Design Document
Team 5
WPEAR

By:
Stephen Harrell
Lala Vaishno De
Mengxue Luo
Dhairya Doshi

I. Purpose:

Forecast accuracy is an important topic in both operational forecasting and atmospheric research. In order to facilitate accuracy in forecasts we must measure the difference in what was forecasted and what happened. To meet this need we will be designing and implementing a pipeline which will take publicly available forecasts and data from weather instruments and create derivative products with statistical information about the accuracy of the forecasts. The products will consist of visualizations on a website as well as grib files that can be used to do further analysis of the data.

Definitions

1. **NEXRAD Radar** - A US based network of radar stations that primarily detect reflectivity. (Map of NEXRAD Radar Stations: https://www.roc.noaa.gov/wsr88d/maps.aspx)

2. **Multi-Radar/Multi-Sensor System (MRMS)** - MRMS is a system with automated algorithms that quickly and intelligently integrate data streams from multiple radars, surface and upper air observations, lightning detection systems, and satellite and forecast models. (Further information: http://www.nssl.noaa.gov/projects/mrms/)

3. **NWS** - US National Weather Service (Further Information: http://www.weather.gov/)

4. **Grid Spacing** - In weather models the grid spacing is the size of each “pixel” in a forecast or observation. A common grid spacing is a 4 sq. km or 2 sq. km square area. (Further information: http://weather.mailasail.com/Franks-Weather/Grid-Length-Resolution)

5. **HRRR** - The HRRR is a NOAA real-time 3-km resolution, hourly updated, cloud-resolving, convection-allowing atmospheric model, initialized by 3km grids with 3km radar assimilation. Radar data is assimilated in the HRRR every 15 min over a 1-h period adding further detail to that provided by the hourly data assimilation from the 13km radar-enhanced Rapid Refresh. (Further information: https://ruc.noaa.gov/hrrr/)

6. **WRF** - The Weather Research and Forecasting (WRF) Model is a next-generation mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting needs. (Further Information: http://www.wrf-model.org/index.php)

7. **RTMA** - The Real-Time Mesoscale Analysis (RTMA) is a NOAA/NCEP high-spatial and temporal resolution analysis/assimilation system for near-surface weather conditions.

This product takes NEXRAD and satellite data and combines them into a gridded format. (Further Information: http://nomads.ncep.noaa.gov/txt_descriptions/RTMA_doc.shtml)

8. **Domain** - The geographical area that a forecast or observation includes.

9. **Grib Format** - A gridded file format that is in use with most weather data systems.

Functional Requirements:

1. Get Data
    a. Two types of data will be downloaded from public sites
        i. Observations
            1. MRMS
            2. RTMA
        ii. Forecasts
            1. HRRR
            2. WRF runs by Dr. Baldwin
2. Convert Data
    a. Convert observation and forecast products to 1km grid spacing with the same domain.
    b. Store converted products for visualization and reanalysis
        i. Store converted observation and forecast products in gribV2 format.
        ii. Store 3 different variables (depending on sources): Temperature, Reflectivity and Humidity
3. Analyze Data
    a. Analyze difference using statistical methods
    b. Use specific statistical methods based on literature review with Dr. Baldwin
4. Visualize Data
    a. Visualize Observations, Forecasts and Derivative Analyses
        i. Static heat maps
        ii. Moving heat maps (over time)
        iii. Graphs/Charts of specific historical trends
5. Generate Website
    a. Website will be auto generated by WPEAR
    b. It will be organized by Year, Date, Hour and Variable
    c. It will be exclusively HTML and CSS and will not include any server side webpage processing

II. Design outline:

1. Hourly Run
    a. Controler
        i. Decides what data needs to be downloaded, converted, analyzed and visualized based
        ii. Make decisions based on the current time of day as well as the what files are currently available on the filesystem
        iii. Responsible for running all tasks below
    b. Download data
        i. Use python web tools to programmatically grab grib2 files from forecast and observation data sources
    c. Convert data
        i. Use grib2 libraries to load downloaded data
        ii. Reduce domain to size of the NWS Indianapolis Office Area
            1. Map of area here: http://www.stormready.noaa.gov/stormmaps/in-cwa.png
        iii. Remove all variable data except for Temperature, Reflectivity and Humidity
        iv. Interpolate grid spacing to 1km
        v. Use grib2 libraries to write out grib files with changed data
    d. Analyze data
        i. Use converted files for observation and forecast
        ii. Analyze using statistical methods defined with Dr. Baldwin
        iii. Create object with the analysis that can be read by the visualizer
    e. Visualize data
        i. Visualizer will take in objects from the analyze step as well as use objects from the forecast and observation converted files
        ii. Create visualizations for each run (hour), each forecast, observation and analysis, for each variable.
        iii. Create moving visualizations over time (past day, past week) for each forecast, observation and analysis, for each variable.
        iv. Create graphs for macro historical day
    f. Generate Website
        i. Generate static html for the visualizations
        ii. Include links to the converted files associated with the visualizations

iii. Create time-hierarchical webpages for visualizations (year, month, day, hour-variable)

III. Design issues: (To-do)

Functional Issues:

Issue 1: Best way to store conversions of observations and forecasts

Option 1: Using a database

Option 2: Storing files in a local or remote file system

Decision:

Option 2 is chosen because databases can be unreliable and have expensive overhead costs when storing and retrieving files frequently.

Issue 2: How to organize stored conversion files? (not for the past few hours but as a permanent store)

Option 1: Storing all files in the same directory with all metadata in its filename (like the way we would be storing raw grib files)

Option 2: Storing files in directories and subdirectories in the form <year>/<month>/<day>/<hour+variable> and some metadata in its file name

Decision:

Option 2 is chosen because this will act as a permanent store for conversion files and would become massive in the long-run. This would make searching using only metadata in the filename very expensive as compared with doing it in a single subdirectory (smaller subset).

Issue 3: How to organize the workflow of the whole program well?

Option 1: Begin with the DataDownloader as the origin of the whole process and keep the program running consistently by calling other classes

Option 2: Create a controller named WEAPERController as a task manager to supervise the overall workflow

Decision:

We agree on creating a dedicated class as controller to control the task assignment of each component to assure the smooth process and interactions between different components. In this way, every component does its own job and there's no extra workload of connecting to other components. Thus option 2 is chosen.

Issue 4: Appropriate file format for our stored data

Option 1: Use simple text-based format

Option 2: Use GRIB2 file format (the most updated version of GRIB format)

Decision:

Option 2 is chosen because first of all, GRIB is a concise data format commonly used in meteorology to store historical and forecast weather data. After we extract the target data from the collected raw data which is also in format of GRIB file, we would store them permanently on server side for future historical data analysis. Thus GRIB2 would be an ideal file format for its storage efficiency.

Issue 5: What visualizations to display when the program is run after a considerable amount of time?

Option 1: Display the visualizations of the latest hour based on the computer clock.

Option 2: Keep track of the last displayed visualization and restart from there.

Decision:

Option 2 is chosen because it allows the user to see a trend on the visualizations rather than seeing the visualization only for the last hour. We also plan to have a historical visualization option that we will work on in Sprint 2 which might affect this decision.

Non-Functional Issues:

Issue 6: How attractive and user-friendly should the website be?

Option 1: simple website written in html (and some css)

Option 2: attractive and user-friendly website using a web framework

Decision:

Option 1 is chosen because visualization and conversion would be more useful for the users of our project and it would waste less developer-time in learning/using web frameworks.

Issue 7: How to handle errors generated during data analysis?

Option 1: ignore the errors and produce results from the working parts of the analysis

Option 2: terminate the entire analysis and produce error report

Decision:

Option 2 is chosen because we will re-run analyses in the future that are missing. If the error is transient it will be fixed automatically if, some administrative interaction is needed once that happens all the missing analyses will be re-run. So in the end no work is lost.

IV. Design details:

Class design:

1. **WPEARController:** This will be the main control of the program and will be responsible for running the program based on command line flags. It will call different parts of the program to perform their individual roles and determine when each part is being called.

2. **DataDownloader:** This class will be responsible for downloading GRIB2 files for the forecasts and observations from the RTMA Products and HRRR sources. The files will be downloaded to memory.

3. **Conversions:** This class will extract the selected domain and the selected variable(s) from the downloaded GRIB2 files and produce smaller GRIB2-format files - ‘conversions’. It would also be responsible for storing conversions to and loading them from the file system. GRIB2-related file access that use the GRIB-2 library calls would be used.

4. **ConversionSuite:** This class will perform calculations on conversion(s) and produce results in the form of new conversion(s). These calculations would include using different statistical methods and mathematical techniques for which they would leverage some known Python libraries/tools.

5. **Visualizer:** This class will generate visualizations of conversions. These visualizations can be static or dynamic depending upon the timeline of the

conversions used or the type of visualization needed. Once drawn, the visualization can be displayed/stored.

6. **WebsiteManager:** This class will generate a simple website that can be used to view hourly/daily visualizations and conversions once the system is automated and running regularly. This class will also handle the format of the website.

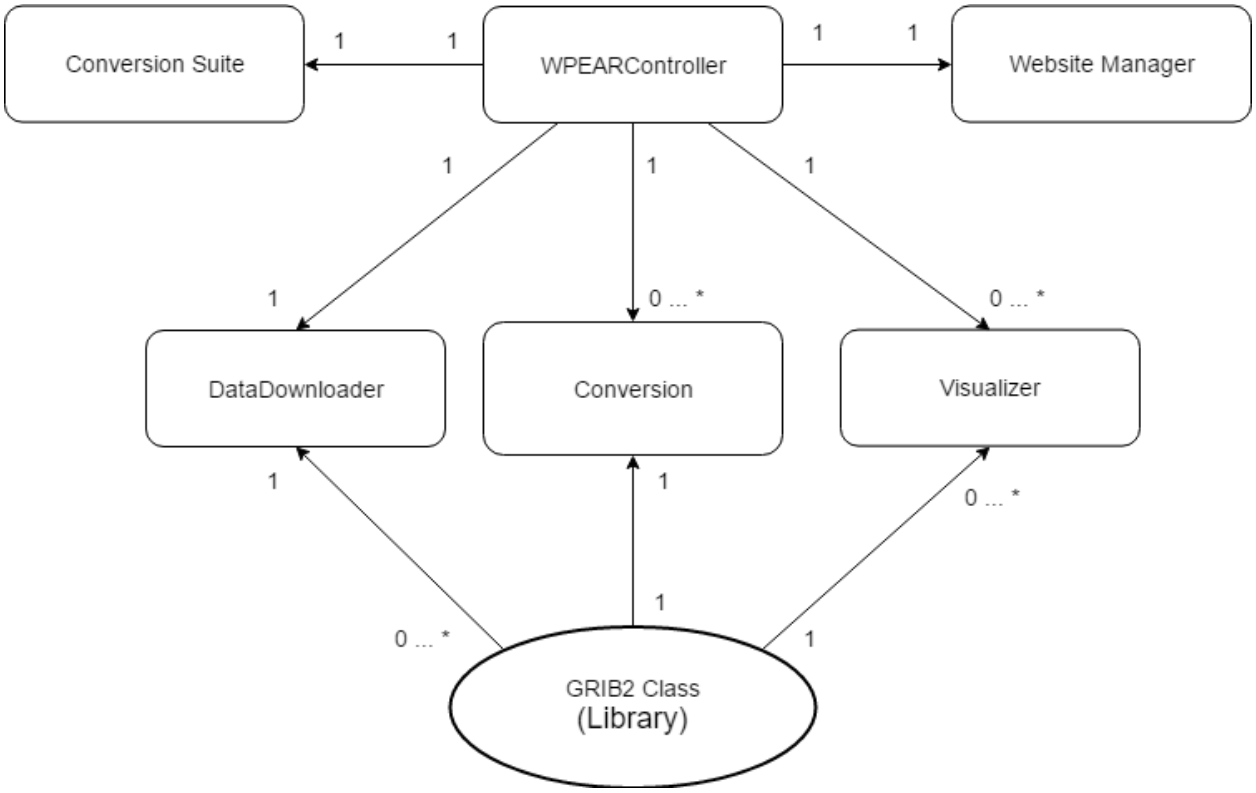

Figure 1.1 : Class Diagram: illustrates the relationship between the different classes and the flow of data and object instances as the pipeline functions.

Sequence diagrams:

1. Working flow diagram

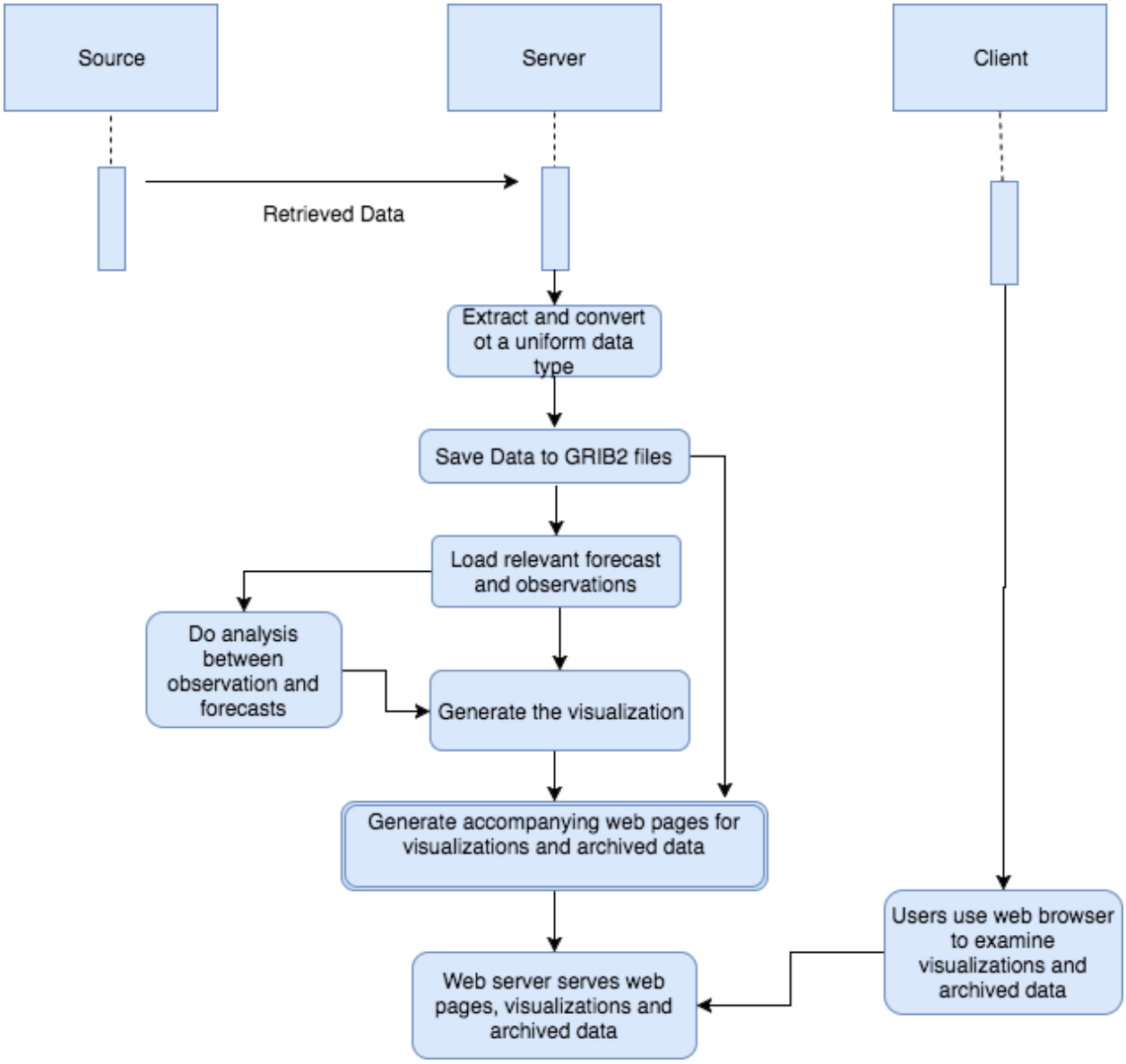

Figure 1.2: illustrates the working flow inside the program. For the backend, the server keeps retrieving relevant forecast and observation data from the given sources hourly. Then the application will extract and uniform the target data and save them in the grib2 file format then the system will load those files, analyze them and visualize the observation, forecast as well as the analysis. Then all visualizations and data will be used to generate web pages and users will use a web browser to browse these visualizations.