge. This discharge forecast will decide based on Equation 1 in the accuracy checking unit. If the condition in Equation 1 is *TRUE*, INFLOW.DAT will be generated from SF data; otherwise, it will be generated from HEC-HMS output. For that we use the comparison graph shown in Figure 3.8.

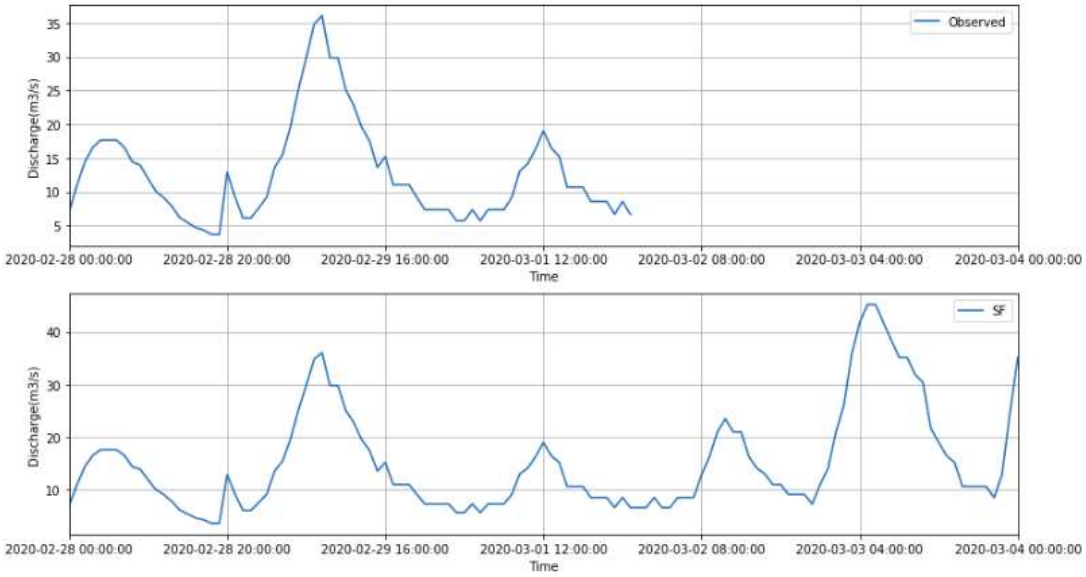


Figure 3.7 Statistical forecast of discharge using the observed discharge

$$\frac{Q_f}{Q_{sf}} \geq \textit{Configurable value } (e.g.,1.5) \tag{3.1}$$

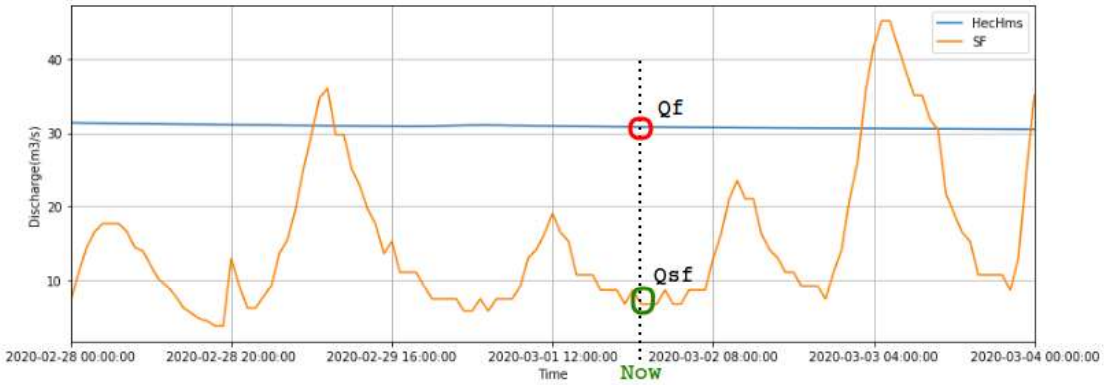


Figure 3.8 SF discharge vs HEC-HMS discharge

### 3.2.3 Infrequent event handling unit

Dam breakage and pump/gate not operating are the main unexpected conditions we can face in the system operation. Unexpected event handling units will identify these events and trigger required one or multiple FLO-2D models to generate water level forecasts.

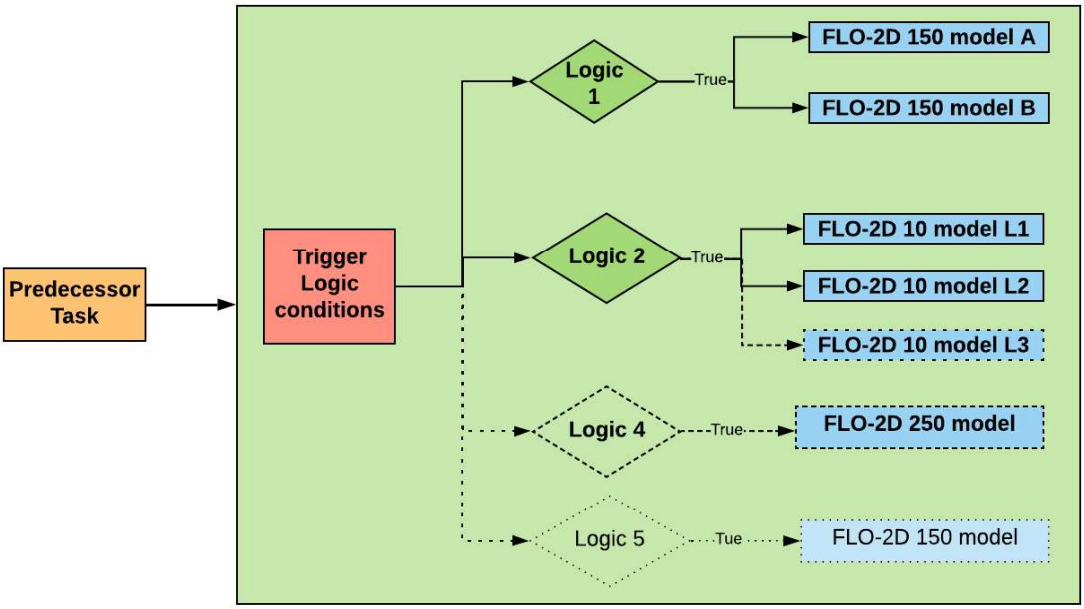


Figure 3.9 Infrequent Event Handling Unit Structure

While adding this unit to the workflow user should customize relevant triggering logic conditions for each FLO-2D model(s) as illustrated in Figure 3.9. The unit will evaluate all the logics listed and trigger relevant models to predict the water levels when an unexpected event occurs.

### 3.2.4 Pump operating unit

Channel network, water-storing area, and various other geographic conditions are represented in the FLO-2D model templates. Therefore, any change in these conditions should be reflected in the model template to make accurate water level forecasts. Moreover, when adding a new pump system or gate to the basin, that should be integrated into the model template.

Figure 3.10 (b) shows three main pumping stations in the Kelani Basin area operated by CUrW; each pumping station consists of five pumps. A pumping station can operate in three modes,

- Full – all the five pumps are in operation
- Medium – three pumps are in operation
- Low – only one pump in operation

Furthermore, CUrW can operate one more pumping station in any of the above modes according to the requirement. Three gate structures are employed for operation along with pumping stations, as shown in Figure 3.22 (a).

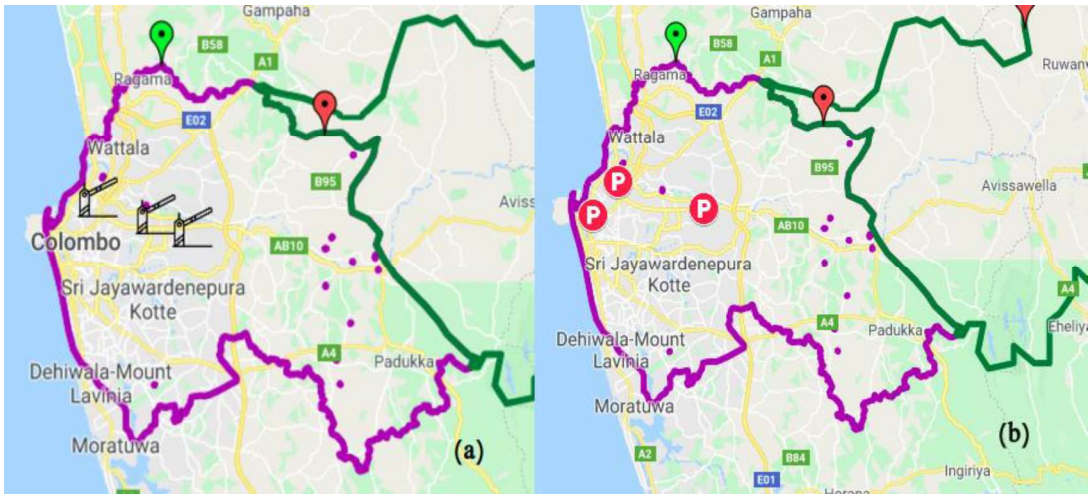


Figure 3.10 Gate locations (a) and Pump locations (b).

To forecast water levels in each state, domain experts have built a set of model templates including each of the above pump operating combinations and gate operating conditions. Figure 3.11 illustrates the pump operating unit with the same structure as an unexpected event handling unit except for the logic evaluation method. Both units have to deal with different model templates to handle each event.

The pump operations unit has inco rporated FLO-2D model templates to cover all the required pump and gate combinations. If defined logic conditions, evaluated as TRUE FLO-2D 150m resolution model(s) will be triggered with the relevant model template(s). That will forecast water levels if the pump and gates are operated as in the model template. If a defined logic condition is evaluated as FALSE, the unit will evaluate the next following logic, and so on. Users can add any number of logic conditions and model template combinations to the unit. Other than the FLO-2D 150m resolution model, users can also use other FLO-2D models or any other required weather model. In CUrW, we have used FLO-2D 150m for the pump operations.

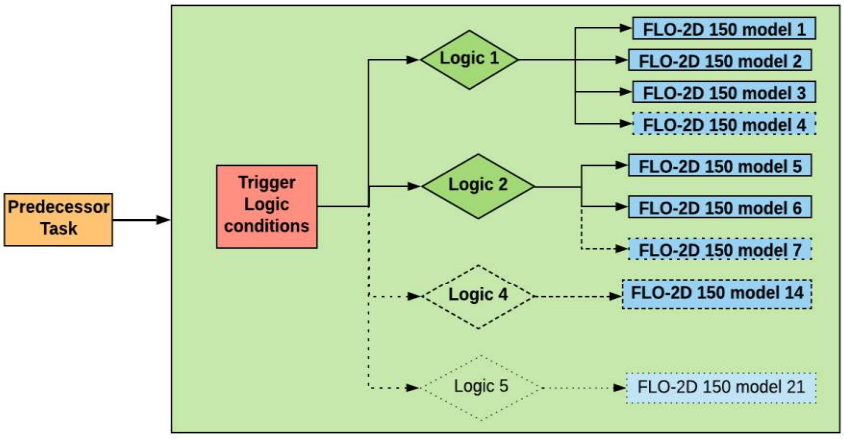


Figure 3.11 Pump operating unit

### 3.3 Rule Engine

The majority of the existing DSS implementations use a Rule Engine to enforce users' control on the decision making at different levels [9] [7] [17]. Here we use the Rule Engine as the primary controller of the system's decision-making, consisting of rule variables and rule logics. See the structure of the Rule Engine shown in Figure 3.12. With the Rule Engine, we can control the weather models'

- execution – by creating rules
- execution order – by setting a priority
- execution flow – by integrating accuracy unit, pump controlling unit and infrequent event handling unit.

Also, the Rule Engine,

- Simplify the complexity of dynamic workflows – Figure 3.1 illustrated a dynamic workflow is a collection of multiple components. Integrating many components together makes it harder to understand the flow. Using rules, we have structured the workflow creation
- Improve the usability – as users can easily identify the basic workflow and decision components separately, they can build complex workflows without invalidating the execution order (WRF, HECHMS and FLO2D)
- Convenient for providing feedback – as Rule Engine structures dynamic workflows, it is easier to add decision-making units to them.

When creating a dynamic workflow, each task (model execution or decision-making) is created as a rule and set a priority order for those tasks in the Rule Engine. Moreover, users can integrate a logic condition to each rule (task) to improve the rule's flexibility. Based on the success or failed trigger of the logic condition, users can build this logic using rule variables and basic logic operators.

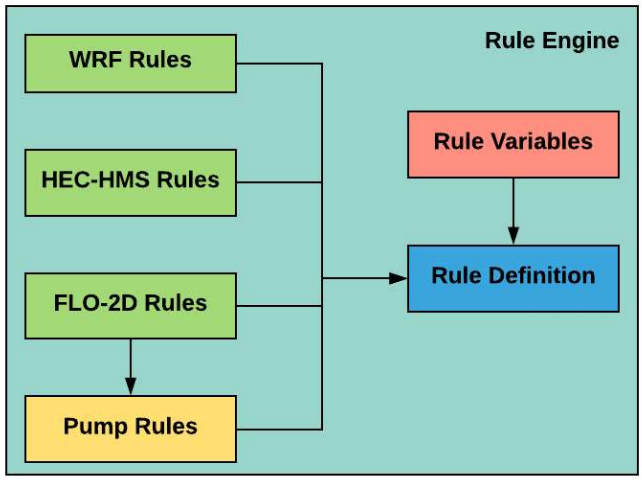


Figure 3.12 Rule engine structure

### 3.3.1 Rule types

This rule engine has multiple rule types for different purposes. One set of rules define the execution of the weather models, while another set defines logical decisions related to the purpose of model execution:

1. WRF Rules - Contains WRF run configurations
    - Namelist wps file parameters
    - Namelist input file parameters
    - WRF version
    - GFS data hour - 00/06/12/18
    - WRF run - d0/d1/….
    - Gfs data availability needed – ***TRUE/FALSE***
    - On which node wrf should run
2. HEC-HMS Rules - Contains HEC-HMS run configurations
    - Forecast days and observe days
    - Rainfall data retrieval method
    - Node Hec-Hms should run
    - Target model - single/distributed
3. FLO-2D Rules - Contains FLO-2D run configurations
    - Target model - 250m/150m
    - Number of forecast/observed days
    - Raincell data from - MME
    - Inflow data from - MME/SF
    - Outflow data from - TSF/MGF
    - On which node flo2d is running
4. Pump Rules - Based on logical condition build using rule variables, if evaluated to ***TRUE*** execute the relevant model and has no ***FALSE*** condition
5. Accuracy Rules - Based on logical conditions build using rule variables, has both ***TRUE*** and ***FALSE*** conditions. One trigger for ***TRUE*** condition and another trigger for ***FALSE*** condition.

### 3.3.2 Rule variables

Rule variables are the main component of building rule logic. They can be defined and maintained by the user's requirements. Each variable has its update cycle (every 5/10/30/60 minute or daily) defined as separate workflows. Variable scheduler (workflow, an Airflow dag) monitors each variable's schedule and triggers the update workflow according to that schedule, except the static variable values. See variable scheduler in Figure 3.13 and variable update scheduling in Figure 3.14.

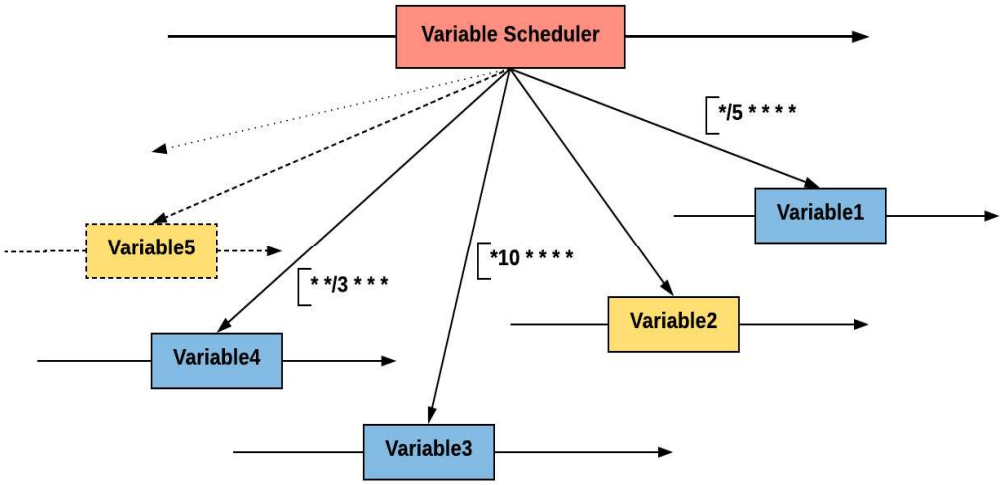


Figure 3.13 Variable scheduler

| id | variable_name | variable_type | dag_name | status | schedule | last_trigger_date |
|---|---|---|---|---|---|---|
| 1 | current_rainfall | Precipitation | current_rainfall_dag | 3 | */10 * * * * | 2020-03-17 12:20:32 |
| 2 | current_water_level | WaterLevel | current_water_level_dag | 2 | */10 * * * * | 2020-01-28 06:50:38 |
| 3 | water_level_rising_rate | WaterLevel | water_level_rising_rate_dag | 3 | */20 * * * * | 2020-03-17 12:20:32 |
| 4 | last_1_day_rainfall | Precipitation | last_1_day_rain_dag | 3 | 0 * * * * | 2020-03-17 12:00:37 |
| 5 | last_2_day_rainfall | Precipitation | last_2_day_rain_dag | 3 | 0 */6 * * * | 2020-03-17 12:00:37 |
| 6 | last_3_day_rainfall | Precipitation | last_3_day_rain_dag | 3 | 0 */6 * * * | 2020-03-17 12:00:37 |
| 7 | rainfall_intensity | Precipitation | rainfall_intensity_dag | 3 | */10 * * * * | 2020-03-17 12:20:32 |

Figure 3.14 Variable update scheduling

Due to different user requirements, multiple types of variables are defined in the system as reflected in Figure 3.15. Furthermore, users can define new variables to the system as they need. The calculation method and update frequency will depend on the type of the variable.

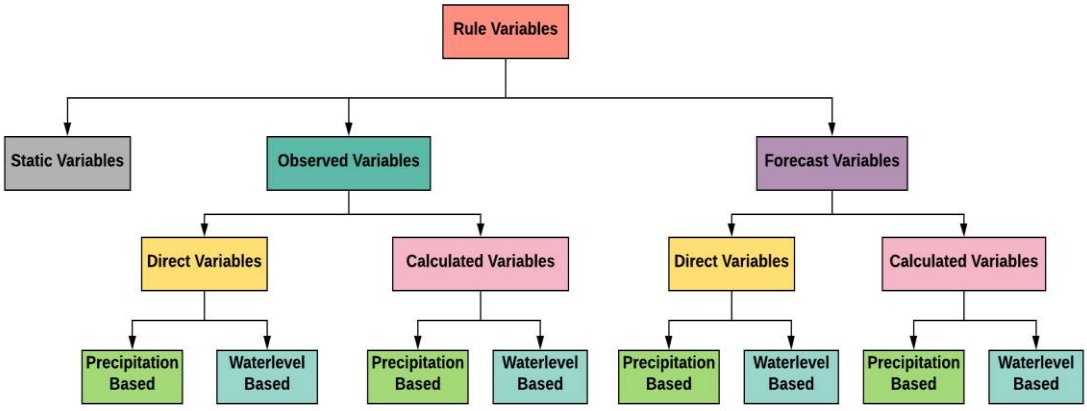


Figure 3.15 Variable types

### 3.3.3 Rule logic

Users can build their required any kind of logic using rule variables and the set of operators, such as *and*, **or**, **<**, **≤**, **>**, **≥**, **!=**, **=**, **==**. In building rule logic, putting correct parentheses is crucial to avoid confusion in evaluating the logic condition.

Also, adding location and variable_type is essential; otherwise, the logic condition has no meaning. Consider following three sample logics,

$$[((location = 'Location\ 1')\ and\ (variable\_type = WaterLevel'))$$

$$and\ ((current\_water\_level > alert\_level)\ or\ (current\_water\_level > warning\_level))] \quad (3.2)$$

$$[((location = 'Location\ 2')\ and\ (variable\_type = 'Precipitation'))$$

$$and\ ((rain\_fall\_intensity > 60)\ or$$

$$((last\_1\_day\_rainfall > 160)\ and\ (last\_3\_day\_rainfall > 320)))] \quad (3.3)$$

$$[((location = 'Location\ 3')\ and\ (variable\_type = 'WaterLevel'))$$

$$and\ ((rain\_fall\_intensity > 80)\ or$$

$$(last\_3\_day\_rainfall\ /\ last\_3\_day\_rainfall > 2))] \quad (3.4)$$

Users can use '=' or '==' to show the equality, if the user added '=' it will be converted to '==' before the evaluation. Logic evaluation is done through MySql where clause or implemented rule evaluator.

### 3.4 Dynamic workflow creation

Using any number of dynamic workflow components in Section 3.2 in any order, users can create dynamic workflows. Other than the mentioned workflow components, users can use any bash script, python script, or MySql query as a workflow task as listed in Table 3.1.

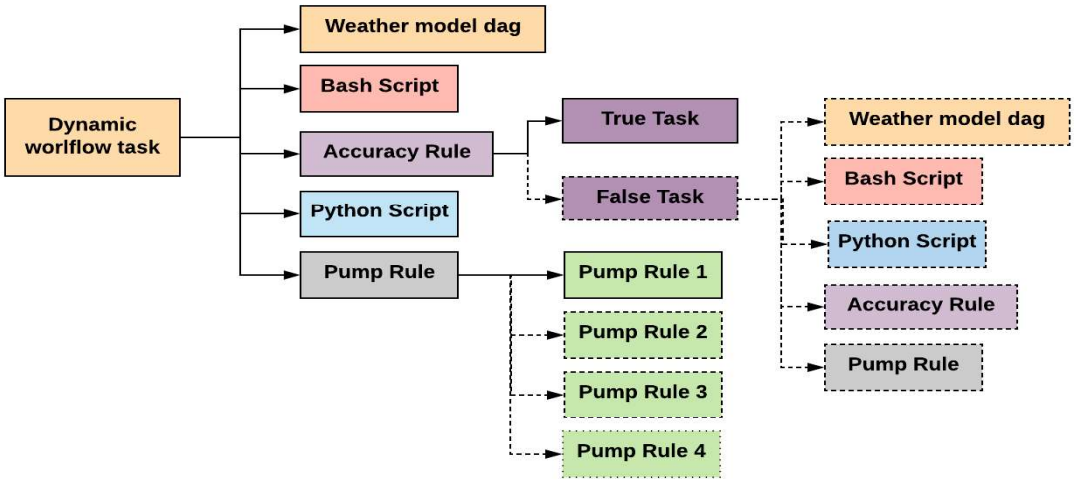


Figure 3.16 Dynamic task types and their arrangement

Table 3.1 Components for dynamic workflow creation

| Field | Purpose |
|---|---|
| task_name | The name of the task should add a unique name within the workflow it belongs to. No spaces and special characters are allowed. Multiple words can connect using the '_' character. |
| task_type | • 1 - Bash script<br>• 2 - Airflow dag<br>• 3 - Python script<br>• 4 - MySql query |
| task_location | • external – task is located in a different server in the same network<br>• internal – task is located in the same location where airflow webserver is hosted<br>• gcloud – the task is located in the Google Cloud server<br>• airflow – task is airflow dag |
| task_content | If the task is,<br>• Bash script – Bash script with the exact path<br>• Python script – Python script with the exact path<br>• MySql query - query<br>• Weather model/Pump rule/Accuracy rule - any identification for the weather model/Pump rule/Accuracy rule |
| input_params | If the task is Weather model/Pump rule/Accuracy rule, this should contain at least the following two params in JSON - {"model_type": "weather model", "rule_id": ruleId} .Users can add "run_date" to specify the model execution date. Also, can add any other params as well for bash scripts/python scripts and MySql queries. |
| time_out | The maximum allowed time limit for the task execution should be as following JSON, {"hours":1,"minutes":30,"seconds":0} |
| task_order | User can specify the task execution order |
| active | 0 - disabled task, will not execute in the workflow execution<br>1 - active task execute in the workflow execution<br>2 - running task<br>3 - task execution has completed<br>4 - error in task execution<br>5 - task has stopped by the admin |

The dynamic workflow task can follow any combination as shown in Figure 3.16. When executing a particular task, it will select the execution path as the user has configured it. However, when it meets an accuracy checking task or pump rule task, it will flow through the resulting path after evaluating each task's logic conditions.

## 3.5 Dynamic workflow execution

After organizing the workflow tasks in a particular order, users can schedule it to run on a particular date or for a scheduled time as mentioned in Table 3.2. The workflow scheduler triggers the workflow execution process, running in a configured time interval (e.g., every five minutes). If multiple dynamic workflows are scheduled simultaneously, they will be triggered concurrently.

Table 3.2 Dynamic workflow scheduling components

| Field | Purpose |
|---|---|
| dag_name | Should be unique, no spaces and special characters allowed. Multiple words can connect using the '_' character. |
| schedule | Syntax of cron scheduling will be applied,<br>* * * * *<br>- - - - -<br>\| \| \| \| \|<br>\| \| \| \| ----- Day of week (0 - 7) (Sunday=0 or 7)<br>\| \| \| ------- Month (1 - 12)<br>\| \| --------- Day of month (1 - 31)<br>\| ----------- Hour (0 - 23)<br>------------ Minute (0 - 59) |
| timeout | Should add timeout for the workflow execution to avoid any task of the workflow hang on completion of that task Maximum allowed time limit for the workflow execution, should be as following JSON,<br>{"hours":2,"minutes":30,"seconds":0} |
| status | 0 - disabled workflow, won't execute<br>1 - enabled for workflow execution<br>2 - running workflow<br>3 - execution has completed<br>4 - error in workflow execution<br>5 - workflow has stopped |

### 3.5.1 Workflow scheduler

The workflow scheduler shown in Figure 3.17 is responsible for triggering every eligible dynamic workflow. It is an Airflow dag as a variable scheduler and runs continuously in a configured interval. The difference between these two schedulers is that the workflow scheduler must have a schedule. Without having a schedule, it cannot find out the execution time of that workflow.

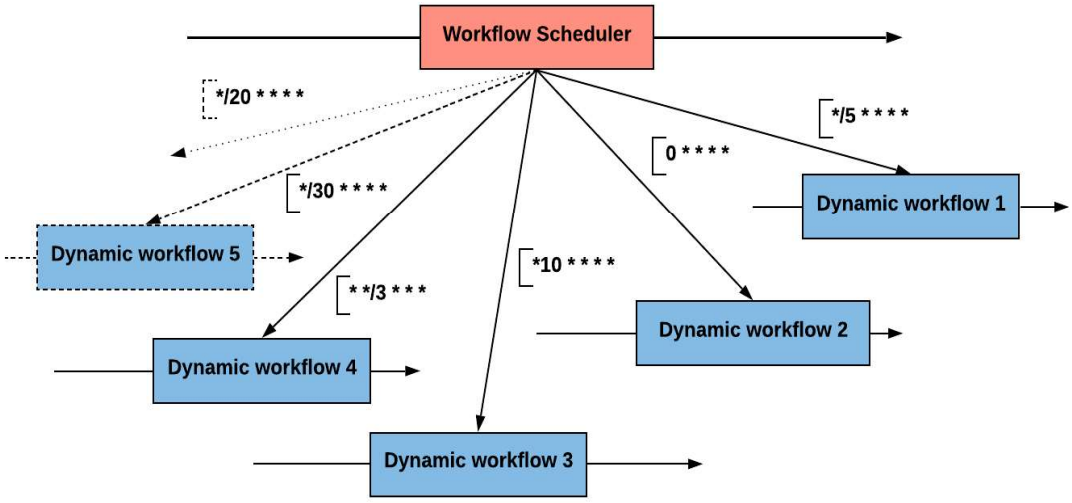


Figure 3.17 Workflow Scheduler

In each iteration,

1. Scheduler picks all the eligible (workflows which are not in running state) dynamic workflows
2. Calculate the next run time of every workflow by parsing the scheduled string value
3. If the current time is smaller than the next run time of a workflow, that workflow will be triggered
4. Update every triggered workflows' state to running state

### 3.5.2 Schematic view of dynamic workflow

The schematic view shown in Figure 3.18 illustrates how dynamic workflow tasks, dynamic workflows, and workflow scheduler, are all connected. Workflow scheduler can trigger any number of dynamic workflows to scheduled time or time intervals where each dynamic workflow integrates multiple dynamic workflow components. If we consider "Dynamic Workflow 1", every task can be any of the following:

- Weather model dag
- Bash script
- Python script
- Accuracy Rule
- Pump Rule

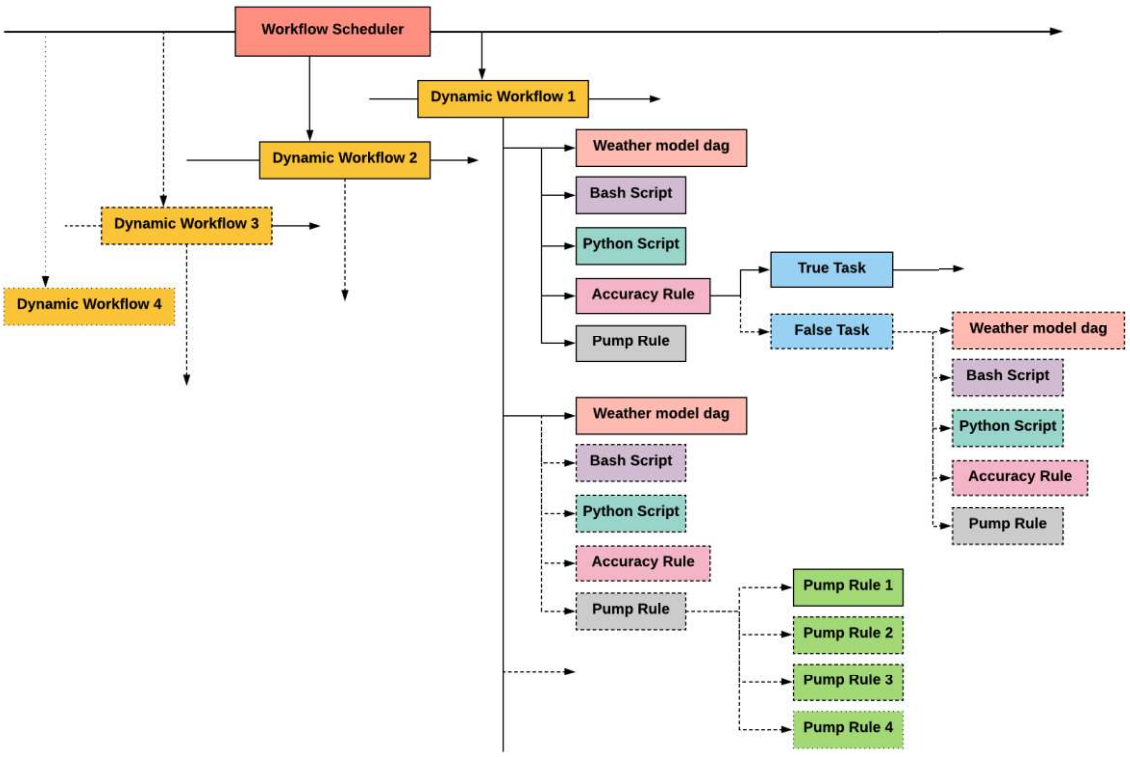


Figure 3.18 Schematic view of dynamic workflow

# 4 PERFORMANCE EVALUATION

The proposed framework is one of the significant components of the CUrW production system. It is developed and deployed in the Google cloud platform as the dynamic workflow management system of CUrW. Section 4.1 presents system implementation. Accuracy level-based decisions are discussed in Section 4.2. Section 4.3 presents unexpected event handling decisions, while pump/gate operating decision is presented in Section 4.4.

## 4.1 Implementation & Integration

Due to the production system's critical nature, the whole system is built on GCP to achieve high availability requirements. Several compute engines (virtual machines) are employed to serve various functions. Figure 4.1 shows how VMs are arranged in the project (This diagram only shows a minimum number of VMs excluding backups and other redundancies) and how this framework integrates with the production system.

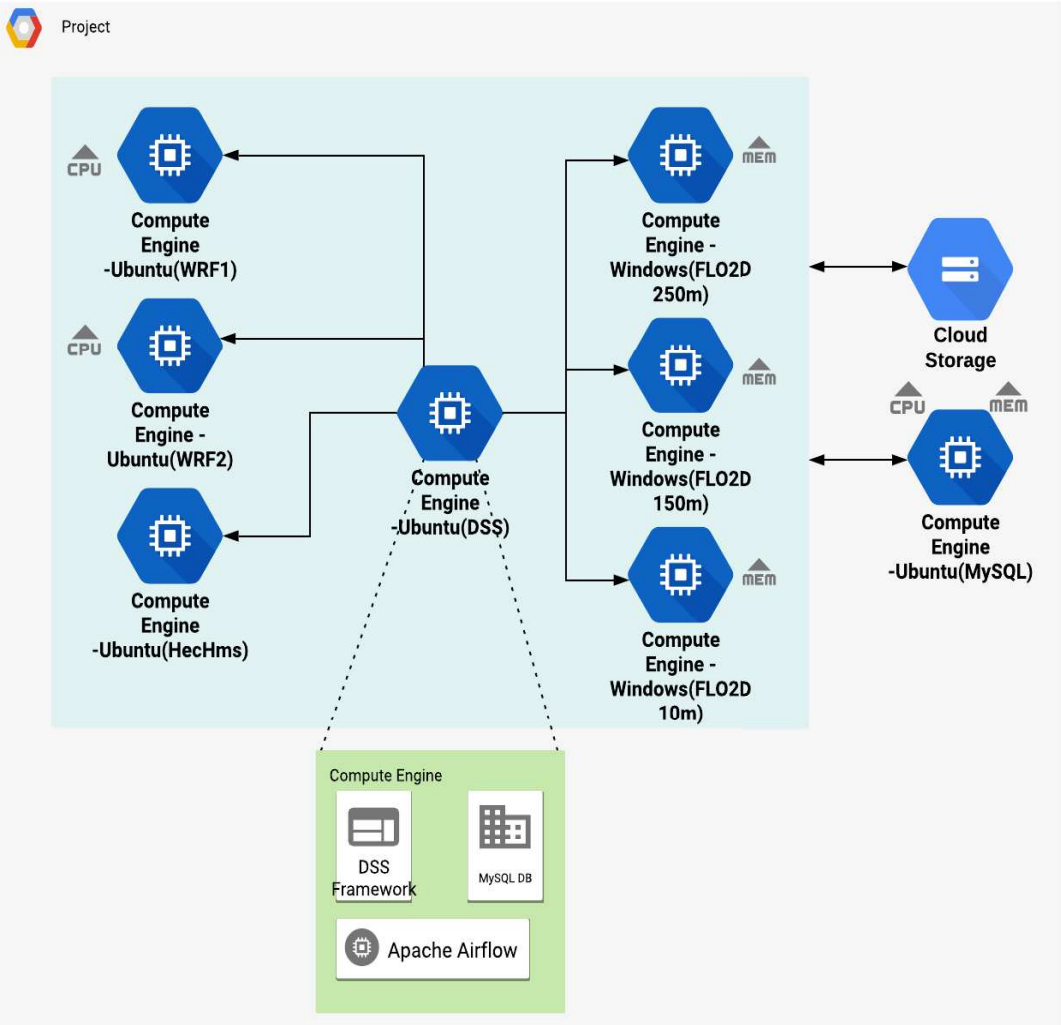


Figure 4.1 Cloud VM allocation

Resource allocation on VMs was done based on their enlisted purposes. Weather models run in a particular VM will need a different configuration than the other. The framework consists of both CPU-heavy (WRF) and memory-heavy (FLO-2D) applications. See Table 4.1 for resource allocation in GCloud VMs'.

Table 4.1 Resource allocation in GCloud VMs

| Compute Engine(s) | Resource Allocation | Purpose |
|---|---|---|
| WRF1 and WRF2 VMs | Intel Broadwell 16vCPU<br>14.4GB RAM<br>OS - Ubuntu 18.04 | WRF model execution. WRF is a numerical model which uses both physics and spatial attributes. |
| HEC-HMS VM | Intel Broadwell 6vCPU<br>12 GB RAM<br>OS - Ubuntu 18.04 | HEC-HMS is light weight model when compared with WRF and FLO2D |
| 3 Windows VMs (FLO-2D 250m, FLO-2D 150m and FLO-2D 10m) | Intel Broadwell 6vCPU<br>22.5GB RAM<br>OS - Windows Server 2016 Datacenter Edition | FLO2D is a Windows platform-based memory-hungry application. |
| DSS VM | Intel Broadwell 4vCPU<br>12GB RAM<br>OS - Ubuntu 18.04 | Serve to Apache Airflow and DSS Framework |
| DB VM | Intel Broadwell 8vCPU<br>30GB RAM<br>OS - Ubuntu 18.04 | MySql instance is running with huge data tables |

The framework's performance analysis can be divided into three main sections: accuracy-based decisions, pump/gate operating decisions, and unexpected event handling decisions. Each decision will update the flow of the model execution dynamically.

### 4.1.1 Model catchment area

A catchment area for a model is a geographical area based on a river or another water source. CUrW has defined its catchment area based on the river, as shown in Figure 4.2.

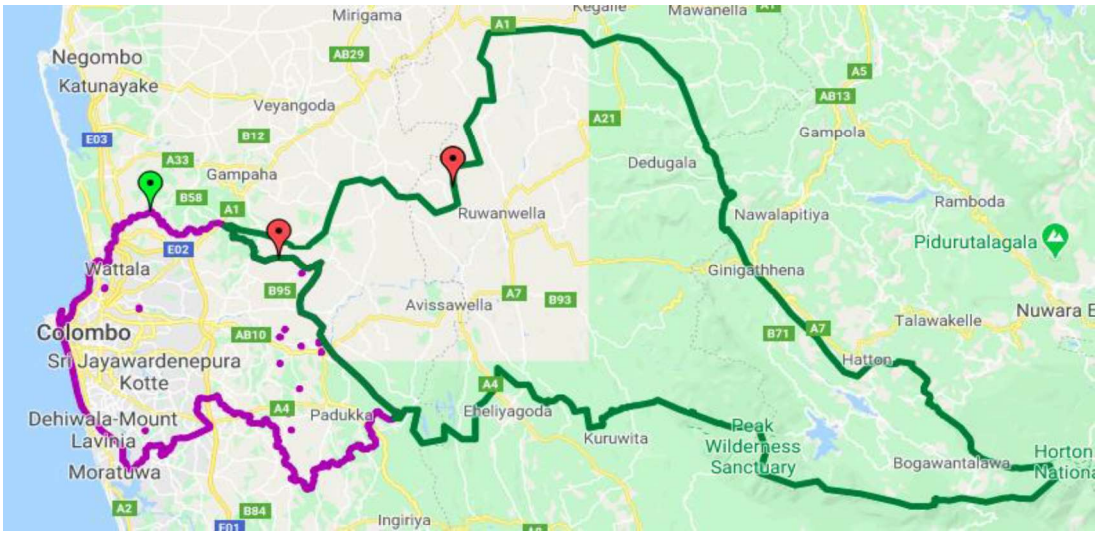


Figure 4.2 Catchment area (purple-lower catchment, green-upper catchment).

### 4.1.2 Observed data sources

Observed data (precipitation and water level) are used as input to the models and for accuracy measuring the model results. CUrW has installed both rain gauges shown in Figure 4.3 and water level gauges shown in Figure 4.4 throughout the catchment area.

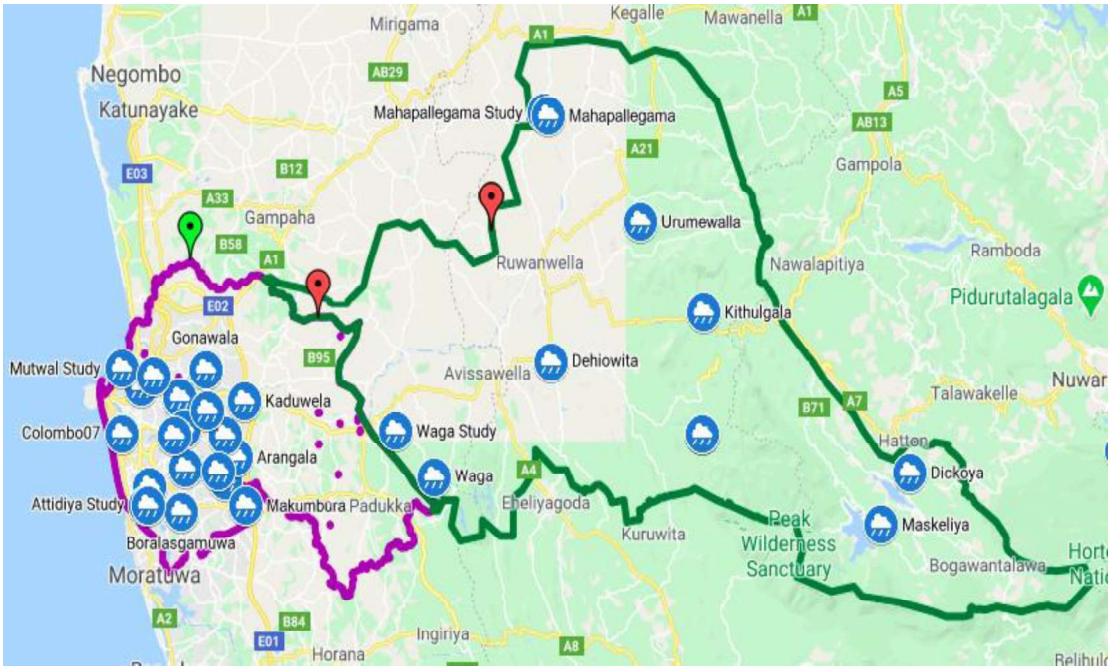


Figure 4.3 Rain gauges' locations

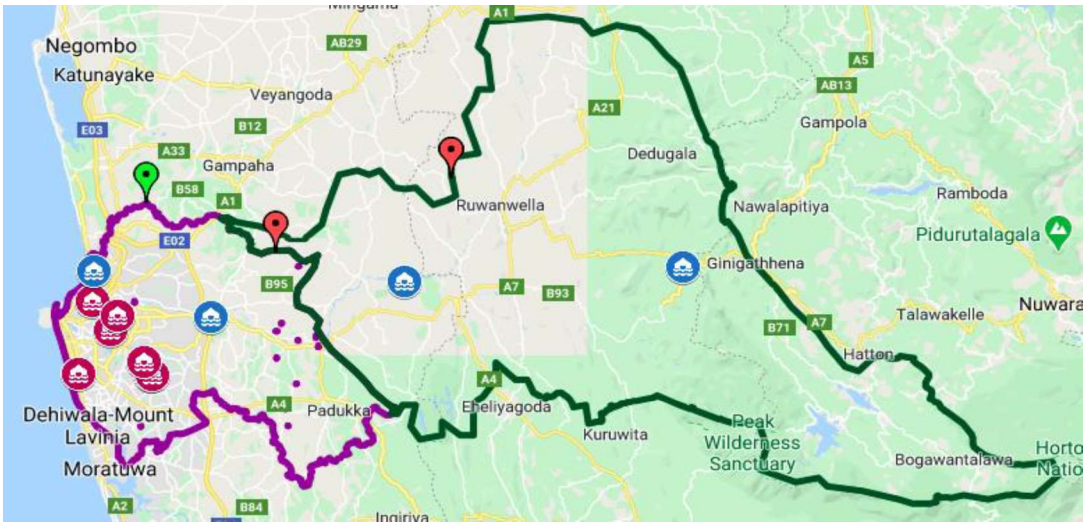


Figure 4.4 Water level gauge locations

## 4.2 Accuracy-level-based decisions

The primary objective is selecting the highest accurate model when running multiple instances of the same weather model.

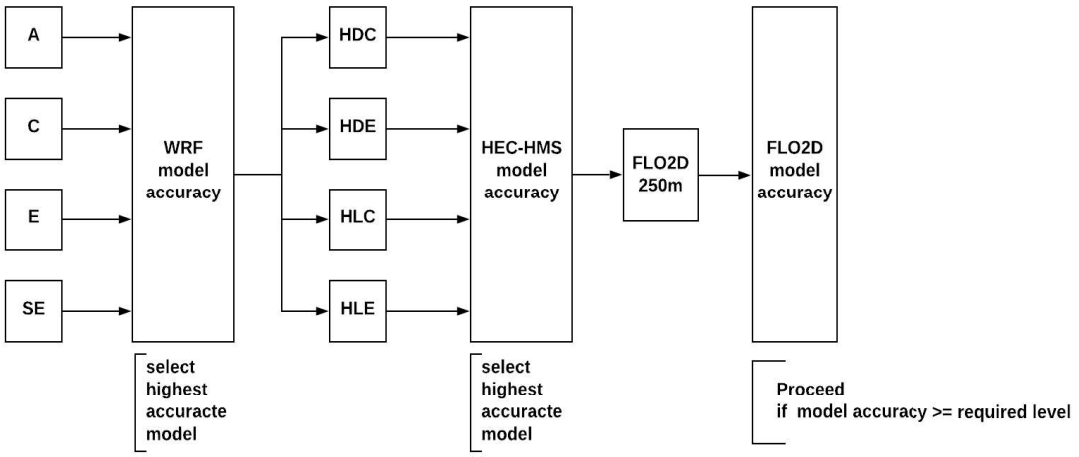


Figure 4.5 Flow of accuracy checking

### 4.2.1 Weather model accuracy

Every model's output time series start from a past date-time and ends in a future date-time. Figure 4.6 shows the model's output time series starting from yesterday at 00:00h and ends with tomorrow at 23:59h. Therefore, there is an overlapped time series between model output and observed time series. Using that overlapped time series, we can calculate the model's accuracy.

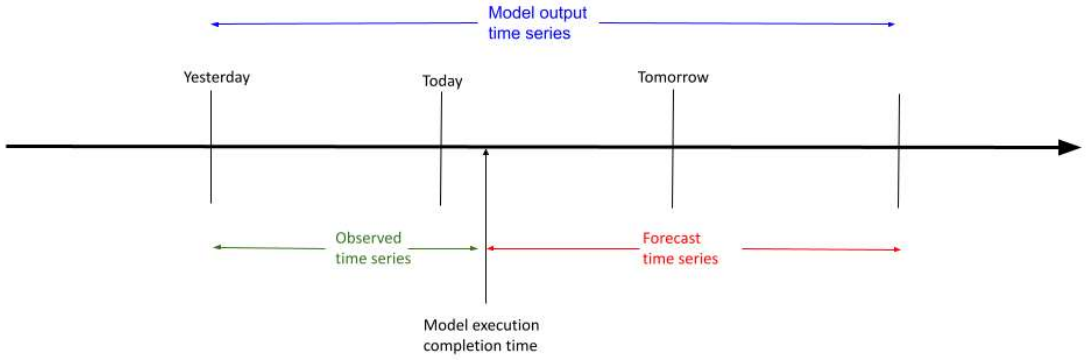


Figure 4.6 Model output time series for accuracy calculation

### 4.2.2 WRF model accuracy

Field experts can build new WRF models by optimizing various physics parameters in the model configuration files. At CUrW, we are running four WRF models parallel for forecasting precipitation optimized for different weather conditions (or weather types).

- Model A – Anticyclonic
- Model C – Cyclonic
- Model E – East
- Model SE – South East

Each WRF model output time series can be divided into observed and forecast, as shown in Figure 4.7. We are running a model on D0; we can use the observed part (D-1 time series) of the output time series for accuracy testing with actual observed data.

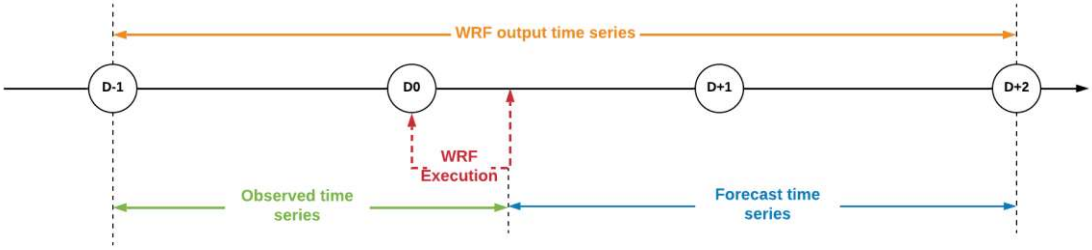


Figure 4.7 WRF time series

WRF runs take around 7-hours to complete the execution, so the total observed time series length will be 31-hours. Accuracy based model selection decision on WRF model is as follows:

1. Calculating RMSE between actual observed data and model output observed data of all the executed WRF models.
1. Select the WRF model which has a minimum RMSE value shown in Figure 4.8.
2. Use that WRF model to later model for their input creation.

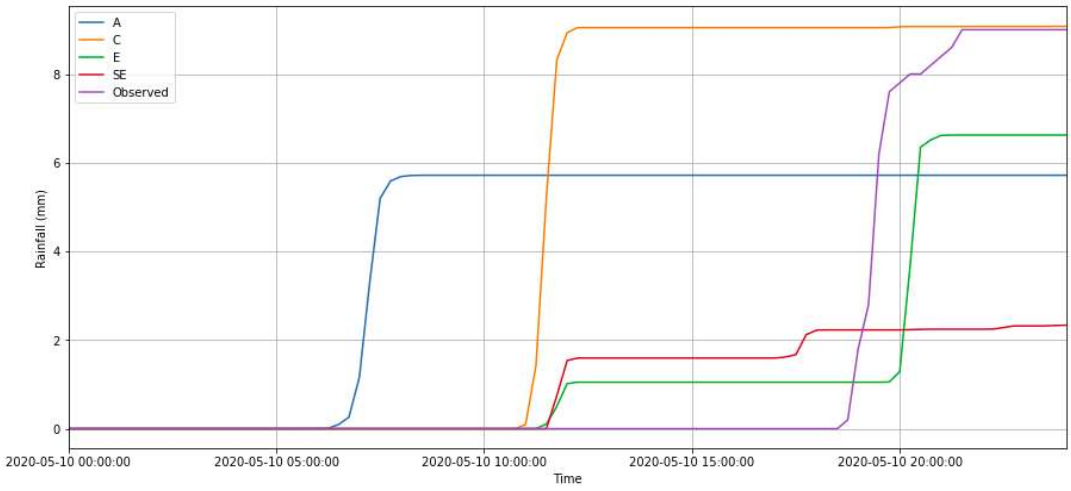


Figure 4.8 WRF model result comparison with observed data

3. Using the same method, we calculate decision accuracy by calculating the RMSE between selected model forecast data with the observed data shown in Figure 4.9.

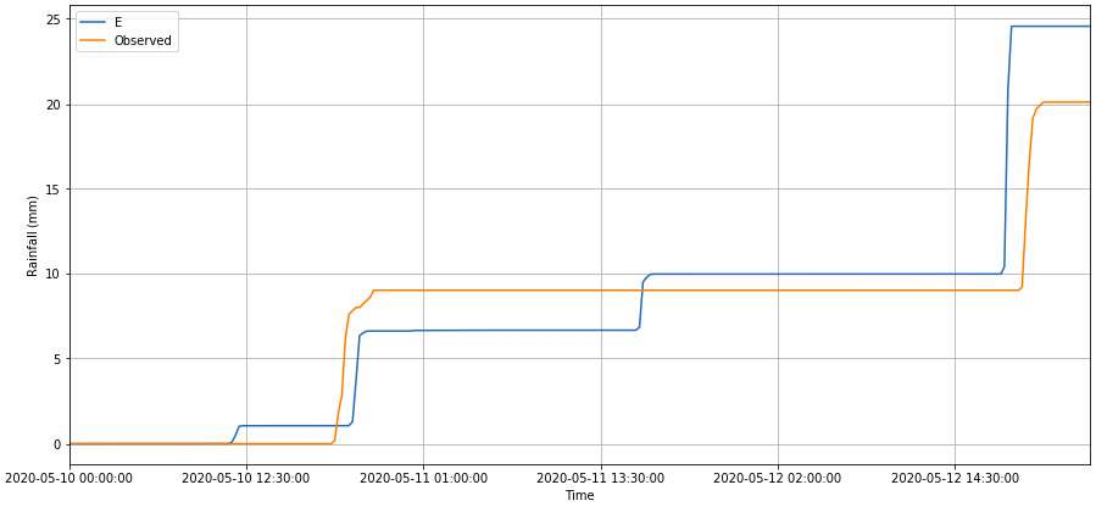


Figure 4.9 Selected WRF model's accuracy

### 4.2.3 HEC-HMS model accuracy

Field experts have built the following four different HEC-HMS models with different physics parameters and catchment definitions:

- HDC - HEC-HMS distributed continuous model
- HDE - HEC-HMS distributed event model
- HLC - HEC-HMS lump continuous model
- HLE - HEC-HMS lump event model

Distributed models' catchment area is divided into a few sub-catchments using geographical conditions in the catchment area. Alternatively, the whole catchment area is a single unit for lump models.

Therefore, distributed models' input is a collection of mean rain in each sub-catchment, and lump models' input is single mean rain for the whole catchment area. There can be discrepancies between each model's accuracy with the input rainfall distribution in the catchment area. Therefore, in CUrW, we run all four models and pick the most accurate model for the descendants' input preparation.

As in the WRF model, HEC-HMS also needs time to stabilize the system. So, it also has observed data time series in its output, as illustrated in Figure 4.10. Using this observed time series of the output, we can determine which model performs well in the existing conditions. Hence following are the steps in the HEC-HMS accuracy decision:

1. Calculating RMSE between actual observed data and model output observed data of all the executed HEC-HMS models.
2. Select the HEC-HMS model, which has a minimum RMSE value that shown in Figure 4.11.
3. Use that HEC-HMS model to descendant model (FLO-2D) for its input creation.

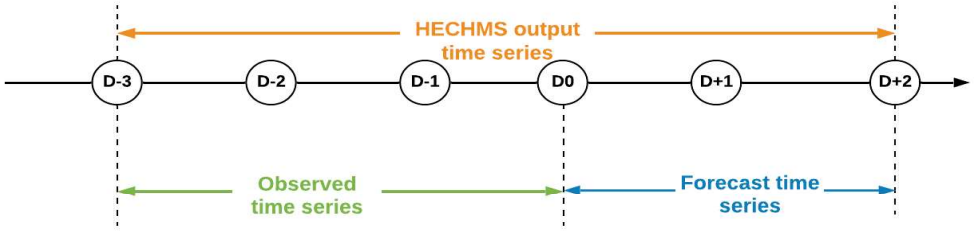


Figure 4.10 HEC-HMS time series

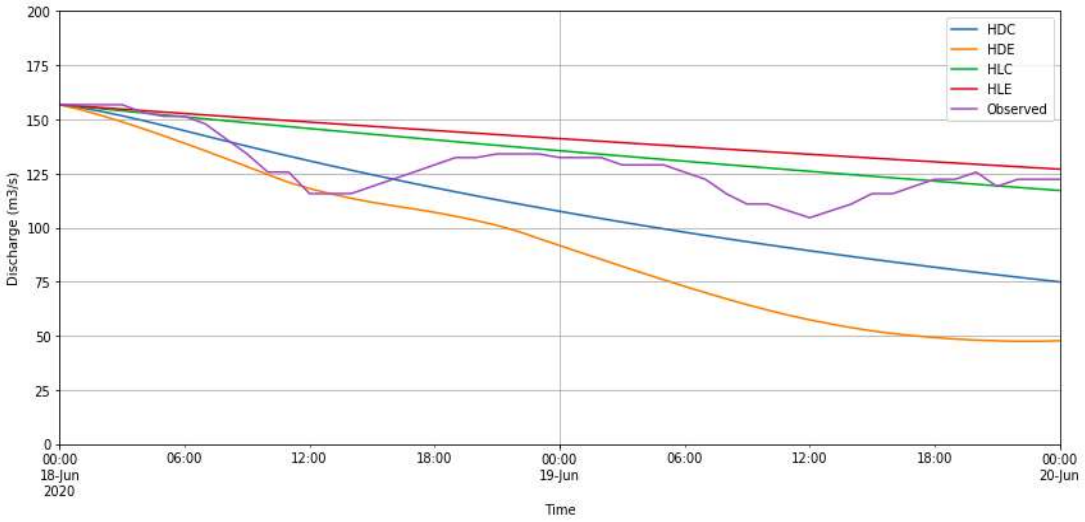


Figure 4.11 HEC-HMS model output comparison with observed data

We can calculate decision accuracy using the same method by calculating the RMSE between selected model forecast data with the observed data. In Figure 4.12, we can see near-constant fluctuation in the discharge because of the sea tide effect on the river water level. HEC-HMS models do not pick this effect as it has a more negligible effect than river discharge. Also, this is insignificant when there is rain.

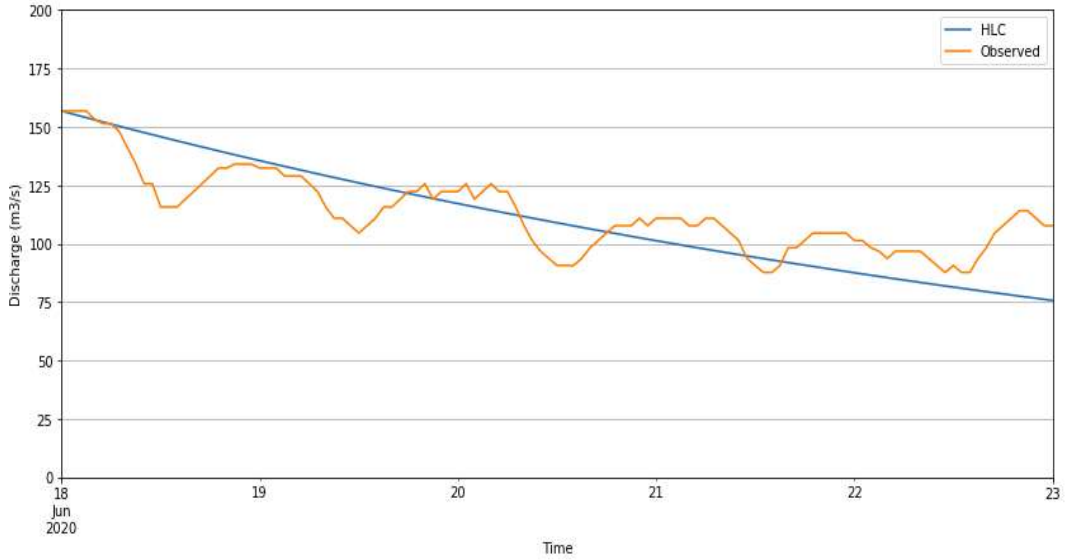


Figure 4.12 Selected HEC-HMS model's accuracy

### 4.2.4 FLO-2D model accuracy

FLO-2D model is the last of the model flow, which provides the water level forecast needed for decision-making on pump operations and loss estimation. FLO-2D models that use in the CUrW have fallen into two main categories. They are resolution-based (catchment area) and the purpose of the model run.

Resolution based models:

- 250m - Catchment area is divided into 250m grids
- 150m - Catchment area is divided into 150m grids
- 10m - Catchment area is divided into 10m grids

In Figure 4.13, see the defined catchment areas in FLO-2D.


Figure 4.13 FLO-2D 250m and 150m catchment areas

Execution purpose-based models:

1. Initial decision making on pump model triggering (250m)
2. For simulating each pump configuration (150m)
3. Flood map generation (150m)
4. For loss estimation (10m)

Though there are several models, initial decision-making is done based on the results of the 250m model. Henceforth FLO-2D model accuracy means the accuracy of the 250m resolution model. When extracting model results, it included various location data covering critical locations for decision making like canal locations, pump/gate locations, and places identified as high impact areas.

1. For accuracy measuring, we use a set of selected locations:
2. Extract output time series of selected locations
3. For the graphs shown in Figure 4.14 calculate RMSE values of all selected locations with their respective observations.
4. The maximum allowed error limit might vary with the properties of the selected location.
5. Calculate the percentage of locations with an error rate below the maximum allowed error as the model's accuracy.

### 4.2.5 Results

Measuring the accuracy of the predecessor models is essential to the accuracy of the descending models in the workflows. If one of the models in the middle of the workflow has less accurate results, using those results to the descendent models will be more error-prone. Making decisions based on erroneous data can be catastrophic in disaster management. Weather model execution time varies with the type of weather model, physics parameters configurations, and catchment area definitions are listed in Table 4.2.

Table 4.2 Model execution times

| **Weather Model** | **Execution time** (approximately with input preparation time and output extraction time) |
|---|---|
| WRF (A, C, E, SE) | 6 - 8 hours |
| HECHMS (HDC, HDE, HLC, HLE) | 10 - 15 minutes |
| FLO2D 250m | 40 - 60 minutes |
| FLO2D 150m | 3.5 - 4.5 hours |
| FLO2D 10m | 20 - 30 minutes |

When considering model execution time and resource consumption, discarding produced results based on inaccurate model predecessor model results will be a massive waste of time and resources.

Also, the rerun of the workflows will add huge latency to the decision-making process. This developed framework can select the most accurate models and move on to decision-making processes to avoid resource wastage and latency. Even though the resource utilization advantage, accuracy-based decisions themselves, can add latency to the workflows. We have gathered resource consumption and latency of accuracy-based decisions on all three weather model types over six months, 4-runs per month.

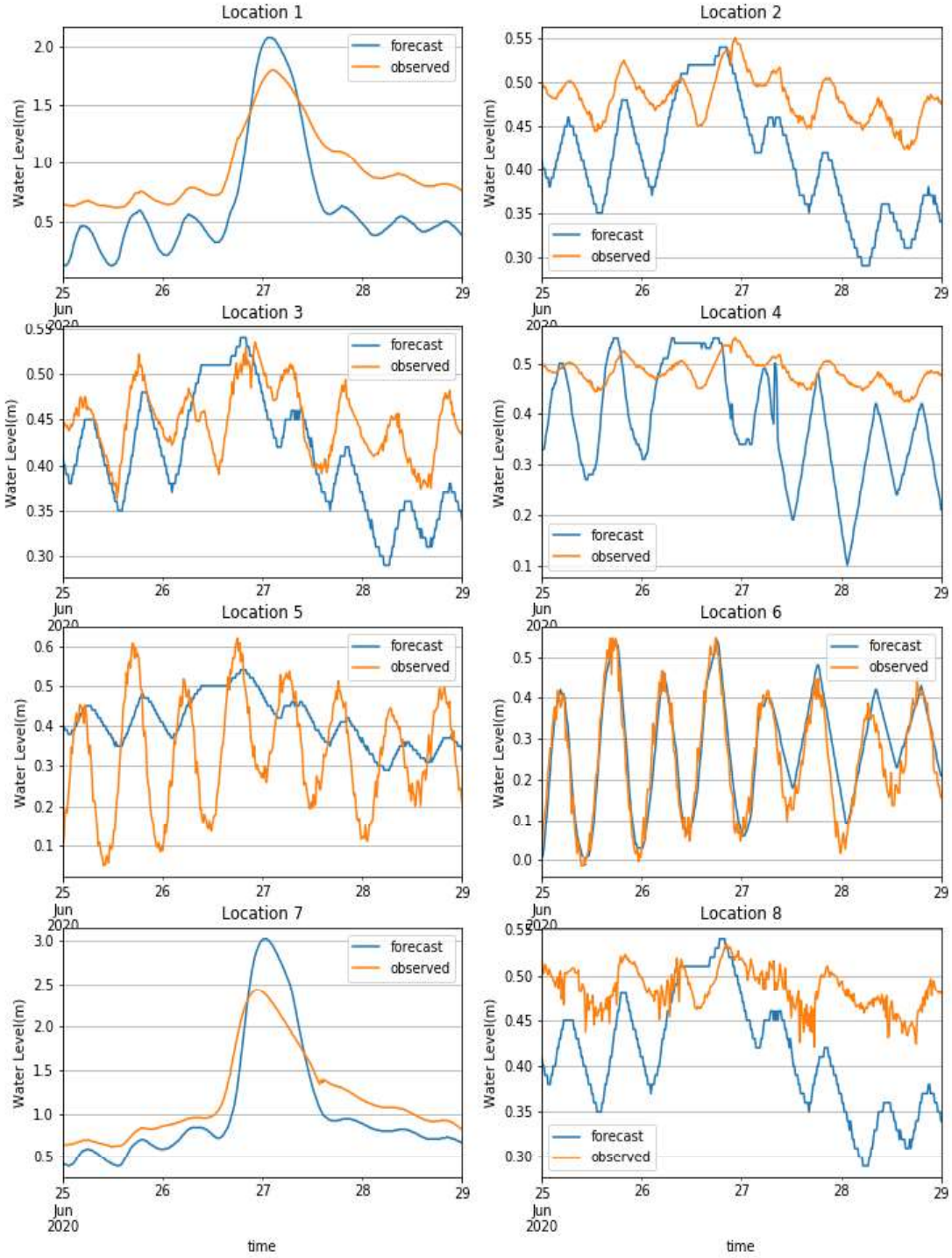


Figure 4.14 Selected location time-series comparison

#### 4.2.5.1 WRF model decision results

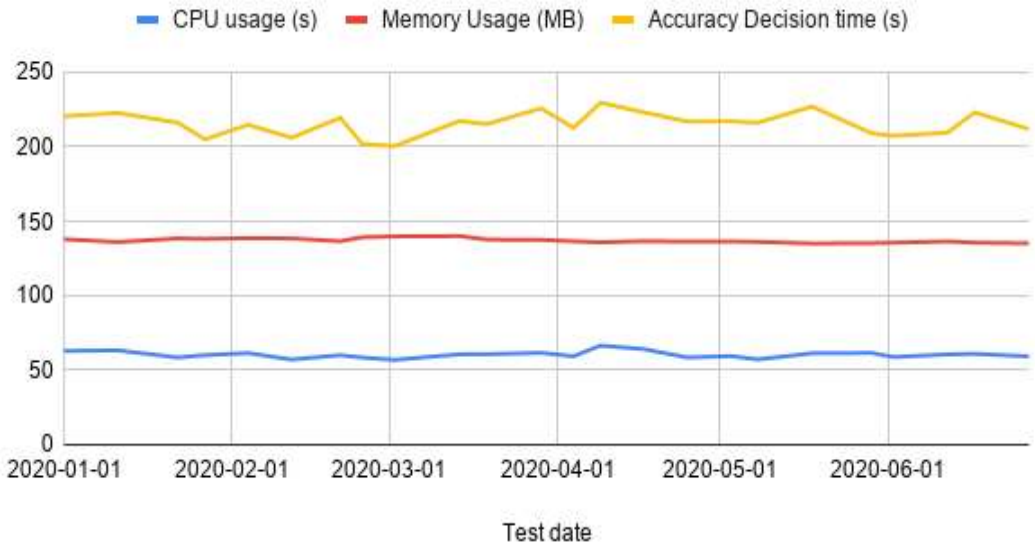


Figure 4.15 WRF accuracy decision resource consumption

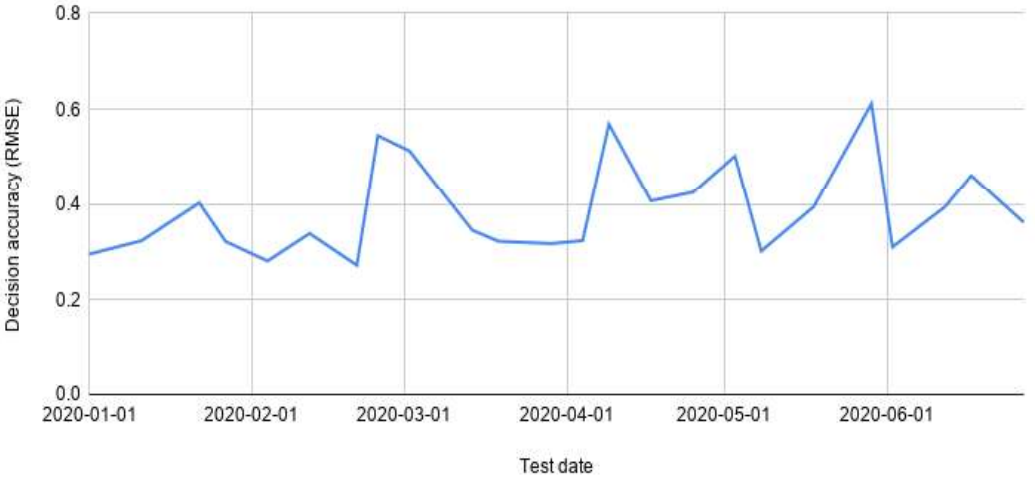


Figure 4.16 WRF decision accuracy (RMSE).

Executed four WRF models parallel inside four Docker containers for a single run. Around 50 observed stations have been installed in the related catchment area to collect observed data. To conduct an accuracy testing system has to retrieve observed time series and four forecast time series for each station from the database. A considerable amount of network delay has been added to the process since the database is in a separate VM, affecting decision-making time. Also, in Figure 4.15, we can observe CPU usage, and memory usage is nearly constant throughout the considered period. The model selection decision's accuracy is measured by calculating RMSE between the selected model forecast and observed data, lying under 0.6, as illustrated in Figure 4.16.

#### 4.2.5.2 HEC-HMS model decision results

We have executed four HEC-HMS models inside four Docker containers for a single run. The WRF model should retrieve four forecast time series and one observed time series from the database. HEC-HMS provides the water discharge forecast at a particular location (used as inflow for the FLO-2D model). Therefore, for HEC-HMS, we only consider one station. Because of that, in Figure 4.17, it has a negligible effect from network delays compared to the other two model types. Also, the RMSE value of HEC-HMS decision accuracy is under 0.8, as shown in Figure 4.18.

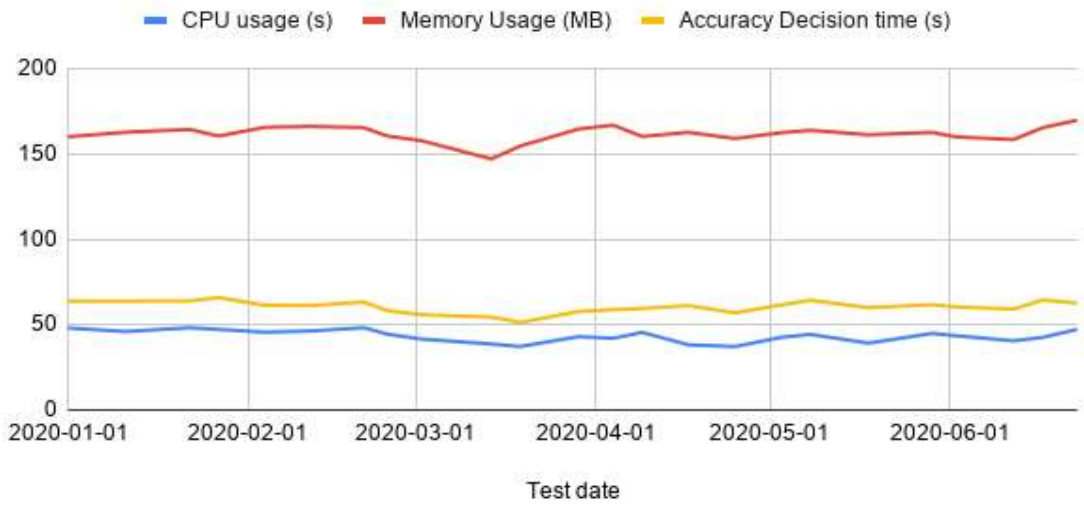


Figure 4.17 HEC-HMS accuracy decision resource consumption

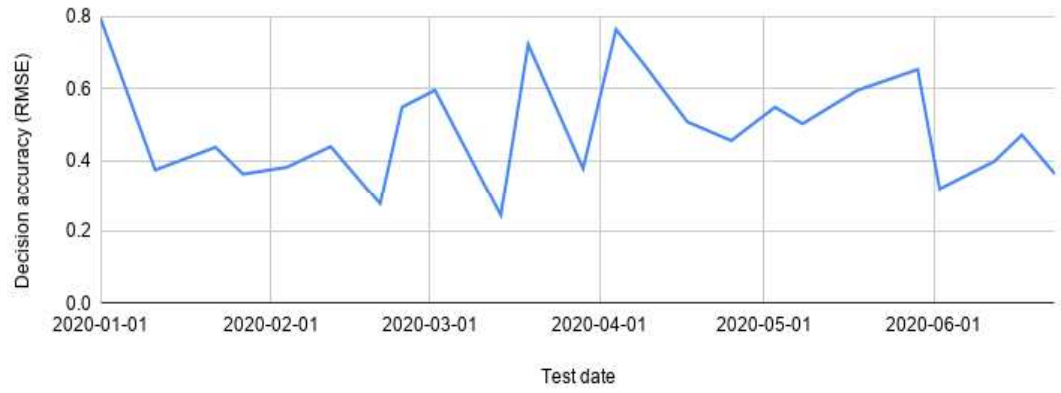


Figure 4.18 HEC-HMS decision accuracy (RMSE).

#### 4.2.5.3 FLO-2D 250m model decision results

For accurate decision-making, we only consider FLO-2D 250m model, which decides to take action on pump operations or unexpected event handling models. Figure 4.19 presents the accuracy decision resource consumption of the FLO-2D 250m model. The single-source data retrieval from the database network delay has an insignificant effect on accurate decision-making.

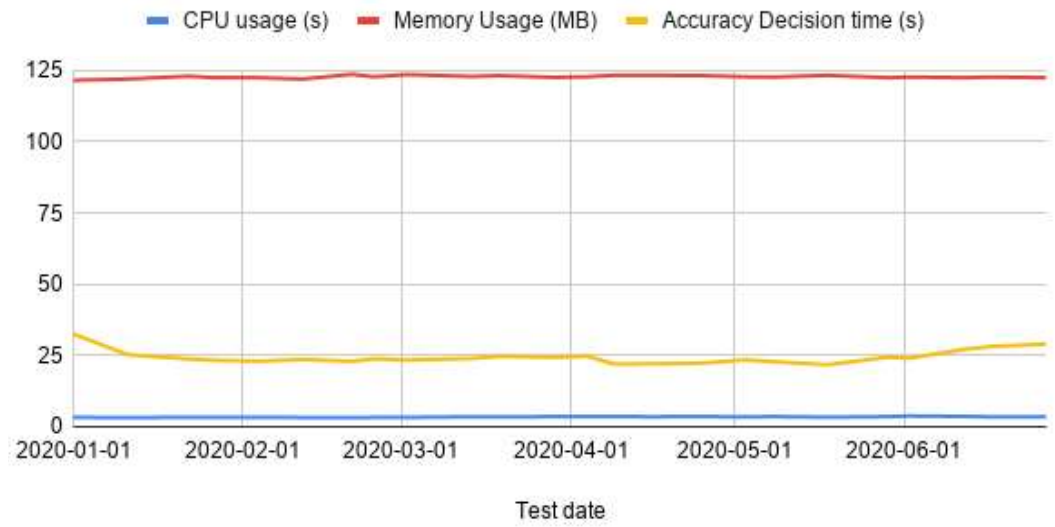


Figure 4.19 FLO-2D 250m accuracy decision resource consumption

We have considered 16 water level gauge stations in the catchment area, significantly impacting decision-making. As usual, each station's accuracy is measured, computing RMSE for each station using both forecast and observed time series. The percentage of the accurate station count from the total considered stations is the accuracy of the model result. As shown in Figure 4.20, the model's accuracy is equal to or above 75% throughout the considered period. We can reassure that the accuracy decisions made for predecessor models are accurate enough to produce valid workflow output for further analysis or decision making. Figure 4.21 presents total decision latency added to the workflow is around 5 minutes, which cannot be achieved in the manual accuracy checking process using a single human resource.

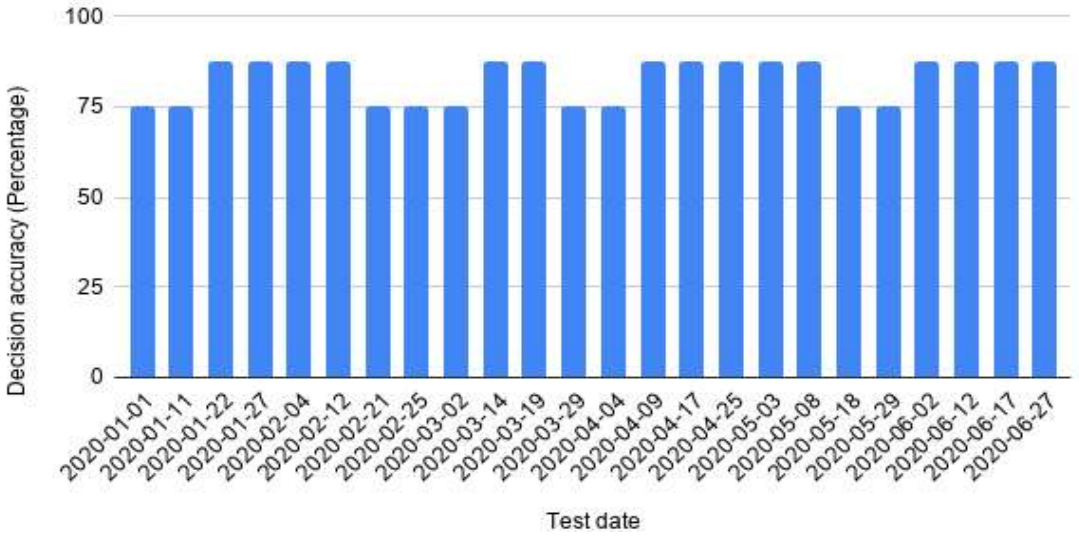


Figure 4.20 FLO-2D 250m decision accuracy (Percentage).

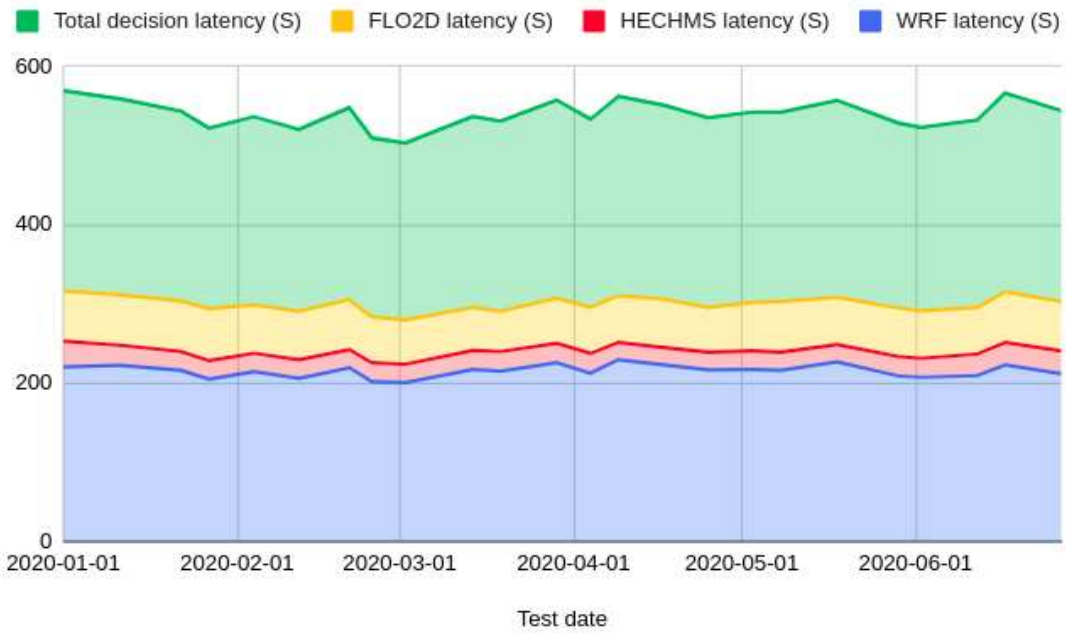


Figure 4.21 Total accuracy-based decision latency

Final accuracy decisions have been subject to the manual verification process and achieved a 100% acceptance rate. This means FLO-2D 250m model results generated by automated workflows can proceed to trigger FLO-2D 150m model for pump operations or unexpected event handling decisions without manual intervention. See Table 4.3 for total accuracy decision latency in the workflow. With this 100% success rate of accuracy-based model selection, CUrW currently employs this system for automated workflow management in their research & development process, which has saved field experts time and effort on initial decision making.

## 4.3 Infrequent event handling decisions

Weather is inherently chaotic; hence, decision-making on its behavior is challenging. When it comes to infrequent events, handling this decision-making process is even more challenging.

Table 4.3 Total accuracy decision latency in the workflow

| Test date | Total decision latency(S) | Allowed to proceed by manual verification |
|---|---|---|
| 2020-01-01 | 316.49 | Allowed |
| 2020-01-11 | 311.5 | Allowed |
| 2020-01-22 | 303.55 | Allowed |
| 2020-01-27 | 293.89 | Allowed |
| 2020-02-04 | 298.83 | Allowed |
| 2020-02-12 | 290.69 | Allowed |
| 2020-02-21 | 305.51 | Allowed |
| 2020-02-25 | 283.83 | Allowed |
| 2020-03-02 | 279.52 | Allowed |
| 2020-03-14 | 295.51 | Allowed |
| 2020-03-19 | 291.04 | Allowed |
| 2020-03-29 | 307.53 | Allowed |
| 2020-04-04 | 295.97 | Allowed |
| 2020-04-09 | 310.78 | Allowed |
| 2020-04-17 | 306.08 | Allowed |
| 2020-04-25 | 295.88 | Allowed |
| 2020-05-03 | 301.9 | Allowed |
| 2020-05-08 | 303.11 | Allowed |
| 2020-05-18 | 308.51 | Allowed |
| 2020-05-29 | 294.93 | Allowed |
| 2020-06-02 | 291.54 | Allowed |
| 2020-06-12 | 295.53 | Allowed |
| 2020-06-17 | 315.35 | Allowed |
| 2020-06-27 | 303.19 | Allowed |

### 4.3.1 Flash floods

Flash floods are flood events where the rise in water is either during or within a few hours of the rainfall that produces the rise. Therefore, flash floods occur within small catchments, where the response time of the drainage basin is short. In part, owing to the rapidly rising, fast-moving waters of a flash flood, the damage from them can be devastating [25].

We can list the following factors to check for flash floods:

- How quickly is the storm moving?
- How intense are the rainfall rates?
- Whether storms are redevelopment and repeatedly impacting the same area
- How much rainfall becomes surface runoff (and where it drains to)?
- The amount and type of vegetation covering the soil
- How much of the land surface is paved and will not absorb water?
- Whether there are storm drains to convey water out of the area effectively
- How steep is the terrain?

Analyzing the above-listed facts, we can narrow our focus to the rainfall intensity of the catchment area. Other factors depend on the catchment's geographical conditions and rainfall intensity. By varying the total rainfall value within an hour, we can build flash flood triggering logics for each catchment area.

We need a set of parameters or logic variables for each condition-based model triggering scenario to build the logical condition. Table 4.4 has listed a few logic rule variables for flash flood model triggering.

Table 4.4 Logic rule variables for flash flood model triggering

| Variable Name | Purpose | Updated Frequency |
|---|---|---|
| last_hour_rainfall | Get hourly rainfall intensity | Every 5 minutes (as rain gauges reporting frequency is 5 minutes) |
| last_1_day_rainfall | For getting an idea about water surface runoff | Every 5 minutes (as rain gauges reporting frequency is 5 minutes) |
| last_2_day_rainfall | For getting an idea about water surface runoff | Every 5 minutes (as rain gauges reporting frequency is 5 minutes) |

Also, we need the following two parameters are required for every variable to make them meaningful.

- location_name – For defining catchment area or nearest observed rain gauge
- variable_type – variable is about rainfall or water level

Using the logic variables, we can set the triggering logic as follows,

$$((location_{name} = '\ Location\ A') \ and \ (variable_{type} = '\ Precipitation') \ and \ (last\_hour\_rainfall\ > Value\ A)) \quad (4.1)$$

Table 4.5 FLO-2D 10m models and triggering conditions status

| FLO2D 10m model | Logic Condition | Model Triggered (*TRUE/FALSE*) |
|---|---|---|
| model1 | ((location_name='Location A') and (variable_type='Precipitation') and (last_hour_rainfall>=50)) | ***TRUE*** |
| model2 | ((location_name='Location B') and (variable_type='Precipitation') and (last_hour_rainfall>=45)) | ***TRUE*** |
| model3 | ((location_name='Location C') and (variable_type='Precipitation') and (last_hour_rainfall>=30)) | ***TRUE*** |
| model4 | ((location_name='Location D') and (variable_type='Precipitation') and (last_hour_rainfall>=42)) | ***TRUE*** |
| model5 | ((location_name='Location E') and (variable_type='Precipitation') and (last_hour_rainfall>=36)) | ***TRUE*** |

Figures 4.22 - 4.25 presents inundation maps for each selected set of locations,

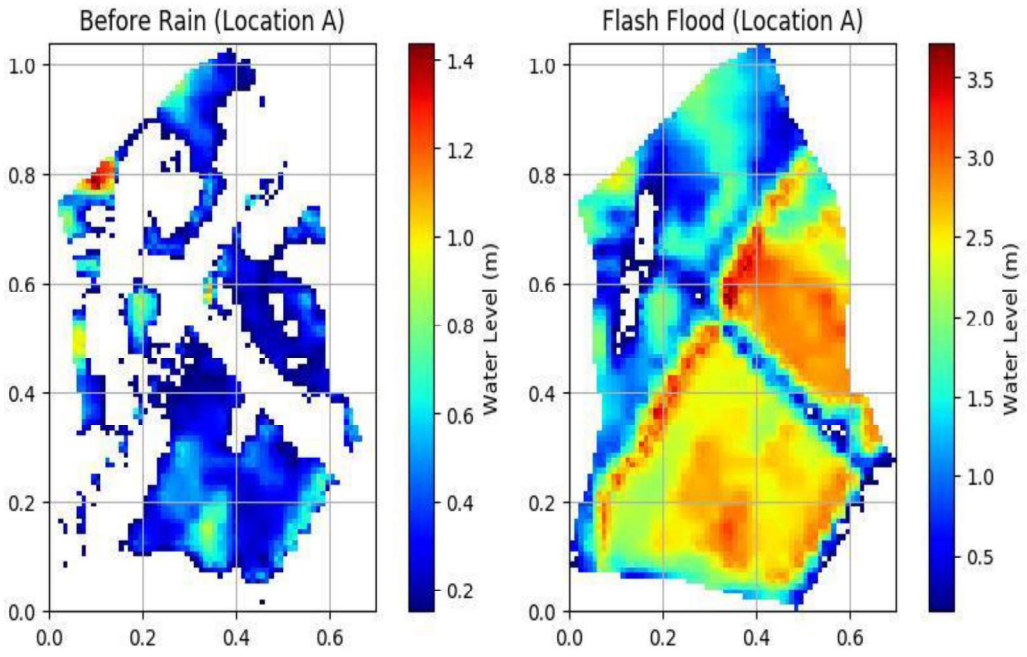


Figure 4.22 Location A flash flood

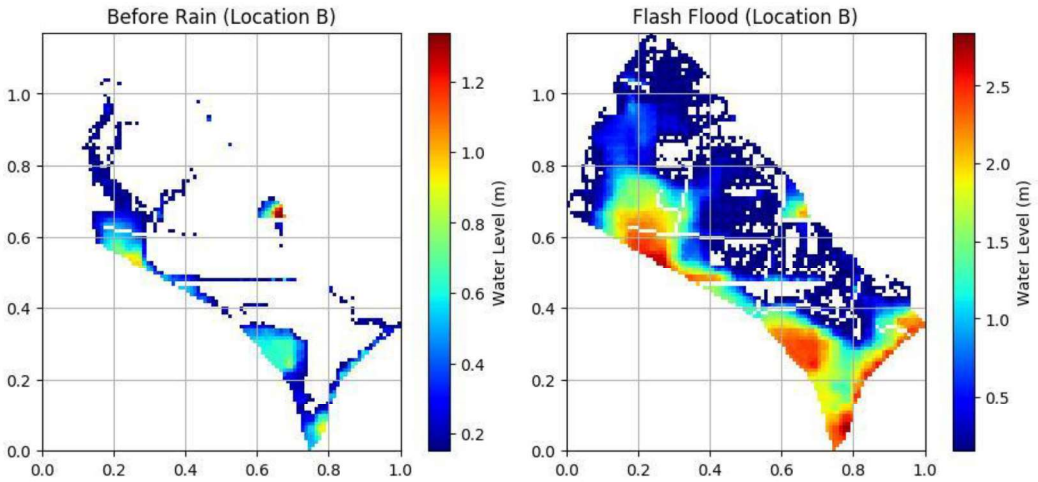


Figure 4.23 Location B flash flood

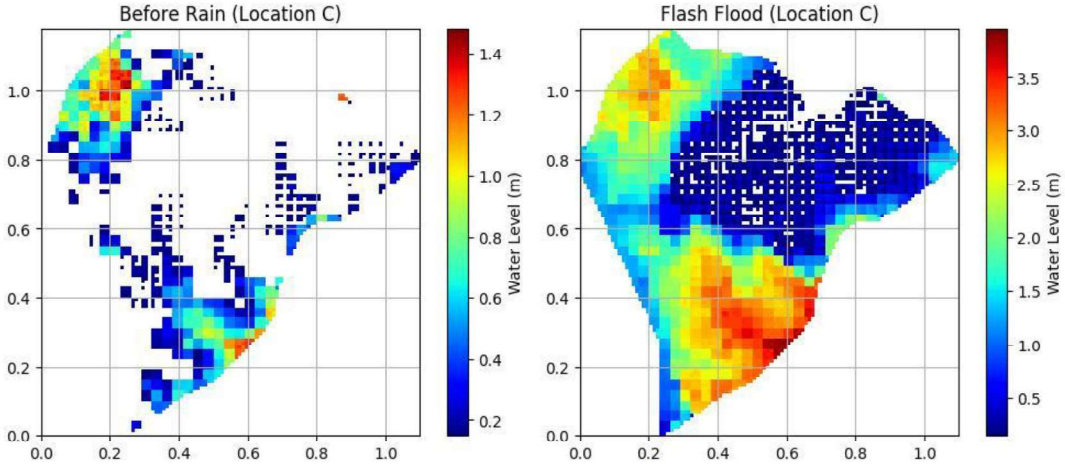


Figure 4.24 Location C flash flood

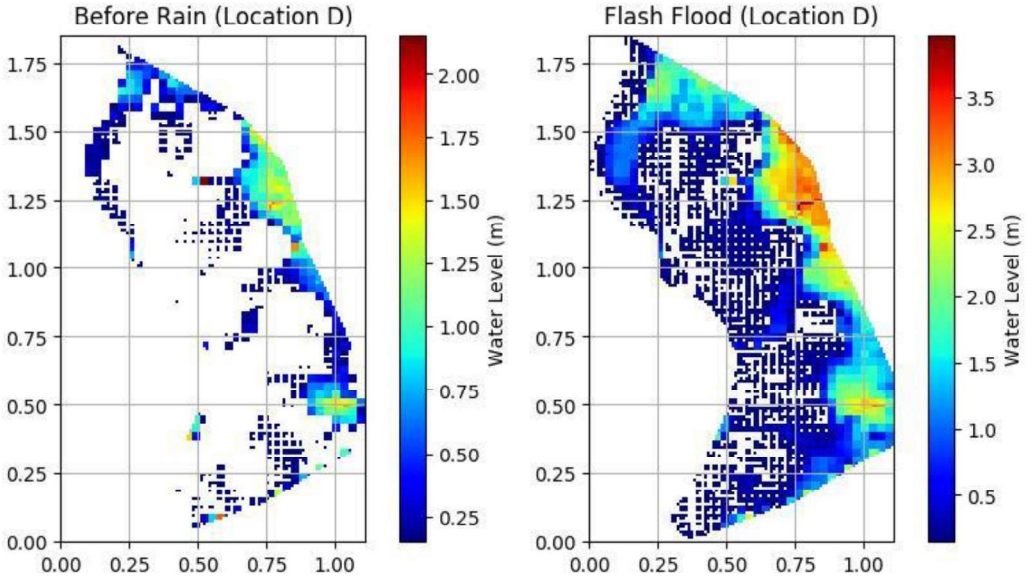


Figure 4.25 Location D flash flood

### 4.3.2 Dam failures

When a reservoir dam fails in the upper catchment, huge water volumes will be added to downstream water paths (in our condition, it is a river), so water discharge will increase without considerable rain in the upper catchment area.

Table 4.6 Logic rule variables for dam failure model triggering

| Variable Name | Purpose | Updated Frequency |
|---|---|---|
| location_k_obs_wl | Get observed water level of "location k" (initial condition point for lower catchment) | Every 5 minutes (as water level gauges reporting frequency is 5 minutes) |
| ub_last_1_day_rain | Last 24 hours rainfall in the upper catchment area | Every 5 minutes (as rain gauges reporting frequency is 5 minutes) |
| hechms_discharge | Highest discharge value from the latest HECHMS run | Every 1 hour |

Therefore, using the parameters listed in Table 4.6, we can build a model trigger logic condition. See Equation 4.2 dam failure model trigger logic condition.

$$((location\_name = 'Location\ K') \ and \ (variable\_type = 'WaterLevel') and \ (current\_water\_level > Value\_K)) and$$

$$((location\_name = 'Location\ K') \ and \ (variable\_type = 'Discharge')$$

$$and \ (current\_water\_level > Value\_Q))$$

$$and$$

$$((location\_name = 'KUB') \ and \ (variable\_type = 'Precipitation')$$

$$and \ (ub\_last\_1\_day\_rain < Value\_P)) \quad (4.2)$$

Consider the following general case shown in Figure 4.26 how catchment water levels behave under normal conditions. In the dam failure scenario, Figure 4.27 shows another two cases as Gate open and Gate close cases.

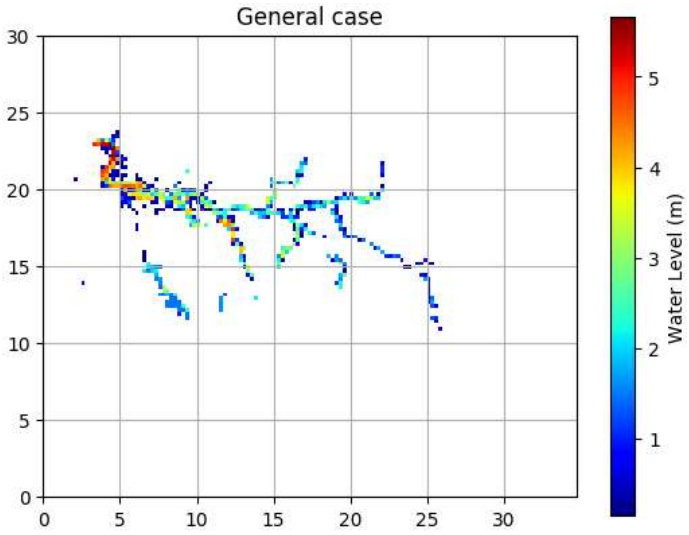


Figure 4.26 Flood map without any gates

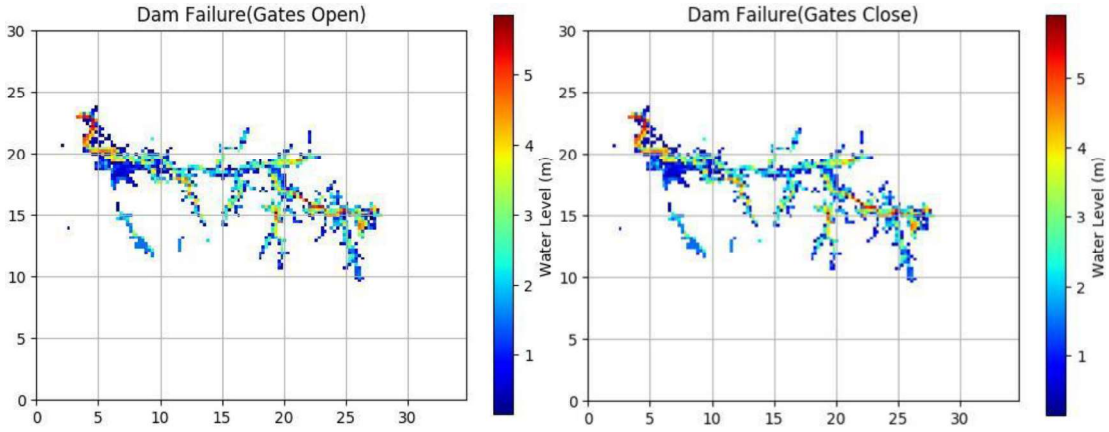


Figure 4.27 Dam failure Flood map

## 4.4 Pump operating decisions

Pumps are designed to pull excess water from the catchment, and gates are designed to stop water from entering the catchment area. Therefore, both types of units are needed to minimize the flood impact. CUrW has employed 3 pumping stations (with different capacities) and three gates. See Figure 4.28 for gate and pump locations.

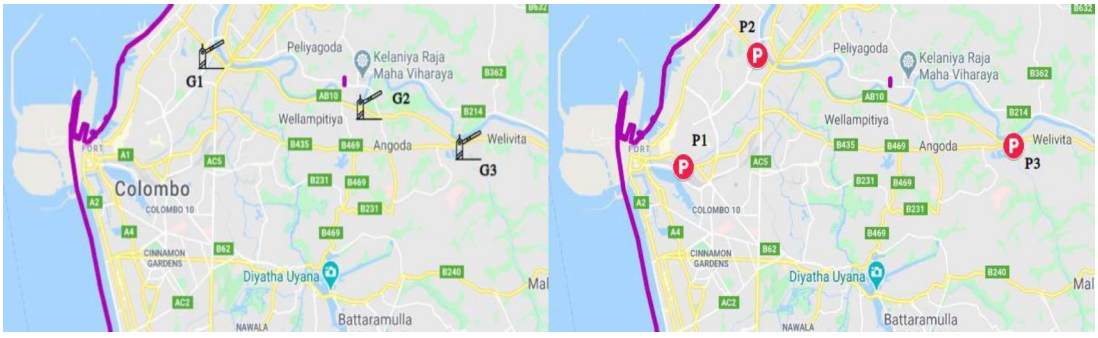


Figure 4.28 Gate (G) and Pump (P) locations

Lower catchment area can be flooded in two scenarios,

- High intense rain occurs in the catchment area,

  After water bodies (canals, ponds, wetlands) are saturated, the area will be flooded. For flood mitigation, excess water should be pumped out from the catchment area to the river or sea.

- River water level increases

  The lower catchment area has a low mean sea level than the river. Due to the heavy rainfall in the upper catchment area, the river water level will be increased. If it exceeds a certain level (alert/warning levels), the river water will enter the lower catchment, creating a flood in nearby areas.

Besides the above two scenarios, both pump and gate units should be operated in this worst-case scenario.

Gates's operations are limited to close or open conditions. In the worst-case scenario, gates are in closed conditions. Therefore, we can keep it static for the pumping configurations. Since pumps have different capacities, they can be operated at different levels, as listed in Table 4.7.

Combining operating modes, we have built a total of 51 pump configurations. Each case represents a different FLO-2D 150m model, so these 51 models will be available for triggering. Decisions on triggering will be based on rainfall distribution, forecast rainfall intensity, and currently observed water levels. Therefore, the parameter set in Table 4.11 has been used to build triggering logic conditions.

Table 4.7 Pump operating modes

| **Operation Condition \ Pump** | **P1 (30CMS)** | **P2 (20CMS)** | **P3 (10CMS)** |
|---|---|---|---|
| Full | 5 | 5 | 2 |
| Medium | 3 | 3 | - |
| Low | 1 | 1 | 1 |

Also, pumps can operate in different combinations, as listed in Table 4.8, Table 4.9, and Table 4.10.

Table 4.8 Single pump operation

| **Cases \ Pumps** | **P1** | **P2** | **P3** |
|---|---|---|---|
| 1 | operating | - | - |
| 2 | - | operating | - |
| 3 | - | - | operating |

Table 4.9 Simultaneously two pumps operation

| **Cases \ Pumps** | **P1** | **P2** | **P3** |
|---|---|---|---|
| 4 | operating | operating | - |
| 5 | operating | - | operating |
| 6 | - | operating | operating |

Table 4.10 Simultaneously all three pumps operation

| **Cases \ Pumps** | **P1** | **P2** | **P3** |
|---|---|---|---|
| 7 | operating | operating | operating |

Table 4.11 Pump logic parameters

| Parameter | Refresh Interval | Description |
|---|---|---|
| current_rainfall | Every 5 minutes | Will be maintained for selected stations, from observed gauges |
| current_waterlevel | Every 5 minutes | Will be maintained for selected stations, from observed gauges |
| klb_total_1_day | Every 5 minutes | Total rainfall in mm for last 24 hours in lower catchment area |
| klb_fcst_total_1_day | After every WRF run | Total rainfall forecast in mm for next 24 hours in lower catchment area |
| klb_total_3_day | Every 5 minutes | Total rainfall in mm for last 72 hours in lower catchment area |
| kub_total_1_day | Every 5 minutes | Total rainfall in mm for last 24 hours in upper catchment area |
| kub_fcst_total_1_day | After every WRF run | Total rainfall forecast in mm for next 24 hours in upper catchment area |
| kub_total_3_day | Every 5 minutes | Total rainfall in mm for last 72 hours in upper catchment area |
| alert_water_level | Static value set | Will be maintained for selected stations |
| warning_water_level | Static value set | Will be maintained for selected stations |
| max_150_wl_fcst | After every new FLO2D 150m model | Will be maintained for selected stations |
| max_discharge_fcst | After every new HECHMS model | Maintain for selected HECHMS extraction point |
| obs_discharge | Every 5 minutes | Will be maintained for the selected single station (exact locations as HEC-HMS extraction point), calculated from the observed water level of the particular location |

Parameter sets that are used to build logic conditions are not static. Users can update parameters in each set. The only limitation is that users should define its value update strategy when adding new parameters. Otherwise, that parameter is invalid for building logic conditions. A significant difference between other model trigger conditions and pump model trigger conditions is, if the condition is evaluated to *TRUE*, it will trigger a set of FLO-2D 150m models with different configurations. The main reason is that it is hard to select the best-performing model based on a logic condition. Therefore, we will trigger a selection of models related to the logic condition and select the model that minimizes the loss. Also, there can be situations where one or more logic conditions are evaluated to *TRUE*.

Using the parameters listed in Table 4.11, we can build the following model trigger logic condition(s).

$$[((location_{name} =' Location\ A') \ and \ (variable_{type} =' WaterLevel')\ and$$

$$(current\_water\_level > alert\_water\_level))$$

$$or\ ((location\_name = 'Location\ A')\ and\ (variable\_type = 'WaterLevel')$$

$$and\ (current\_water\_level >= warning\_water\_level))]$$

$$and[((location\_name = 'KUB')\ and\ (variable\_type = 'Precipitation')$$

$$and\ (kub\_last\_1\_day\_rain >= Value1))$$

$$or\ ((location\_name = 'Location\ A')\ and\ (variable\_type = 'Discharge')$$

$$and\ (obs\_discharge >= Value2))]$$

$$and$$

$$[[(((location_{name} =' Location\ B')\ and\ (variable_{type} =' WaterLevel')$$

$$and\ (current\_water\_level > alert\_water\_level))$$

$$and\ ((location\_name = 'Location\ C')\ and\ (variable\_type = 'WaterLevel')$$

$$and\ (current\_water\_level > alert\_water\_level))$$

$$and\ ((location_{name} =' Location\ D')\ and\ (variable_{type} =' WaterLevel')$$

$$and\ (current\_water\_level > alert\_water\_level))]$$

$$or\ (((location_{name} =' KLB')\ and\ (variable_{type} =' Precipitation')$$

$$and\ (kub\_last\_1\_day\_rain >= Value3)))] \quad (4.3)$$

Users can build more logic conditions by updating Value1, Value2, and Value3, adding more or removing parameters. When we have multiple logic conditions, they will trigger in the given order one after another. Every logic that is evaluated to ***TRUE*** will trigger a set of pump control models. After evaluating results, the minimum loss model will be selected as the given scenario's pump control strategy. Model results evaluating process will be manual, including factors like area affected, total damage, and number of people at risk.

### 4.4.1 Simulation

We used simulated data prepared for the 2010 flood in Sri Lanka (Kelani river basin area) for the performance analysis. As Value1, Value2 and Value3 are configurable, we have prepared 3 logic sets mentioned in Table 4.12 by changing those values.

Table 4.12 2010 Flood simulations cases

| Case No | Value1(mm) | Value2(m3/s) | Value3(mm) |
|---|---|---|---|
| 1 | 100 | 2500 | 70 |
| 2 | 70 | 2000 | 80 |
| 3 | 80 | 1600 | 65 |

Logic conditions have triggered in the order of case1, case 2, and case 3, one after the other with 2010 flood conditions. Each case has six pump configuration models. In the simulation, case 2 and case 3 have been evaluated as ***TRUE***, and relevant model sets have been triggered.

### 4.4.2 Results

First, we consider the flood map for the 2010 flood without any pump or gate in operation. (Figure 4.29 2010 Flood map with existing conditions). By observing, we can identify the flood-affected areas. The gate closure effect is minimum for significant floods, especially when heavy rains are on the lower catchment. Therefore, pumps should operate at their optimized strategy to reduce flood damage.

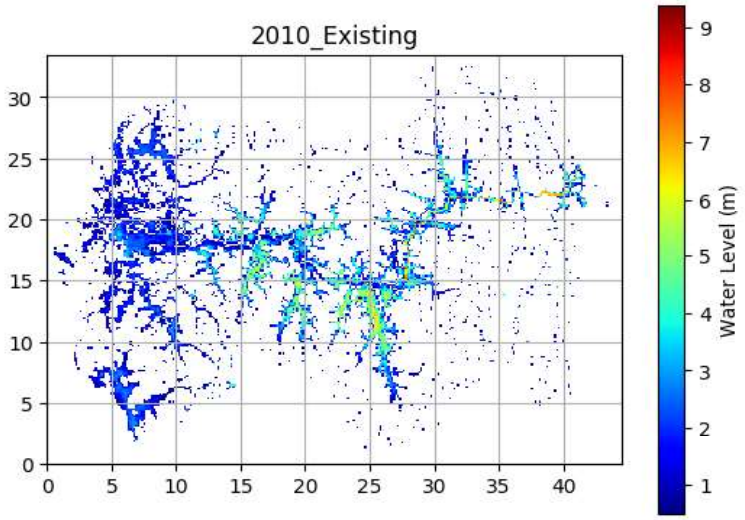


Figure 4.29 2010 Flood map with existing conditions

#### 4.4.2.1 Case 1

Field experts have built six pumping configurations based on a P2 pump station, operating at its total capacity. Case 1 pump configuration is given in Table 4.13.

However, the system triggered none of the models because the case 1 logic condition has been evaluated *FALSE*.

Table 4.13 Case 1 pump configuration

| Configuration | P1 | P2 | P3 | Triggered |
|---|---|---|---|---|
| 1 | F | F | F | No |
| 2 | F | F | L | No |
| 3 | L | F | F | No |
| 4 | L | F | L | No |
| 5 | M | F | F | No |
| 6 | M | F | L | No |

#### 4.4.2.2 Case 2

For case 2, field experts have built six pumping configurations based on a P2 pumping station operating in its medium capacity. As the case 2 logic condition has been evaluated to ***TRUE***, the system has triggered all the models listed in Table 4.14. See flood maps generated from each pump configuration results in Figure 4.30.

Table 4.14 Case 2 pump configurations

| Configuration | P1 | P2 | P3 | Triggered |
|---|---|---|---|---|
| 1 | F | M | F | Yes |
| 2 | F | M | L | Yes |
| 3 | L | M | F | Yes |
| 4 | L | M | L | Yes |
| 5 | M | M | F | Yes |
| 6 | M | M | L | Yes |

Table 4.15 Case 3 pump configurations

| Configuration | P1 | P2 | P3 | Triggered |
|---|---|---|---|---|
| 1 | F | L | F | Yes |
| 2 | F | L | L | Yes |
| 3 | L | L | F | Yes |
| 4 | L | L | L | Yes |
| 5 | M | L | F | Yes |
| 6 | M | L | L | Yes |

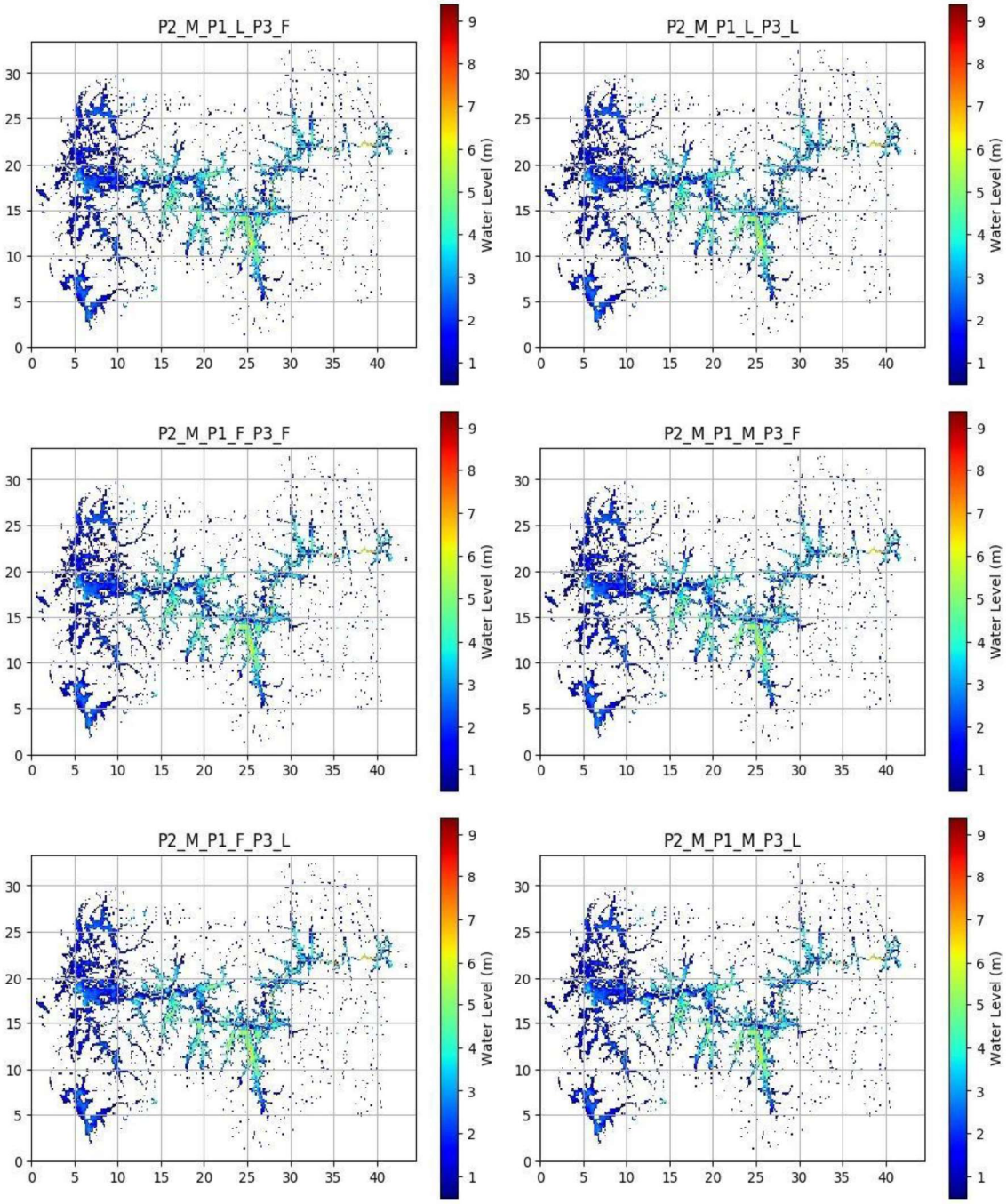


Fig. 4.30. Case 2 flood maps for 6 pump configurations.

#### 4.4.2.3 Case 3

In case 3, field experts have built six pumping configurations based on the P3 pumping station operating in its lower capacity. If the case 3 logic condition evaluates as *TRUE*, the system will trigger all the models listed in Table 4.15. See flood maps generated from each pump configuration results in Figure 4.31.

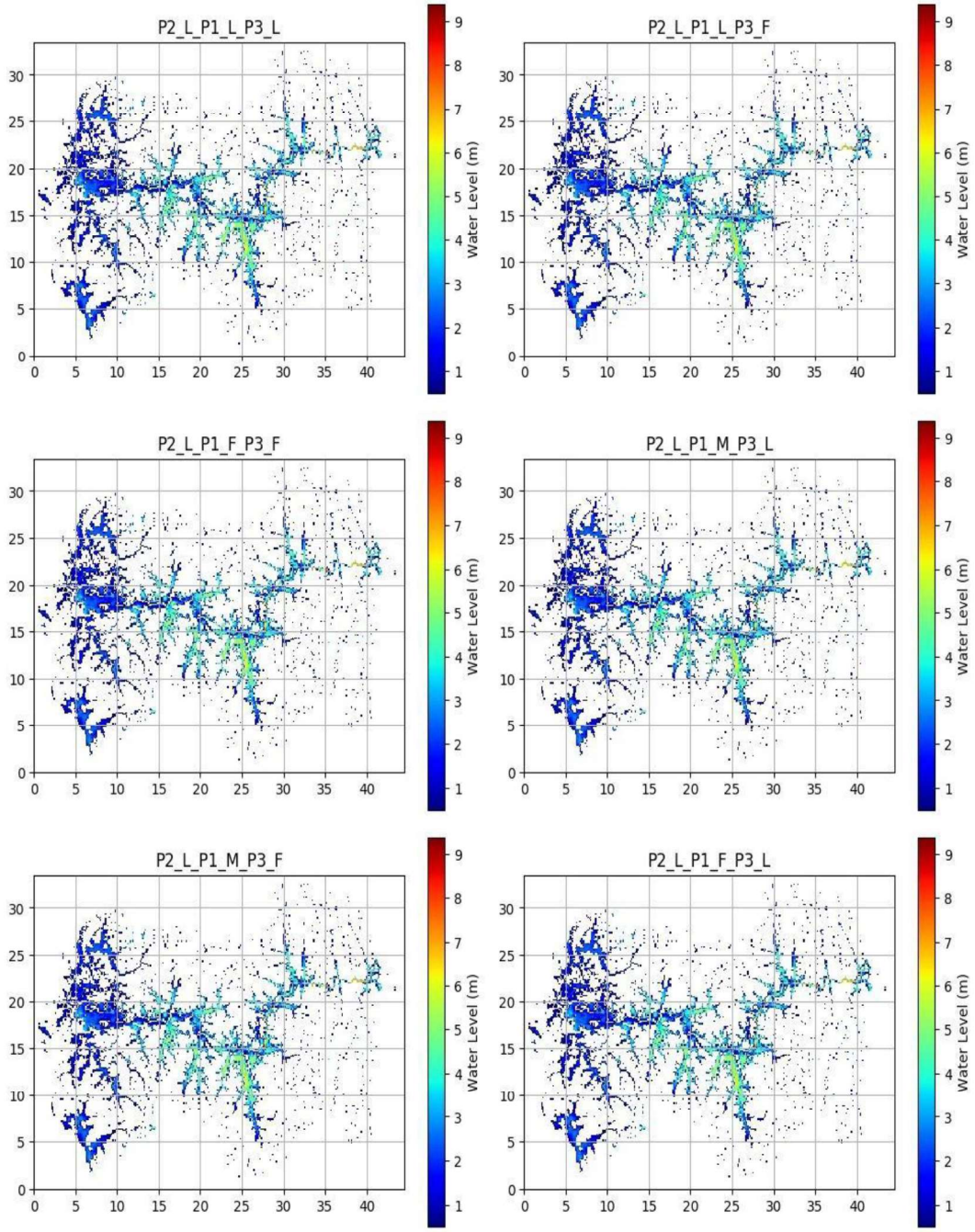


Figure 4.30 Case 3 flood maps for 6 pump configurations

### 4.4.3 Analysis

Due to the low-resolution level, it is challenging to analyze pump configuration results and select the best configuration for pump operation. The system has triggered case 2 and case 3 models to select the best model from 12 models. See Table 4.17 case 2 pump model results and Table 4.18 case 3 pump model results. The results are calculated by referencing the no pump operating scenario mentioned in Table 4.16.

Table 4.16 Loss estimation with no pump operating scenario

| Area (km^2) | Total Damage(m) | People at Risk |
|---|---|---|
| 22.757 | 35888.7 | 98618 |

Table 4.17 Case 2 pump model results

| Case No. | Pump Config | Area saved ($km^2$) | Properties saved (mn) | People Saved |
|---|---|---|---|---|
| 1 | P2_M_P1_F_P3_F | 1.366 | 3012.5 | 5082 |
| 2 | P2_M_P1_F_P3_L | 1.343 | 3007.4 | 4829 |
| 3 | P2_M_P1_L_P3_F | 0.871 | 1410.8 | 1549 |
| 4 | P2_M_P1_L_P3_L | 0.893 | 1432.8 | 1620 |
| 5 | P2_M_P1_M_P3_F | 1.073 | 2100.2 | 2895 |
| 6 | P2_M_P1_M_P3_L | 1.073 | 2112.1 | 2895 |

Table 4.18 Case 3 pump model results

| Case No. | Pump Config | Area saved ($km^2$) | Properties saved (mn) | People Saved |
|---|---|---|---|---|
| 7 | P2_L_P1_F_P3_F | 1.456 | 3155.7 | 5083 |
| 8 | P2_L_P1_F_P3_L | 1.433 | 3143.2 | 5266 |
| 9 | P2_L_P1_L_P3_F | 0.938 | 1457.5 | 1610 |
| 10 | P2_L_P1_L_P3_L | 0.938 | 1452.7 | 1610 |
| 11 | P2_L_P1_M_P3_F | 1.163 | 2191.6 | 3002 |
| 12 | P2_L_P1_M_P3_L | 1.118 | 2185 | 3243 |

Pump configuration models will be selected based on the “Number of people saved” factor, with the highest value. Therefore, according to the pump statistics listed in Table 4.19, the best performing configuration shown in Figure 4.32 for the scenario is case no. 8 (P2 - Low, P1 - Full, P3 - Low).

Table 4.19 Number of potential people save with each case

| Case No. | People Saved |
|---|---|
| 1 | 5082 |
| 2 | 4829 |
| 3 | 1549 |
| 4 | 1620 |
| 5 | 2895 |
| 6 | 2895 |
| 7 | 5083 |
| 8 | 5266 |
| 9 | 1610 |
| 10 | 1610 |
| 11 | 3002 |
| 12 | 3243 |

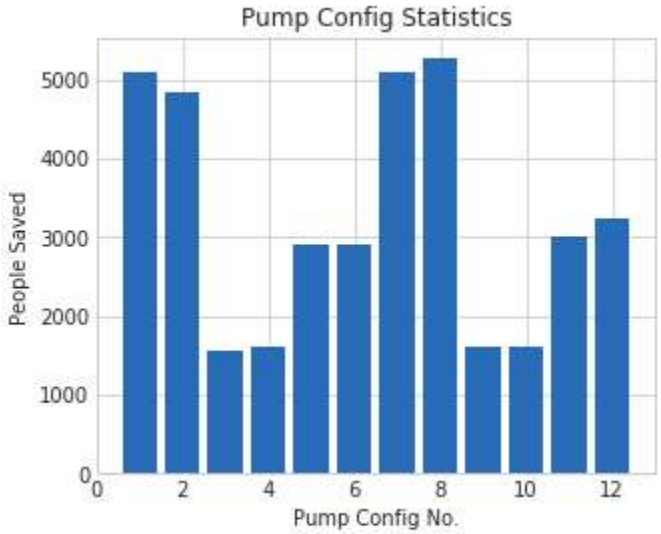


Figure 4.31 Pump Statistics

.

# 5 SUMMARY

This chapter concludes the study by summarizing the problem formulation and findings of the study. Section 5.1 presents the research conclusions, and research limitations are presented in Section 5.2. Finally, Section 5.3 offers future work and a way to expand our work.

## 5.1 Conclusions

We need to integrate several modeling systems for weather monitoring, forecasting, and risk estimations when forecasting weather conditions for disaster control purposes or agricultural needs.

Additionally, due to the weather's chaotic nature, multiple models are executed in parallel to select the most suitable model executions, terminate unfitted executions, and reuse the resources for the next iteration. Moreover, there are cascading relationships between these models where one model's output is transferred to another.

Existing Decision support systems are restricted to a riverbank or narrow scopes, such as making decisions with only floods, storms, or reservoir water level management. Moreover, they are less adaptable to unexpected conditions such as sudden heavy rain, breakdown of pumps and gates, or reservoir damage.

This research proposed a generic Decision support system framework for precipitation precipitation-related decision decision-making. This system can create and manage complex dynamic weather model workflows while allowing users to enforce operational logic as rules on workflow structure and execution. Therefore, the system can reduce field experts' effort on decision decision-making related to weather and saving computing resources.

The proposed system can handle three types of decision conditions. Accuracy Accuracy-based decision decision-making allows users to control workflows based on their accuracy. Due to the cascade relationship between models, it is essential to have the required accuracy in the predecessor model results for the ultimate decision. For a six months' time period6-months, we have measured the accuracy of the system's accuracy-based decisions. We were able to verify the system's accuracy-based decisions with a 100% success rate with experts' decisions (manual verification).

Pump/gate operating decision-making is one of the system's crucial functions, which allows making one of the system's essential functions, allowing users to find the best (minimize the loss) pump control strategy for the current weather conditions. Using the rule engine, users can build relevant logic to identify weather conditions that need pump operations. Using the 2010 Colombo flood scenario, we have verified this functionality.

In Unexpected event handling-based decisions, the system has successfully identified the Flash flood scenario with the given logic. Identification of an event will be based on the defined logic set. To identify more events, users can build their relevant logic conditions. Therefore, users can expand this system's capability by implementing more logics.

## 5.2 Research Limitations

In this section, we discuss the limitations of this research. WRF model input GFS data world widely provided by an organization named National Oceanic and Atmospheric Administration (NOAA). They are pushing GFS data to an FTP server four times a day. Because of some reason, they have removed past data beyond 2019. To conduct full workflow simulation in the past beyond 2019 is not practical, as WRF input data are not available. Therefore, for accuracy-based decision testing, we have used data from 2019, but the selected period has not included any severe weather event.

In this framework, the primary parameter is precipitations, but other factors also affect the rainfall, such as wind speed, wind direction, temperature, and humidity [26]. Therefore, when providing current weather status as feedback, we should also consider those parameters.

For performance testing in pump-based decisions for past flood scenarios, we needed WRF data in 2010, 2016, 2017. Also, we needed rainfall and discharge observed data for the same period. In those years, significant flood events were reported in the selected basin area. As WRF data and observed data are not available, we have moved with synthetic data generated by the field experts. Therefore, WRF model execution and model selection were not included in this pump decision simulation workflow.

Also, the field experts have not finalized pump model decision logics and models by the time that analysis has been conducted. Therefore, in the results, we cannot see a significant difference from the generated flood maps due to low water level changes as the models are not optimized. Therefore, for showing the impact of the pump operation, we have included results from the post analysis to show the impact of the pump operation.

In infrequent event handling, we mentioned flash floods. However, we cannot forecast flash floods with existing models and resources. We cannot predict there will be a flash flood in certain areas. That is because the current models (WRF) cannot forecast the rain that causes a flash flood. For that, we need satellite data and radar for detecting storms [27].

## 5.3 Future Work

We have provided the room to trigger external scripts or applications, so in the future, if field experts create such a script, we can integrate them. Moreover, once pump model triggering conditions have been finalized, we should also implement their value update strategies if they require new rule variables. We can add a new Airflow DAG with the required update frequency. If new weather model types need to be added, we need to create an automation sub-workflow for that model type.

As mentioned in Chapter 4, we are using the RMSE value of the WRF results with observed rainfall in accuracy-based decisions. Then we select the model with the lowest RMSE value as the basic WRF accuracy rule, but we have observed that a single WRF model result cannot match both time and quantity in all the locations. Therefore, WRF model selection should not depend only on the RMSE value. It should have further analysis (spatial analysis) to match the time and quantity both.

Kubernetes is an open-source system for automating deployment, scaling, and managing containerized applications [28]. By integrating Kubernetes with this framework, we can leverage this into a more efficient scalability and resource utilization system.

# REFERENCES


[1] NCEI, "Numerical weather prediction," 2020. [Online]. Available: https://www.ncei.noaa.gov/products/weather-climate-models/numerical-weather-prediction. [Accessed 24 2 2020].

[2] J. C. Chambers, "How to choose the right forecasting technique," 7 1971. [Online]. Available: https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique. [Accessed 25 2 2020].

[3] "Weather research and forecasting model," 2020. [Online]. Available: https://www.mmm.ucar.edu/weather-research-and-forecasting-model. [Accessed 2 3 2020].

[4] US Army Corps of Engineers, "HEC-HMS," US Army Corps of Engineers, 1 1997. [Online]. Available: https://www.hec.usace.army.mil/software/hec-hms. [Accessed 2 3 2020].

[5] FLO-2D Software Inc, "FLO-2D," 2020. [Online]. Available: https://www.flo-2deurope.com/en/flo-2d/. [Accessed 6 3 2020].

[6] M. Zink, E. Lyons, W. David, J. Kurose and D. L. Pepyne, "Closed-loop architecture for distributed collaborative adaptive sensing of the atmosphere: meteorological command and controlClosed-loop architecture for distributed collaborative adaptive sensing of the atmosphere: meteorological command and control," *International Journal of Sensor Networks,* vol. 7, no. 1/2, pp. 4-18, 2010.

[7] A. Marcomini, G. Suter and A. Critto, "A decision support system for the management of the Sacca di Goro (Italy)," in *Decision Support Systems for Risk-Based Management of Contaminated Sites*, Springer, 2009, pp. 1-24.

[8] L. Saez A, J. Regodón, L. Panizo and M.-d.-M. Gallardo, "A DSS for reservoirs operation based on the execution of formal models," in *Hydroinformatics 2014*, New York, 2014.

[9] S. Belginova, I. Uvaliyeva and A. Ismukhamedova, "Decision support system for diagnosing anemia," in *2018 4th International Conference on Computer and Technology Applications (ICCTA)*, 2018.

[10] K. and G. Peter, "Decision support systems : a research perspective," Cambridge, Mass. : Center for Information Systems Research, Afred P. Sloan School of Management, 1980.

[11] TechTarget Contributor, "What is a decision support system (DSS)," TechTarget Contributor, 2018. [Online]. Available: https://searchcio.techtarget.com/definition/decision-support-system. [Accessed 8 5 2018].

[12] H. Ltifi, G. Trabelsi, M. Ben Ayed and A. M. Alimi, "Dynamic decision support system based on bayesian networks," *International Journal of Advanced Research in Artificial Intelligence,* vol. 1, 2012.

[13] H. Stubblefield, "Infections caught in the hospital," healthline, 6 2017. [Online]. [Accessed 29 2 2020].

[14] C. Fortescue and M. YK Wee, "Analgesia in labour: non-regionaltechniques," *Continuing Education in Anaesthesia, Critical Care & Pai,* vol. 5, 2005.

[15] GeekforGeeks, "KDD process in data mining," 2020. [Online]. Available: https://www.geeksforgeeks.org/kdd-process-in-data-mining/. [Accessed 29 2 2020].

[16] Howard J. Hamilton, "Overview of the KDD process," University of Regina, 7 2018. [Online]. Available: http://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1_kdd.html. [Accessed 29 2 2020].

[17] S. El-Sappagh, J. M Alonso, F. Ali and A. Ali, "An ontology-based interpretable fuzzy decision support system for diabetes diagnosis," *IEEE Access,* vol. 6, pp. 37371-37394, 2018.

[18] F. Mansouripour and S. Asadi, "Development of a Reinforcement Learning-based Evolutionary Fuzzy Rule-Based System for diabetes diagnosis," *Computers in Biology and Medicine 91,* 2017.

[19] M. M. N. Kumar, M. M. Abedin and M. S. Islam, "Comparative approaches for classification of diabetes mellitus data: machine learning paradigm," *Computer Methods and Programs in Biomedicine,* vol. 152, pp. 23-24, 2017.

[20] MathWorks, "Mamdani and Sugeno fuzzy inference systems," The MathWorks, Inc, 2020. [Online]. Available: https://www.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html. [Accessed 29 2 2020].

[21] NCAR, "Weather research and forecasting model," National Center for Atmospheric Research, 2020. [Online]. Available: https://www.mmm.ucar.edu/weather-research-and-forecasting-model. [Accessed 2 3 2020].

[22] Hydrologic Engineering Center, "HEC-DSSVue," U.S. Army Corps of Engineers, 2020. [Online]. Available: https://www.hec.usace.army.mil/software/hec-dssvue/. [Accessed 5 3 2020].

[23] Apache Airflow, "Apache Airflow documentation," Apache Airflow, 2020. [Online]. Available: http://apache-airflow-docs.s3-website.eu-central-1.amazonaws.com/docs/apache-airflow/latest/index.html#. [Accessed 6 3 2020].

[24] Docker, "Use containers to build, share and run your applications," Docker, 2020. [Online]. Available: https://www.docker.com/resources/what-container. [Accessed 10 3 2020].

[25] D. P. M. B. and D. Z. , "Analysis of flash-flood runoff response, with examples from major european events," *Treatise on Geomorphology,* vol. 7, pp. 95-104, 2013.

[26] Department of Meteorology, "Weather Forecasts," Department of Meteorology, 2016. [Online]. Available: http://www.meteo.gov.lk/index.php?lang=en. [Accessed 15 12 2021].

[27] NOAA National Severe Storms Laboratory, "Severe Weather 101," NOAA National Severe Storms Laboratory, 2021. [Online]. Available: https://www.nssl.noaa.gov/education/svrwx101/floods/detection/. [Accessed 15 12 2021].

[28] The Kubernetes Authors, "Production-grade gontainer orchestration," The Linux Foundation, 2021. [Online]. Available: https://kubernetes.io/. [Accessed 16 12 2021].