Honours Thesis

University of Cape Town
Department of Computer Science

Computer Science Honours
2008

Weather Forecasting Using Dynamic Bayesian Networks

Weather Visualization System

Pascal Haingura

Supervised By:
Dr. Hanh Le
Dr. Mark Tadross
Dr. Anet Potgieter

Abstract

This paper presents an investigation on techniques and proven methods for representing weather information, considered as benchmarks for a weather visualization system developed as part of weather forecasting project.

Three weather variables in maximum temperature, minimum temperature and precipitation were considered for the project. South Africa is the region of focus with all the project data coming from weather points from around the country.

The study showed that weather visualization is quite a complex problem and making the visualizations more understandable to users is even harder, thus applying usability on weather visualizing applications is fairly challenging.

Several contour plotting algorithms were considered, it became clear that most algorithms are data dependent i.e. could only work well for that formatted data.
Contour maps were chosen as the form of weather visualization to be used in the project.

In the end a web based Java application was built, and integrated in it were the contour plotting sub routines, together forming a weather visualization interface.

Usability tests were done on the application. Users were divided into two groups i.e. users will knowledge in meteorology as expert users, and no non-expert users. Results from the usability tests are presented and the usability questionnaires appended.

Key words: Weather Visualization, Contour, Contour Plotting, Weather forecasting

Acknowledgements

The author would like to thank everyone who directly or indirectly contributed to the project. Firstly the project supervisors, Thanks to Dr Hanh Le, her dedication to the project and ongoing eagerness to see us improve made all the difference. Dr Anet Potgieter, for the countless mind-opening advises and notable input to the project. Dr. Mark Tadross, for the expert figure and priceless expert input.

Secondly, the EGS department staff members Chris Lennard and Chris Jack, who provided us with the data we used in the project. It's rare to find free data these days and their kindness made things a lot easier.

My project partners Matthew de Kock and Michael Kent for their inspiration work ethic and communication and support.

And finally, Dr. Hussein Suleman for his 'open door policy', always welcoming and ready to offer technical help.

Table of Contents

List of Figures

List of Tables

1. Project Introduction

1.1. Introduction

Weather prediction and visualization is a difficult problem. As with most complex data, weather data requires a considerable amount of structuring and reasonable visualization to become user friendly.
One of the reasons that weather forecasting is considered a difficult problem is because it deals with volumetric multi-field data [1].

In South Africa, just like in many regions around the world, changes in weather conditions are not a slow process. With the weather being likely to change in a matter of hours if not minutes, people who rely on weather information highly appreciate sound and accurate weather forecasts.
As weather depends on a multitude factors such as heat energy, moisture in the atmosphere, wind, and air pressure, weather forecasting therefore requires complex and time-consuming calculations taking into consideration all these factors. [3]
On top of these, the resulting forecast data is hard to read for many weather users, the non-experts especially. A lot of effort is needed to design systems that can represent these data in formats that users can understand; this is what weather visualization generally is all about – representing weather information in such a way that it becomes understandable for human use.

Figures 1 and 2 below illustrate two ways to represent weather information in a human understandable form. Figure 1 is an example of a contour map, while figure 2 is a classic form of representing weather information; it is the type many weather service TV stations across the world use.

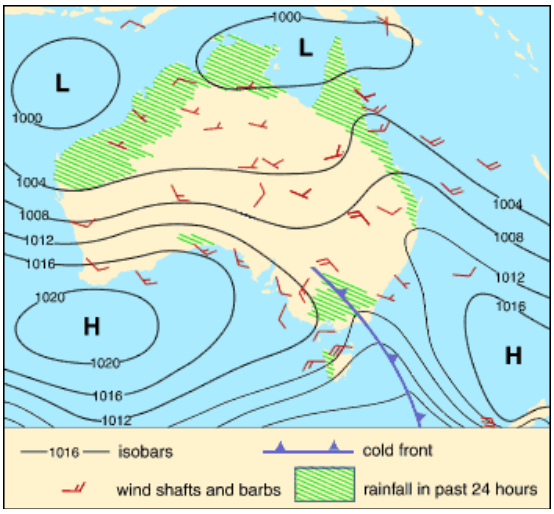

Figure 1: Representing Weather Information using Contours Plots information

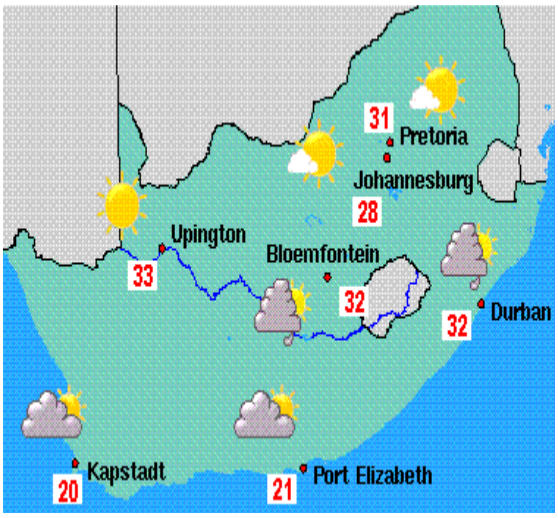

Figure 2: A classic way of representing weather

1.2. Project Aims

The primary aim of this project was to implement a weather model that would produce forecasts using Bayesian Networks. The model was divided in three different sections, a Causal Model, a Dynamic Bayesian Network, and a Visualization tool. The Causal model was used to define the structure and node dependencies of the Bayesian network. This structure(s) were then used to construct a Dynamic Bayesian Network (DBN), The Visualization tool to graphically represent, through a web interface, the produced data in a human understandable form.

The initial scope of the project was restricted to forecasting average temperature, based on past data and predictions. But this was changed as we (team members) learned more about the field, in the end it was decided that we would forecast minimum, maximum temperature and precipitation. The visualization tool was designed to represent all these three weather information variables.

1.3. Project Team

The project team included partners Michael, Matthew de Kock, and myself (Pascal Haingura) each responsible for one of the three sections that make up the whole project i.e. The Casual Model, The Dynamic Bayesian Networks and The Visualization system respectively.

The Causal Model
This section is concerned with building static structures and inter-station (node) dependencies of the Bayesian Network. These structures (networks) are then prepared for the next section, The Dynamic Bayesian Networks. This section was done by **Michael Kent**

The Dynamic Bayesian Networks
This section, developed by **Matthew de Kock**, involves constructing the final Dynamic Bayesian Networks based on the static structures and the dependencies developed in *The Causal Model* section. Output from this section is the weather forecasts, which will be used in the visualization system – to graphically represent the data in a user friendly form.

The Visualization Section
In this section the system to represent weather information in a form understandable to users (non-experts as well) will be developed. Output from this section will be graphical representations of the weather data on a java application interface through a web browser. Usability will be considered in this section, as this would be important for users to be able to translate the information represented on the interface(s). This section was developed by **Pascal Haingura**.

Figure 3 summarizes the project task allocated to team members.

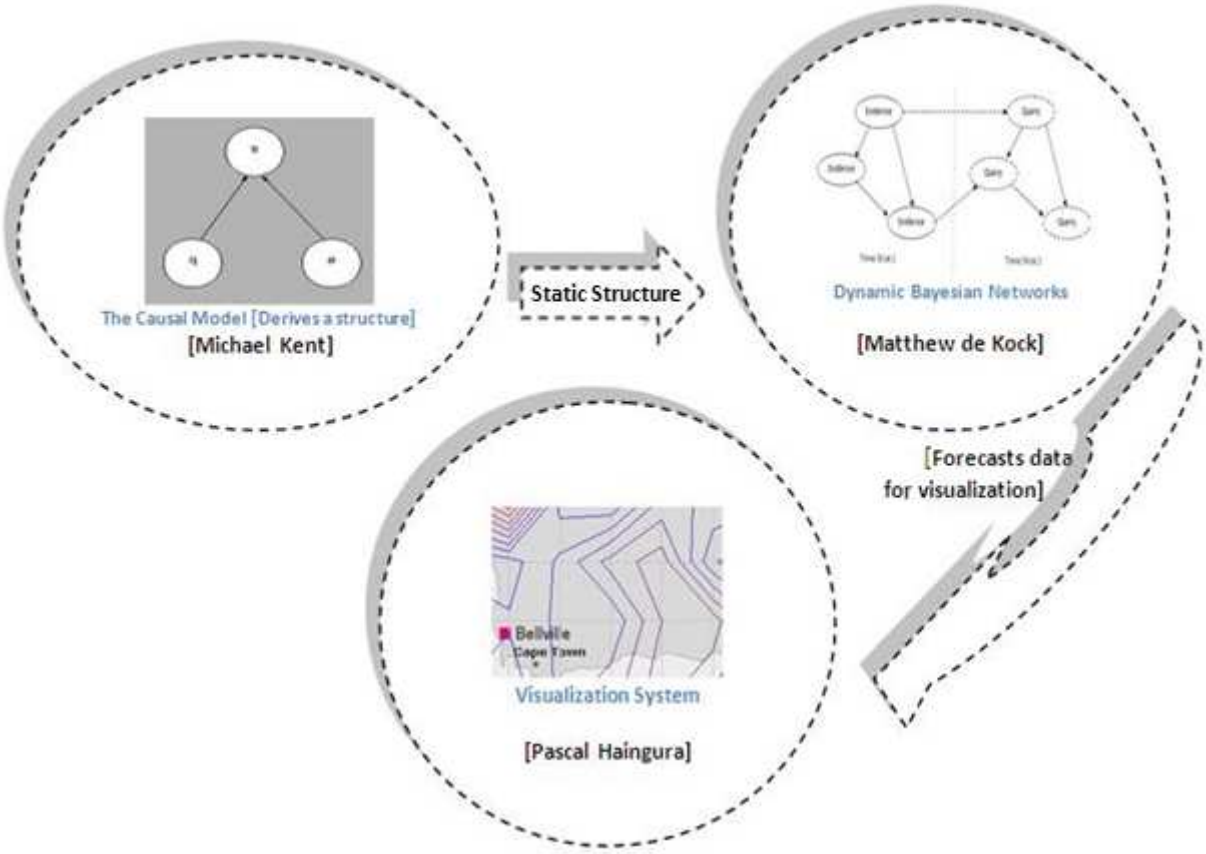

Figure 3: Project tasks breakdown

1.4. Overview of Report

This document discusses the procedures and considerations followed to implement the weather visualization system, and all other relevant factors or information across the field of weather data visualization that we came across during the course of the project, an attempt was made to give the reader a view of the whole project – though much of the emphasis were on the Visualization tool.

The report is divided into eight major sections, including the ***Introduction*** section.

The second section from the introduction is the ***Background***; this section discusses the relevant background research related to the project, more emphasis on the *Visualization* section.
The ***Design*** section, following the *background*, presents the design decisions that we taken into considerations to see through the implementation of the Visualization system. Overall, it presents an abstract view of how the system was implemented and also gives readers a generic picture of the complete system.

Following the *design* chapter is the ***Implementation*** section; it covers a discussion of how the design specifications were realized. It communicates the procedures that were taken to meet the system specifications. An outline of tools used to design the system is also discussed here.
The ***Evaluation*** section follows the *implementation* stage. System testing considerations are discussed in this section. A test plan and testing strategies are presented. The section is concluded with findings or outcomes of the tests that were done on the system.

The ***Conclusions*** section wraps up the main section of the project report. This section starts with a discussion on whether the system meets it's specifications before a consideration on possible future work and recommendations.

The ***References*** section, which comes just before the ***Appendices*** chapter, presents a list of references, as cited in the document.
The last section of the report is the **Appendices**; it is for any additional or supporting material used in the project, or in this report.

2. Background

Weather data can be graphically represented in forms understandable to users in a variety of ways. The two commons forms include *weather maps* and *contour maps*.
Weather maps (as shown in *Figure 2*) is an example where weather information (temperature, rainfall, wind direction etc.) is displayed on a map, usually on points representing major cities, and is aided by static images representing the weather conditions predicted for the respective places.

Contour maps on the other hand (*Figure 1 above*), the primary focus of the system this report is based on.
Contour maps are a type of weather maps illustrated with contour lines, i.e. the map is filled with curves connecting points with equal values. Forms of contour lines include *isotherms* (see *figure 4* below), which are contour lines connecting points of equal temperature; *isohyets,* contour lines connecting points of equal precipitation and *isogons*, which are lines along which the wind direction, expressed as an angle, is everywhere the same. [1]

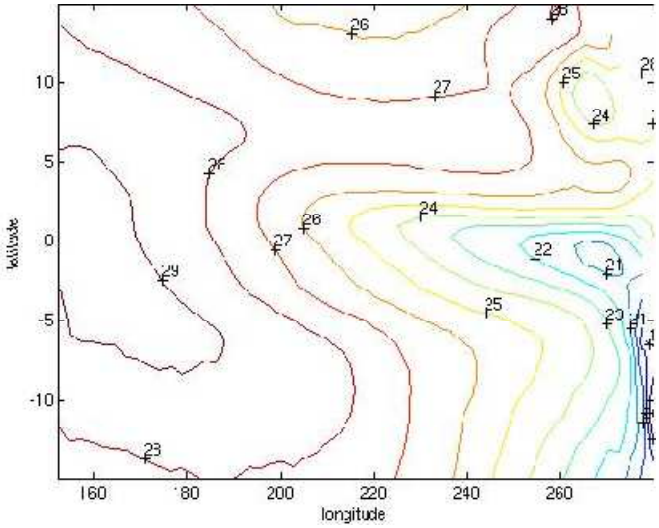

Figure 4: Isotherms, contour lines joining points of equal temperature

Other forms of weather visualization exist; rainfall for example, can be represented in 3D storm graphics. [4]
Another form is described in [12], where they made a kind of 3D satellite pictures that present the weather described by the grid points in an atmospheric model, i.e., the weather of tomorrow.

2.1. Weather Forecasting

Traditionally, statistics have been used in the field of weather forecasting.
A host of other previous studies have established statistical relationships between the severity of environmental conditions, weather reports and gridded short-term numerical forecasts.
For example, the statistical theory of extreme values can be used in the analysis of weather extremes and their impacts. [14]

Numerical methods are also used in weather forecasting. Numerical weather prediction [13] uses current weather conditions as input into mathematical models of the atmosphere to predict the weather.
Numerical weather models [16] are initialized using observed data from surface weather observations or weather satellites.
These data, which usually is irregularly-spaced is processed by [13] *data assimilation* and some data analysis methods, all with a common goal of converting the data to a format usable by the model's mathematical algorithms.
The algorithms then use these processed and analyzed data to predict the state of the weather into the future.

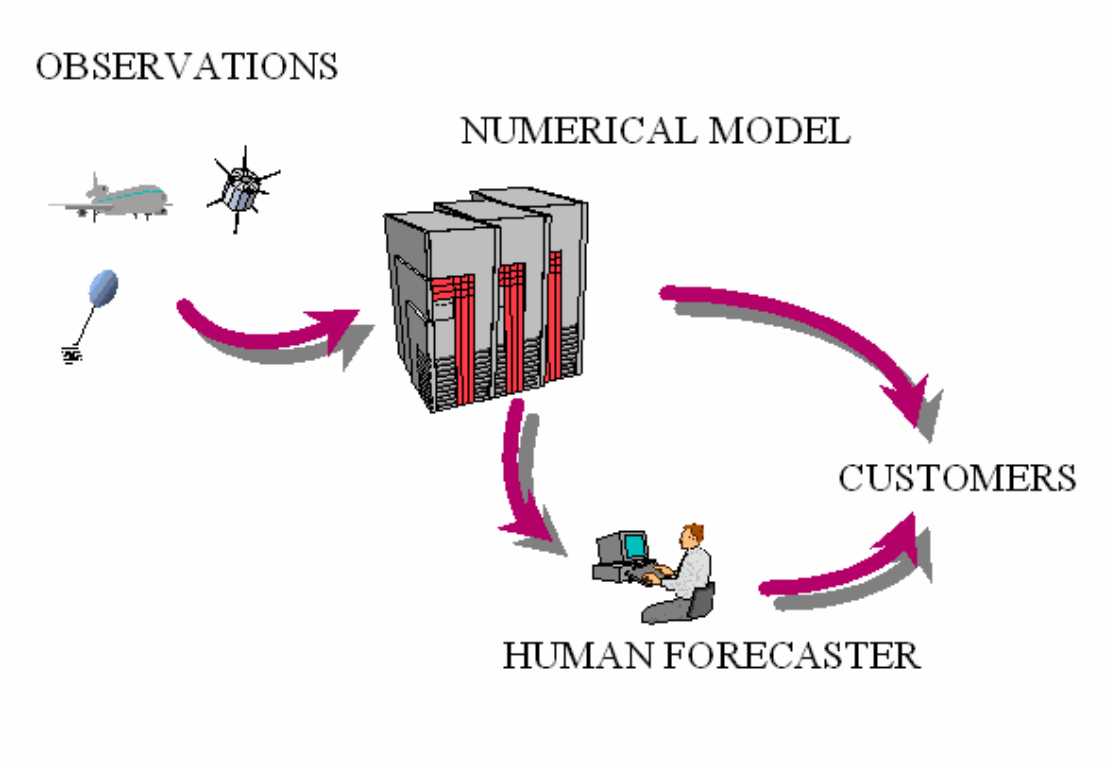

Figure 5: An example Numerical Method from UK Met Office showing the process of producing a weather forecast.

In *figure 5* is showing a process where international projects of various parameters such as temperature, wind, humidity etc, of the atmosphere are made and these are fed into a

numerical model of the atmosphere to produce an objective forecast. This forecast is then studied by a human forecaster, who may correct errors or add extra detail, before a forecast is issued to the customer. [13]

Several numerical weather prediction models and methods exist; [16]Ensemble Kalman Filtering is an example of the latter, it basically is a sequential Monte Carlo method commonly used in meteorology to track atmospheric states and make numerical weather predictions.

Examples of models include the Global Forecast System [18], which basically is a numerical weather prediction model that produces weather forecasts for up to 16 days. It is freely available is used on online weather service stations and many institutions meteorological departments.

Another numerical weather prediction model is [19] The Unified Model, which basically is a set of several numerical weather prediction models. It is used by the UK Met Office and it includes the main suite of a Global Model - a UK and North Atlantic model, in addition to a series of other models. The models are grid based i.e. they use gridded data.

Numerical weather prediction systems, however, have several drawbacks. For example to compute large datasets and perform the complex calculations required when dealing with weather data requires the use of powerful supercomputers. [16]
This is a computational feasibility problem; this basically means that without supercomputers numerical weather prediction models become 'unusable'.

This project uses Bayesian Networks as they can model the spatial and temporal dependencies among the different stations using a directed acyclic graph. This graph is learnt from the available datasets and allows deriving a probabilistic model consistent with all the available information. [20]

2.2. Weather Visualization

Weather visualization is a concept that involves making weather information understandable to users.

Implementing the graphical representation of weather forecasts can be done with almost [2] any graphics development tool. But choosing which to use is very important, this is because it will determine how much effort a developer will have to put in to implement the system.

Several weather visualization systems exist. Commonly, [8] weather visualizations consist of a set of maps showing isolines and isotherms representing temperature, air-pressure, and precipitation.

2.2.1. Weather Visualization Systems

Several weather visualization systems exist. Some systems are 'complete packages' i.e. they do both the forecasting and visualization, while others either use external data to generate visualizations (SVG discussed below) or just do forecasting.

The rest of this section presents reviews of some famous weather forecasting and visualization systems out there.

2.2.1.1. The Weather Processor (WXP)

The Weather Processor (WXP) can be briefly described as a suite of programs for analyzing and representing meteorological data. It is intended to be a general-purpose weather visualization tool. [21]

The system gets its data from the National Weather Service. It decodes and parses the data, before finally generating graphic representations for visualization. The visualizations come in varying degrees of complexity, with meteorologists as its primary audience. [21]

The Weather Processor (WXP) can be logically divided into three data processing sections i.e. ***ingest***, ***decoding*** and ***analysis***, as *figure 6* below illustrates.

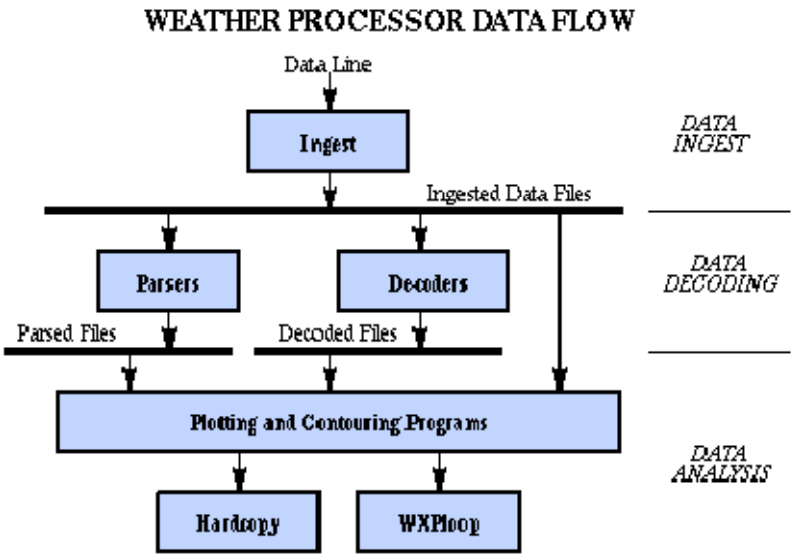

Figure 6: The Weather Processor Data Flow

Data input takes place in the *ingest stage*, and then after is the *decoding stage;* this stage involves parsing and decoding of the data from the *ingest* stage.

Analysis is the last stage of the process, it is the where final analysis of the processed data takes place before display. This is the stage where the visualization algorithms are, these includes the contouring and plotting programs.

WXP can be used to view and analyze various weather data (e.g. satellite imagery and air-data). It also includes routines for Standard Meteorological Analysis, e.g. routines that can plot upper air parameters for a specific level or layer.

2.2.1.2. SVG Weather

SVG Weather [7] is a weather visualization system developed using Scalable Vector Graphics (SVG). It uses an SVG frontend to produce an interactive visualization of weather forecasts in different zoom levels. To realize this, a range of techniques is required on both server and client side.

The overall structure of the application as illustrated in *figure 7* comprises of the following components:

- An interactive SVG front-end for visualization and dynamic reload of the forecast data
- A number of PHP pages on the server for communication from front-end to server
- A database in which the weather forecast data is stored
- Scripts and applications for updating and pre-processing weather forecast data in the database

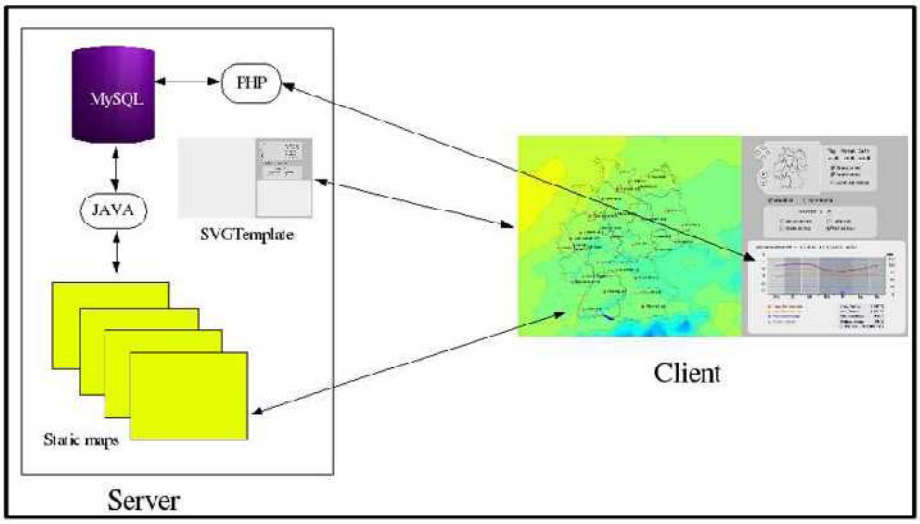

Figure 7: The SVG Weather Application Design

Scalable Vector Graphics, with which SVG Weather was designed, is a language or specification for portraying two-dimensional graphics and graphical applications in XML, both static and animated. It was created by the W3C's SVG Working Group. Many other applications have been designed with Scalable Vector Graphics. [7]

2.2.1.3. Digital Atmosphere

Digital Atmosphere [22] is a powerful weather forecasting program that allows creating detailed maps of real-time weather for anywhere in the world. It makes extensive use of weather visualization techniques and algorithms that are comparable to some of the best. It uses free data from government and university sources on the Internet.

Digital Atmosphere supports all sorts of data e.g. land synoptic, aircraft data report, gridded binary, Australian AXF comma-delimited observational data, and radar images. Some of the many functions and features of Digital Atmosphere includes drawing detailed contour maps, and has a host of import capabilities – it can import all kinds of data format, and it's ability to generate meteorological data analysis graphs.

2.2.1.4. Storm Predator

StormPredator is a second generation weather forecasting and visualization program developed and is maintained by the IntelliWeather. [23]
This system monitors storms and produces the resulting feedback through its interface.

StormPredator was developed around several requirements such as ease of use by the meteorologist as well as ordinary users without special training, significant storm alerting and tracking tools, and an easy to use distance measuring tool to determine how far away storms are. The storm alerting and tracking tools includes ETA (estimated time of arrival) of the storm calculations [23]

The Local Weather Model visualization System, which this paper focuses on, may have fewer features and capabilities, but the overall process is no difference

Relate it to *The Weather Processor* for example*;* unlike the weather processor that gets it's data from external sources, produces it's own data, which it parses, and restructure before using it is used in the learning and prediction algorithms.
The output from these algorithms is then used in the visualization section, where it is again restructured to compliment with the data input requirements of the visualization algorithms.

2.2.1.5. Comparison

Here is a summary of the systems we discussed above in comparison to the *local weather model visualization system*.

System	Core Function(s)	Primary Source of data for visualizations	Other Functions
The Weather Processor	Weather Visualization	National Weather Service	[21] Also includes routines for Standard Meteorological Analysis
SVG Weather	Weather Visualization	Elsewhere	Geographical Information [7]
Digital Atmosphere	Weather Visualization, and Analysis	Uses free data from government and university sources and other institutions	None specified
Storm Predator	Weather prediction and visualization	Uses observed data to predict and use the predicted data for visualization	None specified
The Local Weather Model	Prediction and Visualization	Data for predictions from The UCT Environmental Sciences department and the output to this used for producing visualizations	None

Table 1: A comparison of the visualization systems presented in background chapter and that of the project

2.2.2. Weather Service Sites

Weather service sites are basically websites that provide weather information as a service Many weather service sites exist on the web. They range from regional weather service sites, provide forecasts for the nearby community, to global weather service sites. Two weather sites; WeatherBug and Weather Underground will be discussed in this section.

2.2.2.1. Weather Bug

WeatherBug, developed by AWS Convergence Technologies, Inc [24] is a weather service site that provides weather updates for American cities. It is a standalone site i.e. it doesn't use weather data from external sources. The site uses the AWS' own WeatherBug application to produce predictions for the site.

Among other forms of weather visualizations, WeatherBug produces live isotherms and isohyets of the temperature and rainfall forecasts across the USA. [24]

2.2.2.2. Weather Underground

Weather Underground is a commercial weather service site that provides real-time weather information. Weather Underground, based in Ann Arbor, Michigan, provides weather reports for most major cities across the world, as well as local (USA) weather reports for newspapers and Web sites. Most of its United States data comes from the National Weather Service (NWS).
The Web service site is available in many languages.

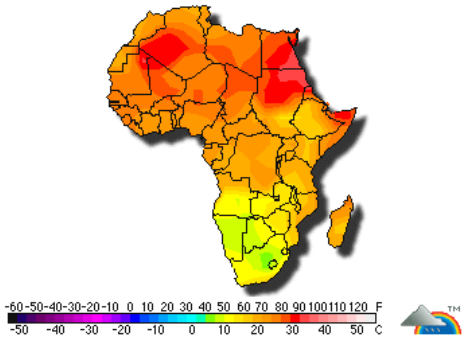

Figure 8: Live Africa Temperature from the weather underground site

The fact that Weather Underground provides weather information from all over the world (*as figure 8 shows Africa as an example)* makes it the most broad and useful between these two weather sites.
The Local Weather Model Visualization system's weather service site will be similar to the ones discussed above, except it will have a narrower coverage, providing weather information for selected points in South Africa.

2.3. Techniques

2.3.1. Multi-Field Weather Visualization

Multi-Field Weather Visualization is a technique presented by Kirk Riley, David Ebert, Charles Hansen, and Jason Levit in the paper *Visually Accurate Multi-Field Weather Visualization.* [4]

It starts with the system rendering weather particle fields based on the particle properties. To visualize these particles in an accurate manner, the input data fields are translated into particle concentrations.
Volume rendering techniques are then used to represent a continuous field through transfer function mapping of data values to colors and opacities that correctly reveal the spatial relationships between structures (e.g. thick, thin, opaque etc…); therefore, it is necessary that

they use volumetric techniques to be able to render multi-field data in a visually accurate manner.

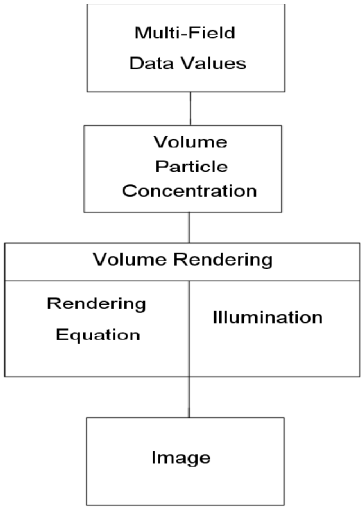

Figure 9: A Multi-Field Weather Visualization System Flow

Generally this is a very good visualization technique, especially when representing precipitation.

2.3.2. Contouring

Contours are lines or curves joining points of equal value. Contours are not only used in meteorology, but in many other fields as well.
In geography for example, they are used to demonstrate elevation of points above see level where points of equal will be connected by the same contour line. [9]
And in science, physics especially, contours can be used do demonstrate magnetic fields. [10]

- **Contours in Meteorology**

In meteorology, contour lines are based [26] on generalization from point data received from weather stations. The fact is that weather stations are hardly exactly positioned at a contour line, unless their reading precisely equals the value of the contour. Instead, [26] contour lines are drawn to best approximate the locations of exact values, based on the gridded information available.
Weather contour lines include isotherms (join points of equal temperature), isogons (join points where the wind direction is everywhere the same) and isohyets (join points of equal precipitation).

2.4. Usability of Weather Visualization Systems

Designing systems to perform certain tasks is one thing, but making them to be usable is another.
[27] Fayish, A.C and Jovanis, P.P did a study on web based weather information systems for travelers. Where a number of travelers from several states across the USA were asked to visit web-based roadway weather information systems as part of a laboratory-based trip-planning experiment and provided ratings of site attributes.

Site ratings revealed that [27] it was important that travelers accessed information about conditions along their intended route with minimal searching. Map scale and resolution were important graphical elements. With precipitation, road surface condition, visibility, and air temperature being the most relevant weather information identified by users.
This information was useful to designers of web based weather information systems, since it provided them with a requirement platform.

The lesson here is that visualization interfaces should be clear and simple and should provide just enough of the information that users want the most.

3. Design

This section discusses the design specifications of the weather visualization system. This system is the user interface part of the Weather Forecasting Model project.

3.1. Requirements

In essence, the system should allow users to view contours for different daily weather variables i.e. minimum and maximum temperature, and precipitation on a *contour map*. They, on top this, should be able to view, separately, the forecasted data of these variables for the respective stations or weather points.

The system must also be able to produce contours in a time series form, with each set representing the forecasted data on that day for whichever weather variable. The system should give users the option to view the time series as animation or manually where they manually skip through the different sets.

After studying the problem description and the background chapter it became clear as to what problems and limitations we could face when implementing the system and all these will be highlighted in this section (*Design*).

3.2. Use Cases

There are several use cases for the systems, the main use cases include;

- A user using the interface to view daily forecasts.
- A user wants to view forecasts for a particular station
- The interface applet accessing weather data from a web server or a local client via a .NET URL connection.
- A user using the interface to view average and change in the forecasts over the forecast range.

The use case diagram (*figure 10)* below shows the general functionalities the system will offer

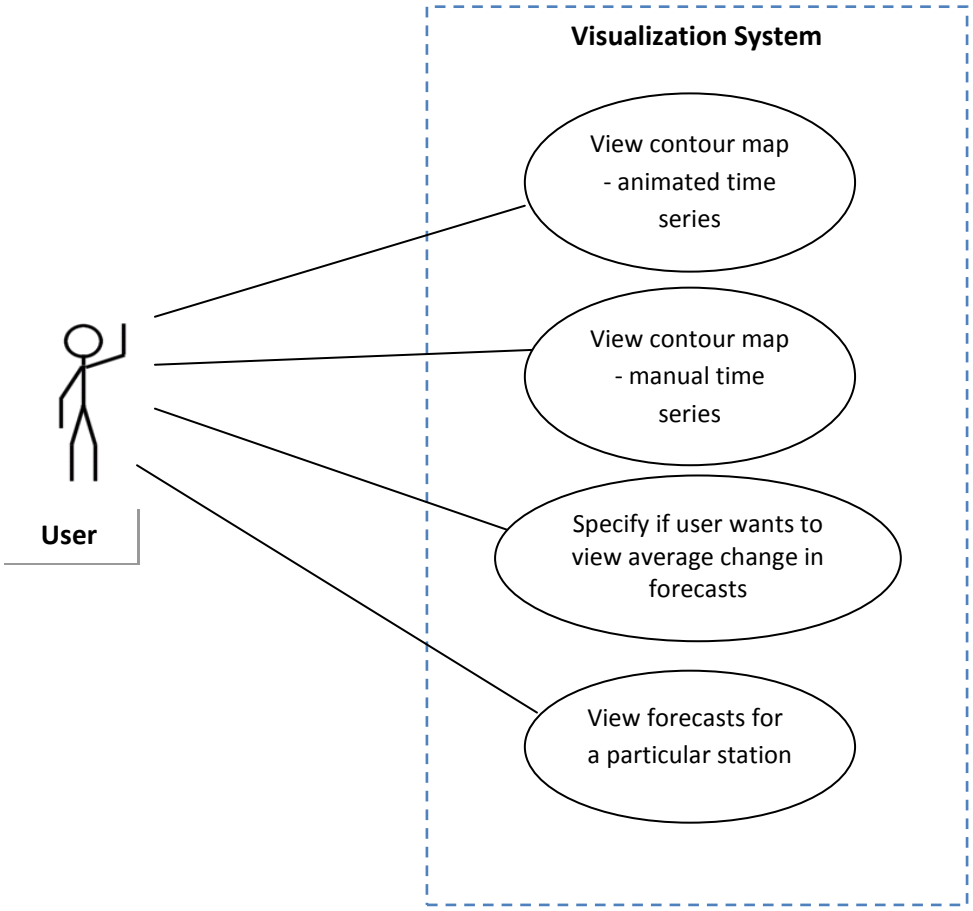

Figure 10: Use case diagram showing the key user options the system provides – showing the key functionality of the system

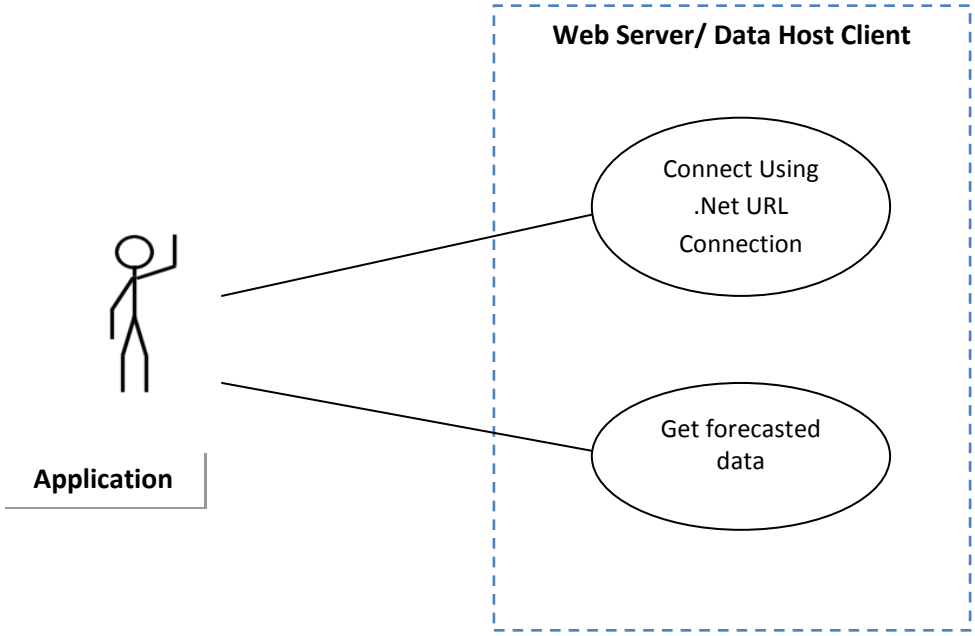

Figure 11: Use case diagram of how the application retrieves data from a data host

Below is a detailed discussion of the several use cases for the system.

3.2.1. Activate Animated Time Series

Description
The user wants to view contour maps of the different forecast days in a sequence by activating the animation functionality so the visuals can then be produced without further input from the user.

Sequence of Events

Basic Flow
User would activate the animated time series function.
The system would access the data for that forecast range e.g. Monday to Thursday.
The system would then display the *contour maps* for each of the days in a sequence starting from the first day to the last with a time delay of roughly five seconds.

Alternative Flows

- No forecasted data available, the system would inform the user of the problem i.e. there is no data to display.
- Connection to web server or client holding data crashes, the system would inform the user accordingly

Pre-Conditions
The Weather Visualization System should be running in a java enabled browser. Data sets must be available on web server or local client.

Post-Conditions
The system could start the time series, and would continue to display forecasts for the following days.
The system could inform the user that there is no data on the web server or the specified client or that it lost connection to data host.

The Sequence diagram for this use case is shown in the *figure 12* below.

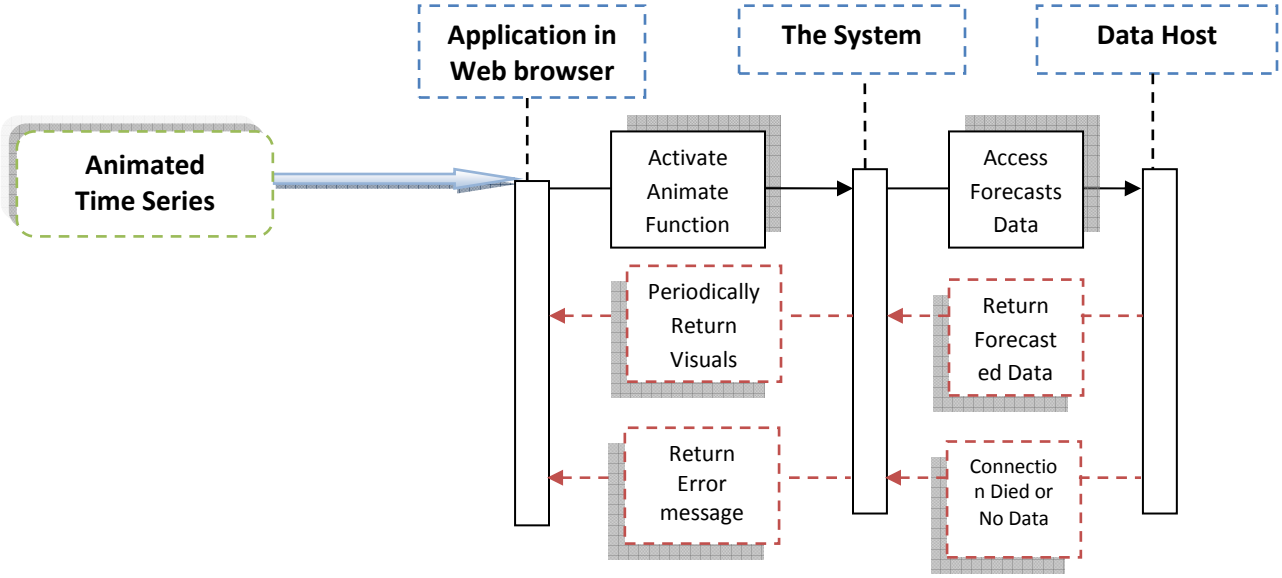

Figure 12: Sequence diagram - View Animated Time Series

3.2.2. View Forecasts for a Particular Station

Description

The user wants to view the weather forecasts (maximum temperature, minimum temperature and precipitation) for a particular station. He does so by selecting the station from a drop down menu and the data is displayed.

Sequence of Events

Basic Flow
User would select station he/ she would like to see forecasts for.
The system would access the forecast data for that station
The system would then display the *forecasts* for the station for each of the days in the forecast range.

Alternative Flows

- No forecast data available, the system would inform the user of the problem i.e. there is no data to display.
- Connection to web server or client holding data crashes, the system would inform the user accordingly

Pre-Conditions
The Weather Visualization System should be running in a java enabled browser and the forecasted data must be available on web server or data host client.

Post-Conditions
The system would then display the *forecasts* for the station for each of the days in the forecast range.
The system could inform the user that there is no data on the web server or the specified client or that it lost connection to data host.

The Sequence diagram for this use case is shown in *figure 13* below.

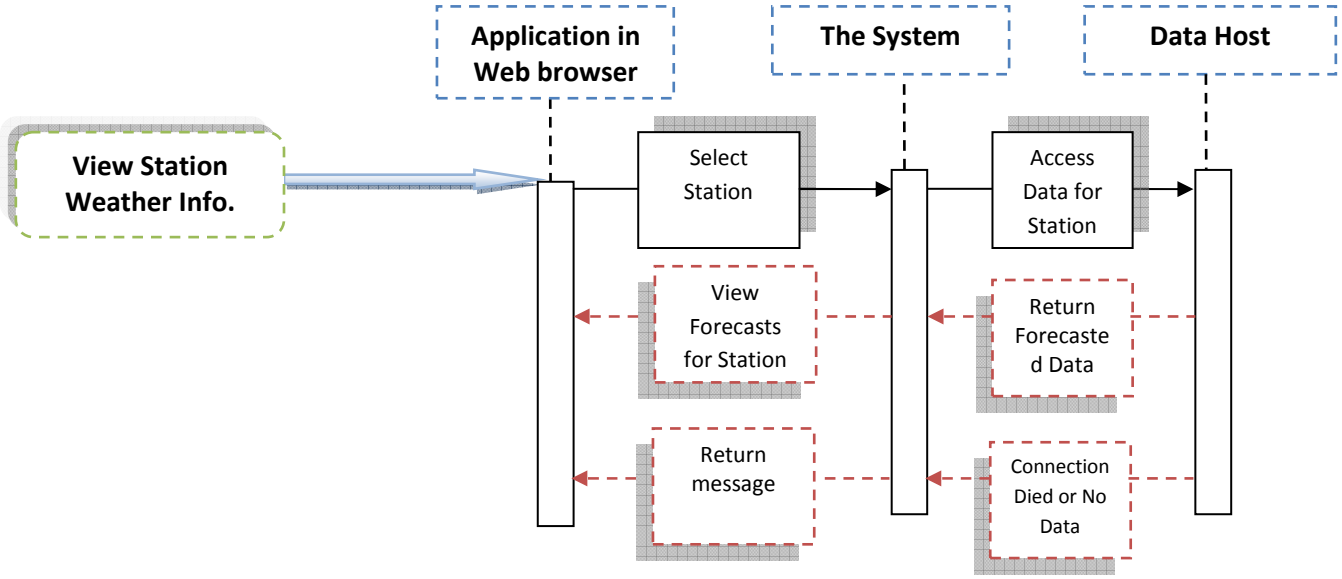

Figure 13: Sequence diagram – view forecasts for a specific station

3.2.3. View Time Series – Manual Time Steps

Description
The user wants to view contour maps of the different forecast days in a sequence by clicking on a button. The visuals are displayed in a sequence as the user clicks the 'next' button.

Sequence of Events

Basic Flow
User would start by clicking on a button for the current day's forecasts and will then manually go through the different days by clicking on a 'next day'.

The system would access the data for that forecast range e.g. Monday to Thursday.
The system would then display the *contour maps* for each of the days in a sequence starting from the first day to the last, only displaying a new contour map after the user clicks on the 'next day' button.

Alternative Flows

- No forecasted data available, the system would inform the user of the problem i.e. there is no data to display.
- Connection to web server or client holding data crashes, the system would inform the user accordingly

Pre-Conditions

The Weather Visualization System should be running in a java enabled browser and data sets must be available on web server or local client.

Post-Conditions

The system could display the contour map for the current day and would continue to do so for the following days.
The system could send an error message informing the user that there is no data on the data host or that it has lost connection to the data host.

The Sequence diagram for this use case is shown in *figure 14* below.

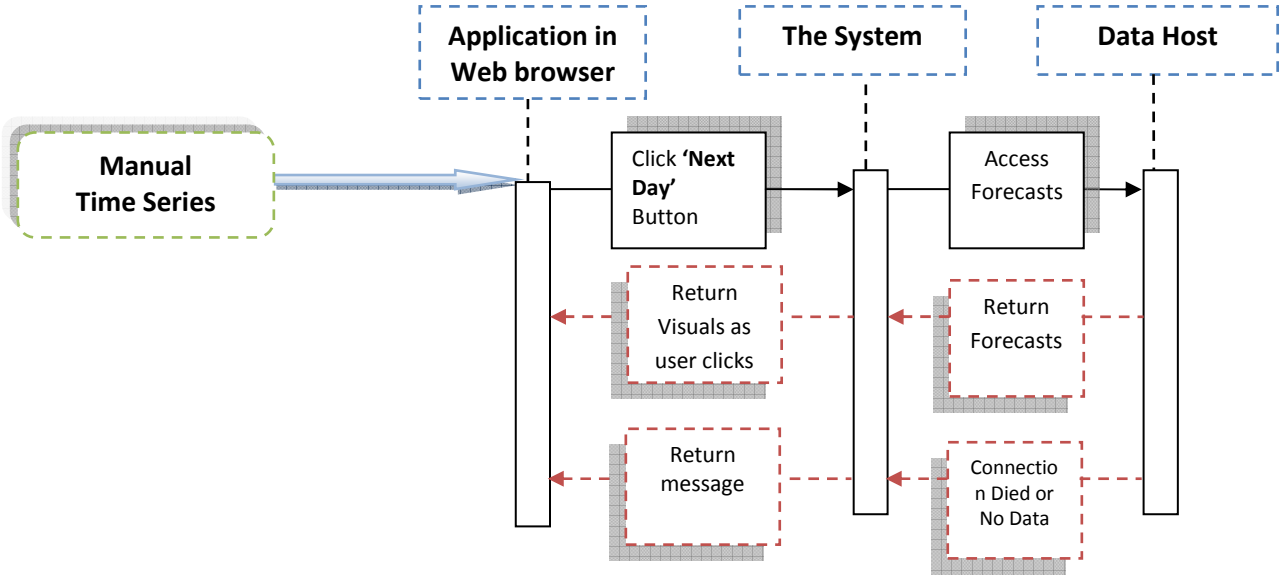

Figure 14: Sequence diagram - View Time Series, Manual Time Steps

3.3. System Architecture

The implementation of visualization system will be divided into three major components; *the data input and structuring* section, the *visualization algorithms (contouring drawing routines and routines to produce the interface*) section, and *interface and contour maps* section. Each of these sections will constitute as core entities.

Figure 15 below shows the architecture of the weather visualization system.

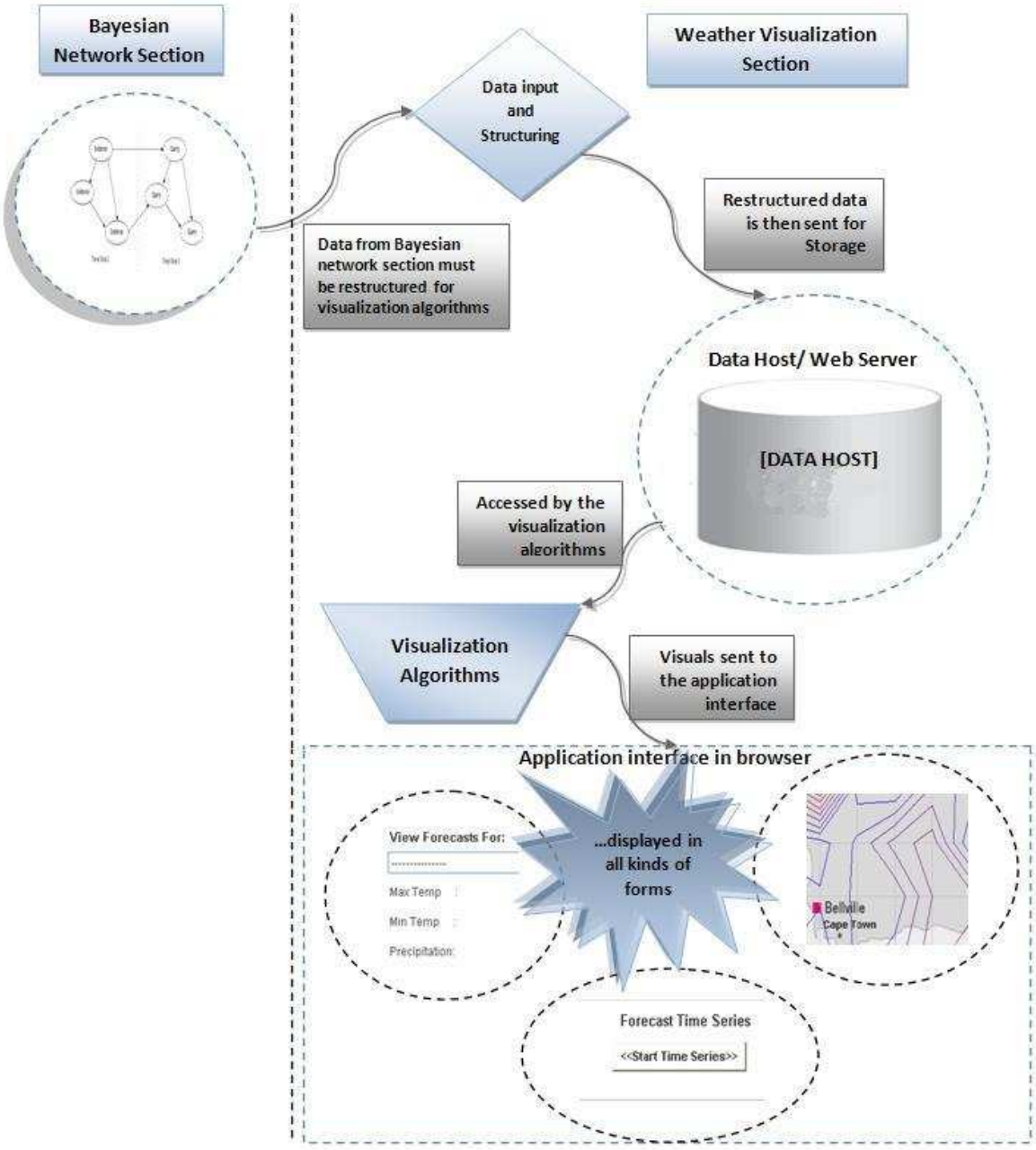

Figure 15: System Architecture for the weather visualization system

3.3.1. Data input and Structuring

This section will take in as input the forecasted data from the Bayesian networks section. This data will then be restructured in a format that would be fitting for the visualization algorithms, more detail about data structuring will be discussed in the implementation section.

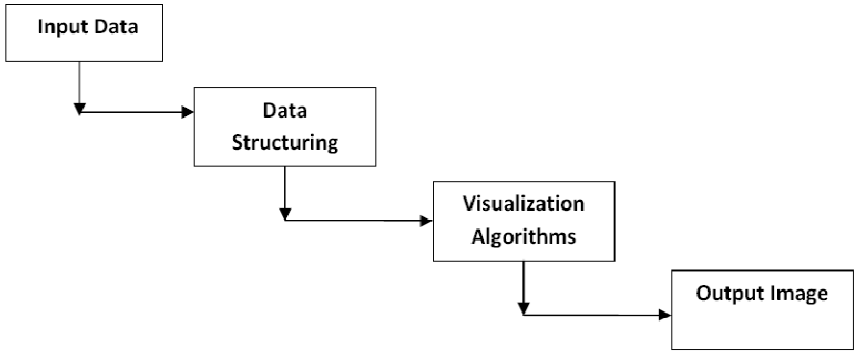

Figure 16: A general overview of the visualization system

3.3.2. Visualization Algorithms

This is the key section to producing the contours, which are then displayed on a map at the application interface. This section will comprise of contour data parsing and gridding routines which will primarily prepare the data for the contour plotting 'kernel'. Drawing the contours will be done in several procedures, a detailed discussion of the algorithm and its sub-procedures will be covered in the implementation section.

3.3.3. Application Interface

This is the front-end for the visualization section of the project. It is the user interface component of the project; it provides all the user options we considered under the *Use Cases* section in *section 3.2*

3.4. Class Diagrams

This section provides a conceptual illustration of the entities that would be used to implement the visualization system.

Figure 17 below shows a detailed version of the *class diagram* with major functions and variables in the respective entities listed.

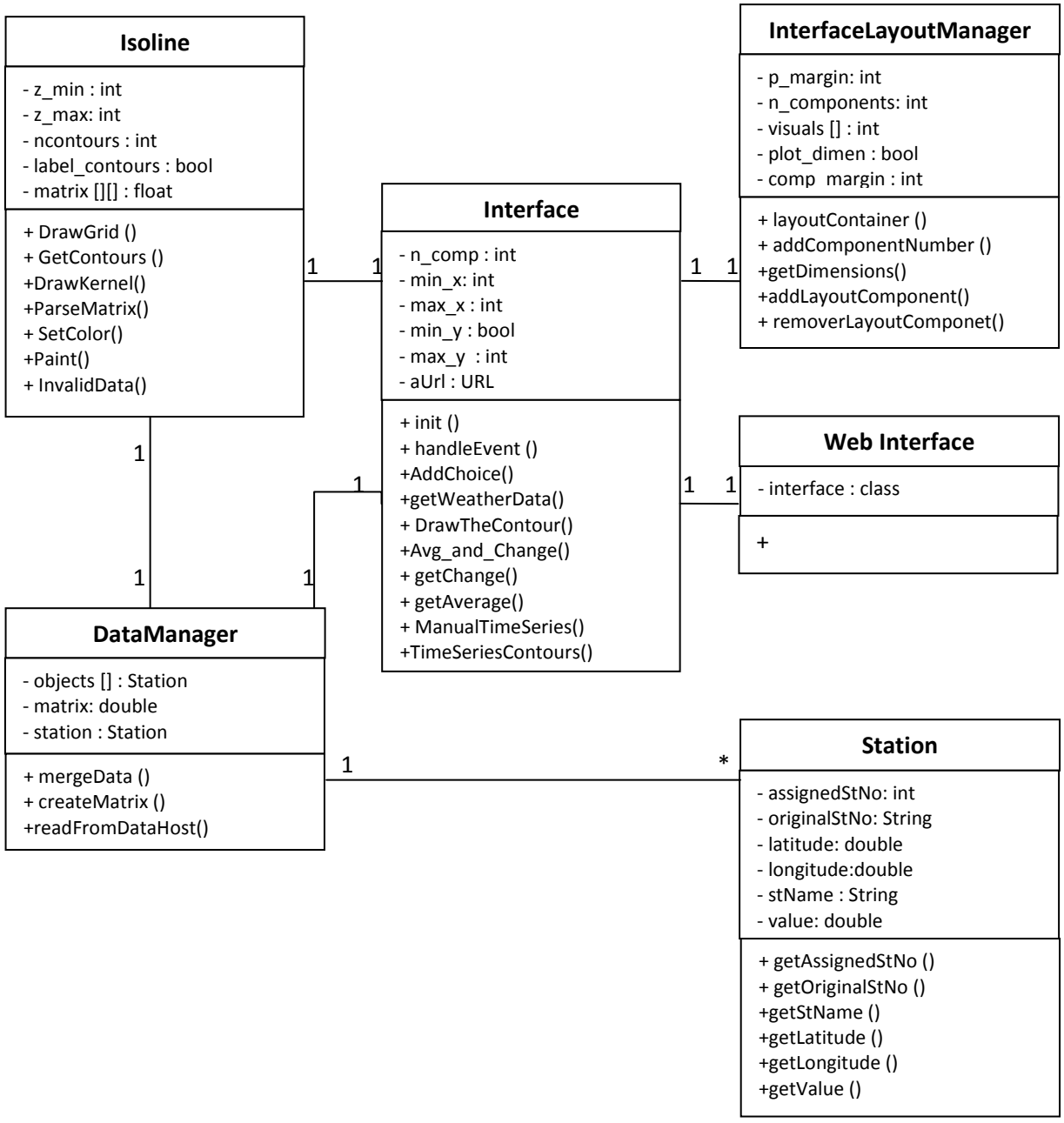

Figure 17: Detailed Class Diagram for the weather visualization system

3.5. Risk Analysis

As with all software development projects, there are several factors that could affect or influence the development process.

The following table illustrates the potential risks for the Weather Forecasting project.

Risk	Likelihood	Impact
Supervisor not available	Medium	Critical
Hardware Failure	Medium	Very Critical
Project team member falls ill during development period	Medium	Critical
Other commitments for project team member	Low	Fair
Technological restrictions	High	Critical
Less background information on topic	Low	Critical
Second supervisor not available for help	Medium	Critical
Lack of expert support and evaluation	High	Critical
Team member drops out of project	Medium	Critical

Table 2: Risk Analysis – potential project risks

The following table on the other hand illustrates how the risks we mentioned in Table 2 above were monitored and managed. It asserts strategies that were proposed and those that were actually used to mitigate the risks.

Risk	Possible Outcome	Action	Monitoring
Supervisor not available	No advice on progress	Continue working on project and meet with partners periodically	Contact supervisor via email or phone and consult with project partners to consider other possible solutions
Hardware Failure	Possible loss of project data	Keep periodically updated backups on several storage places	Periodically check status of hardware, work with system administrator
Project team member falls ill during development period	Project progress could be delayed depending on task for project member. As a result project could not be completed in time.	Ensure task independence i.e. project team members are able to test their sections independently.	

Other commitments for project team member	Team member has less time to work on project	Continue working on project	Advice team member to consider cutting down on external commitments
Technological restrictions	Unable to implement certain features or functionality of the system	Continue working on project.	Survey for alternative technological options
Less background information on topic	Lack of background knowledge on topic, could lead us into implementing a totally wrong system.	Continue working on project: Seek for ways to get materials on topic, consult the supervisor for help and suggestions	Meet supervisor for suggestions on sources of background information, do further research in the field.
Second supervisor not available for help	No advice on project progress, could be heading in the wrong direction	Continue working on project, try the first supervisor.	Try to contact the supervisor via email and/ or phone
Lack of expert support and evaluation	No expect advice on work progress. Could result in us going in the wrong direction.	Continue working on project	With the help of the project supervisor try to find expert support.
Team member drops out of project	No one to finish team members section of project. Project could not be completed.	See if it is possible to continue working on project, together with the supervisor come up with the best possible solution	Communicate to team members of personal progress and problems.

Table 3: Project risks Mitigation and Monitoring strategies

4. Implementation

This section considers the implementation process that was employed to realize the requirements as specified under *use cases* in the design section. It further discusses the different technologies and techniques that were used to implement the system. Technological limitations and several other factors that interfered and affected the implementation process have also been noted.

4.1. Programming Environment

Many development tools were suggested in the earlier versions of the weather visualization systems' design document. These includes: The QCodo Development Framework and the OpenGL specification. But these changed as we gained more background knowledge on the topic.

- **Programming Language**

In the end, the Java programming language was chosen as the primary implementation language with HTML as the markup for the web interface.
The reason behind this decision was because of Java's rich pool of [31] web development features. One notable reason behind the decision to go with Java was the Applets for web applications development option it provided. [30]

Lutz Prechelt presents a detailed comparison of Java and several other programming and scripting languages in [31] Empirical Comparison of C, C++, Java, Perl, Python, Rexx, and Tcl. Java came out to be the most memory consuming language out of all as Lutz concludes; [31] The typical memory consumption of a script program is about twice that of a C or C++ program and for Java is slightly higher.
Lutz also considered actually designing and writing the program in Perl, Python, Rexx, or Tcl takes no more than half as much time as writing it in C, C++, or Java but the resulting program is only half as long.

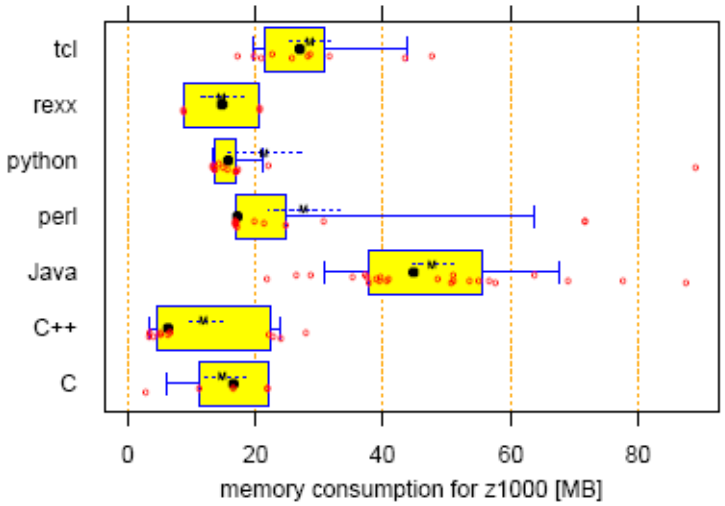

Figure 18: Amount of memory required by a program in each of the languages [31]

From *figure 18* we can see that among its several kinds of drawbacks, Java consumes a lot of memory as well.

But as much as these drawbacks could have hampered the implementation process of the system, Java still stood out to be the language of choice that could enable us to realize the requirements stipulated in the *Design chapter*. With Java, it was possible [29] to develop an application that was able to run in browser and this was one of the many goals for the end product.

4.2. System Development: Modular Programming

Developing the system was not done in one go, a rapid prototyping methodology was employed. The main problem was broken into several sub-problems and each sub-problem was solved independently and the respective solutions to the sub-problems combined made up the solution to problem, which is the *weather visualization system*. The technique being employed here is that of ***Modular* Programming**, considered here [32] as subdividing your program into separate subprograms such as functions and subroutines.
Subprograms make your actual program shorter, hence easier to read and understand. They also make it simpler to figure out how the program operates, reducing possibilities of bugs in the program significantly.

The main sub-problems to which the problem was broken down into includes; *Data Organization and Structuring*, *Contour Plotting Functions, Time Series, Interface Layout Manager, Interface, .NET URLConnection* and the *Web Component*

4.2.1. Data Organization and Structuring

Project Raw Data

Data for three weather variables was available for the project. The variables include precipitation, maximum temperature and minimum temperature.
It consisted of sets of weather stations each with it's recording for each of the variables.

Our initial task in the data organization stage was to study it carefully and try find patterns or inconsistencies in the data.
Two inconsistencies we found were that of missing data and some stations not common for all datasets.
To solve the missing data problem, stations that had extended days of missing data were not used for forecasting, and an estimation technique (expected maximization algorithm) was used to predict missing data values for stations that had a single day or two of missing data.
And as for stations not common for all data sets, we had to only use data for stations that were common for all datasets and

The data resulting from the data cleaning stage was used in the Causal Model and Bayesian Networks sections to build the networks and train the learning algorithms to do predictions. The station details, however, was also sent across to the visualization system to position the stations on the map (using the latitude and longitude lines).

Data Access (from Bayesian Networks Section)

A common data format was agreed on at the beginning of the project, and to ensure that the design and implementation of the visualization system was not put on hold until data was available from the *Bayesian Network* Section.

Station No	Station Name	Max Temp	Min Temp	Precipitation
……	……	…?…	?	?

Common format between Bayesian Networks and Visualization System

As a result sample data conforming to this format was used to start building the visualization system. The sample data was drawn from the original project data.

Merging Station Coordinates with Data from 'Common Format'

As you can see in the common format diagram above, it does not have the coordinates of the stations. As a result, another data file with the stations and their respective latitude and longitude values had to be included.

This data file merged with the 'common format' formed the data sets that were used in the visualization system.

Station No	Station Name	Latitude	Longitude	Max Temp	Min Temp	Precipitation
……	……	?	?	?	?	?

Data format for Visualization System

This sub section of the data organization stage, just like the others, was implemented in the ***Data Manager*** entity of the visualization system (see figure 18 above).

Station No	Station Name	Latitude	Longitude	Max Temp	Min Temp	Rainfall
0725756AW	SHINGWEDZI	-23.1	31.4333	26.5	11.0	0.0
0241214_S	TONGAAT	-29.5667	31.1333	22.0	16.0	0.0
0084700_P	0084700	-32.1667	18.9	24.0	13.0	0.0
0332150AW	GOLDEN_GATE	-28.5	28.5833	24.5	13.5	38.0
0182877_S	UMZUMBE	-30.6167	30.5	16.5	13.5	0.0
0211546_S	ILLOVO_MILL	-30.1	30.8167	19.5	14.5	2.0
0403886AW	FRANFORT	-27.2667	28.5	27.5	17.0	0.0
0043200_P	GROOTVLAKTE	-33.3333	19.6167	24.0	14.5	22.8

Table 4: Sample data - Data format for used in the Visualization System

Data Gridding

The contour plotting function and its sub functions use data in grid format to draw contour lines. Therefore, the data in the format above needed to be converted to grids. Each position on the grid represents a data value for a particular weather variable for the station whose latitude and longitude values corresponds to that position on the grid.
Each weather variable has its own grid.

4.2.2. Contour Plotting Functions

Grid Parsing

This is the first stage of the contour plotting routine. It is a sub function of the routine. This sub routine basically checks if the grid does not have any other data except the data values for the particular variable. It does the parsing row by row, using a row parsing sub routine.

```
{22.0, 20.0, 21.0, 21.5, 22.5, 22.0, 22.0, 22.5},
{21.0, 21.0, 21.5, 22.5, 24.5, 22.0, 22.0, 23.0},
{21.0, 22.0, 18.5, 19.0, 23.5, 22.0, 20.0, 20.5},
{17.5, 16.2, 19.5, 20.0, 24.3, 22.0, 20.0, 22.0},
{22.0, 20.4, 23.5, 23.0, 24.5, 24.0, 23.5, 27.4},
{21.5, 21.0, 23.5, 23.0, 22.0, 22.0, 22.5, 21.0},
{22.5, 23.0, 21.5, 22.0, 22.0, 23.0, 21.5, 21.5},
{23.0, 18.0, 19.5, 22.0, 21.5, 21.0, 21.0, 22.5},
{21.0, 22.5, 21.0, 23.0, 21.5, 17.5, 20.0, 21.0},
{18.5, 19.0, 20.0, 15.0, 16.0, 15.5, 19.0, 17.5},
{18.5, 18.0, 17.0, 17.7, 17.5, 22.9, 20.6, 16.6},
```

Table 5: Example grid - Maximum Temperature

Contour Values

Given the grid information, the program has a sub routine to calculate the contour values for the contour map. The subroutine interpolates between the maximum and minimum value on the grid to find the contour values. The number of contour values can be changed. But for the contour map to look less clumsy a few number of contours would need to be used.

Contour Drawing Kernel

This is the core function of the contour plotting routine. It is based on William Snyder's FORTRAN algorithm from 1978. [33]
With the aid of its many sub functions, it produces the contours that are painted on a map at the interface.

Possible states or scenarios for the contour drawing pen

a. at boundary of grid map
b. about to cross another contour line
c. is point visited already
d. can contour be continued

'**a**' is a case when 'drawing pen' is at any of the borders of grid map. The 'drawing pen' either goes to start drawing another contour line or continues with same contour line from a different boundary.

'**b**' is a case when 'drawing pen' is about to cross another contour line. The 'drawing pen' looks for any nearby points with same contour value, moves to a point it can without having to cross another contour line.

'c', this state indicates whether the point the 'drawing pen' is about to step to has been visited already or not. If it has and a contour line has already been drawn on it, it tries another route.

'd', this state basically communicate to the drawing kernel whether or not the contour that is currently being drawn can be continued in next iteration. It basically checks if next step will hit boundary, will cross another contour or whether or not it will be on a point already visited.

Each of these scenarios was implemented as a separate function.

Paint Method
This method uses Java's [29] Graphics library to print the work of the ***Contour Drawing Kernel*** to a grid map.

- **Summary of the Contour Plotting Algorithm**

Below is a brief discussion of the major functions that implements the contour plotting algorithm for the visualization system.
The implementation was broken into several sub-programs for a cleaner coder and easy debugging.

i. Parse Matrix

This subroutine parses the matrix of weather data. It checks that the matrix does not have any other kinds of data other than the rows of weather data separated by commas. After the matrix is confirmed clean, the matrix is now ready for the plotting algorithm.

```
loop until end of matrix
        ignore all characters until start of matrix
        if row of matrix is confirmed 'clean'
                add row to a temporary matrix
        else
                send error message

copy temporary matrix to main matrix //Main matrix is now confirmed clean, matrix can be accessed now.
```

ii. Get Matrix Extremes

This sub program reads through the z matrix and gets the minimum and and maximum data sets. These values will be used in the various sub-programs.

```
assign min = matrix [0][0]
assign max = min;

loop until end of matrix
        if min > value on current position in matrix
                min = value on current position in matrix
```

if max < value on current position in matrix
max = value on current position in matrix

iii. Assign Contour Values

This subroutine uses a delta value to calculate contour values. These values are saved to an array called *ContourArray* and it's this array the Contour Kernel uses as it travels around the grid to draw the contours.

delta = max in matrix - min in matrix /(number of contours)

loop until counter equals number of contours
countourArray := min + delta
counter++
return ContourArray

iv. Paint

This sub program is only executed after all necessary data structures and variables for the Contour Plotting Kernel have been initialized.
It is the Subroutine that uses Java's Graphics tool to draw the contour map on the applet in the web browser. It does so by calling supporting sub programs.

it calls the DrawGrid() method, which initializes the grid with the given width and height

and then calls the Contour Kernel sub routine (which draws the contour lines on the Grid)
- ContourKernel(Graphics) [see routine 6].

v. Draw Grid

This sub routine initialises the contour grid. It uses Java's graphics tool.

Graphics.clearRect(x, y, width, height)
- this method draws the grid rectangle on the Applet, starting at position (x, y)

Graphics.drawImage(Map, x, y, Object)
- Draws the map on which contours will be drawn. The map is automatically scaled by the Graphics object to fit the Grid.

loop until counter equals number of town/ stations to add //Add selected few stations to map,

Graphics.drawString(Station, latitude_x, longitude_y)

//now complete grid, by drawing grid lines

loop until counter equals number of x steps
Graphics.drawLine(x_1, y_1, x_2, y_2)

loop until counter equals number of y steps
Graphics.drawLine(y_1, x_1, y_2, x_2)

vi. Contour Kernel

This is the core function of the algorithm. It uses several sub-functions to draw contours. Selection of variables was based on Snyder's Algorithm.

//core variables in sub function
contourIndex := 0;
prevXY = -1; (any value below 0)
flag = 6;

[//Main sections in the routine]
//Calls the method that does the drawing (the "drawing pen")
//start the contour with flag = 6
DrawKernel(Graphics) [see function 7]

if DetectBoundary() returns false
DrawKernel(Graphics)

if CrossedByContour() returns false
DrawKernel(Graphics)

vii. Draw Kernel

It is the "drawing pen", it is called directly by ContourKernel() or by one of it's supporting routines in order to draw a segment of an Isoline or to set the pen position "prevXY". Its action depends on "flag":
flag == 6 means Set pen position
flag == 1 means continue the Isoline
flag == 2 means Start an Isoline at a boundary
flag == 3 means Start an Isoline not at a boundary
flag == 4 means Finish the Isoline at a boundary
flag == 5 means Finish closed Isoline not at boundary
If the public constant "label_contours" is true then when completing a contour

i.e. when "flag" == 4 or 5, the Contour value is drawn adjacent to where the Contour ends.

It is uses Java Graphics "drawLine(x1, y1, x2, y2) method to draw the line segments
with x1, x2, y1, y2 changing depending on the “flags”.

4.2.3. Time Series

This is a functionality that would allow users to view forecasts for different days over a forecast range in a sequence.
It was implemented in two versions. The first version allows the user to view forecasts for different by manually clicking on the next ‘day’ button.
Every time this button is clicked, the system would access the Data Host and retrieve the appropriate data and send this to the contour drawing function.

The other version is the ‘animated’ one. With this one, the user just has to activate the animation, and the system will itself retrieve the weather data for the respective days from the Data Host and send these to the contour drawing function. It does so until it reaches the last data set in the forecast range.

A timer in Java’s Threads was used for the time delay between displays in the time series.

4.2.4. Interface Layout Manager

While designing the interface, it became clear that using Java’s default [29] Layout Manager was not going to be a good option, since the interface required a good deal of customization. In the end, a customized version of Java’s Layout Manager had to be developed. And this was done by creating an entity that implemented the Java LayoutManager class.

This class made it possible to place components on the interface in all kinds of ways. It made organizing the interface layout much easier.

4.2.5. Interface

This is the application interface component. It uses the customized Layout Manager class to organize its components.

This section was implemented as a Java Applet. The *use cases* discussed in the *design* chapter were implemented here. This section is a kind of a ‘middle man’ between the *user* and the *visualization* functions.

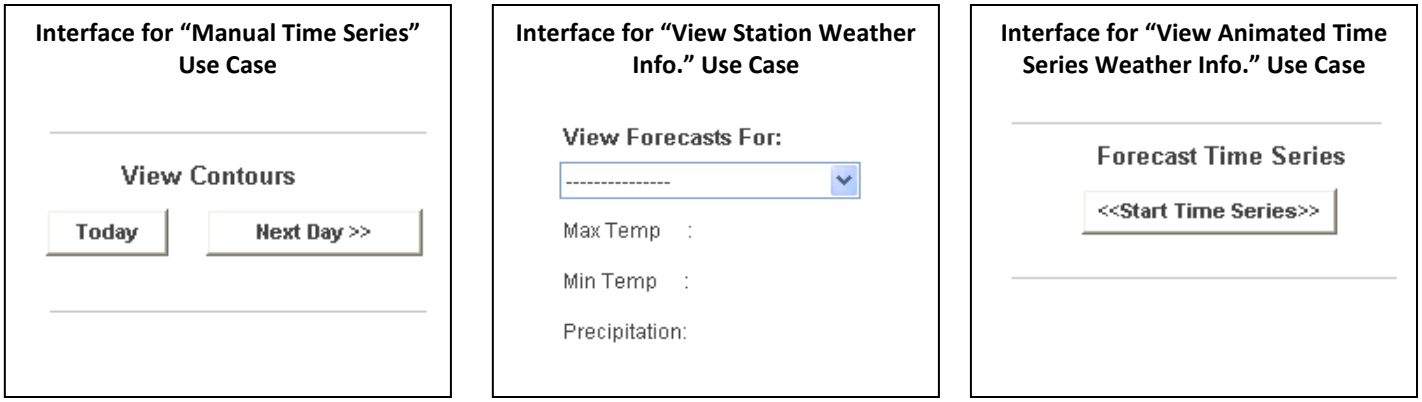

Figure 19: Some of User options on the interface

4.2.6. Web Component

This is the web page in which the Interface Applet will execute. It was implemented in HTML.

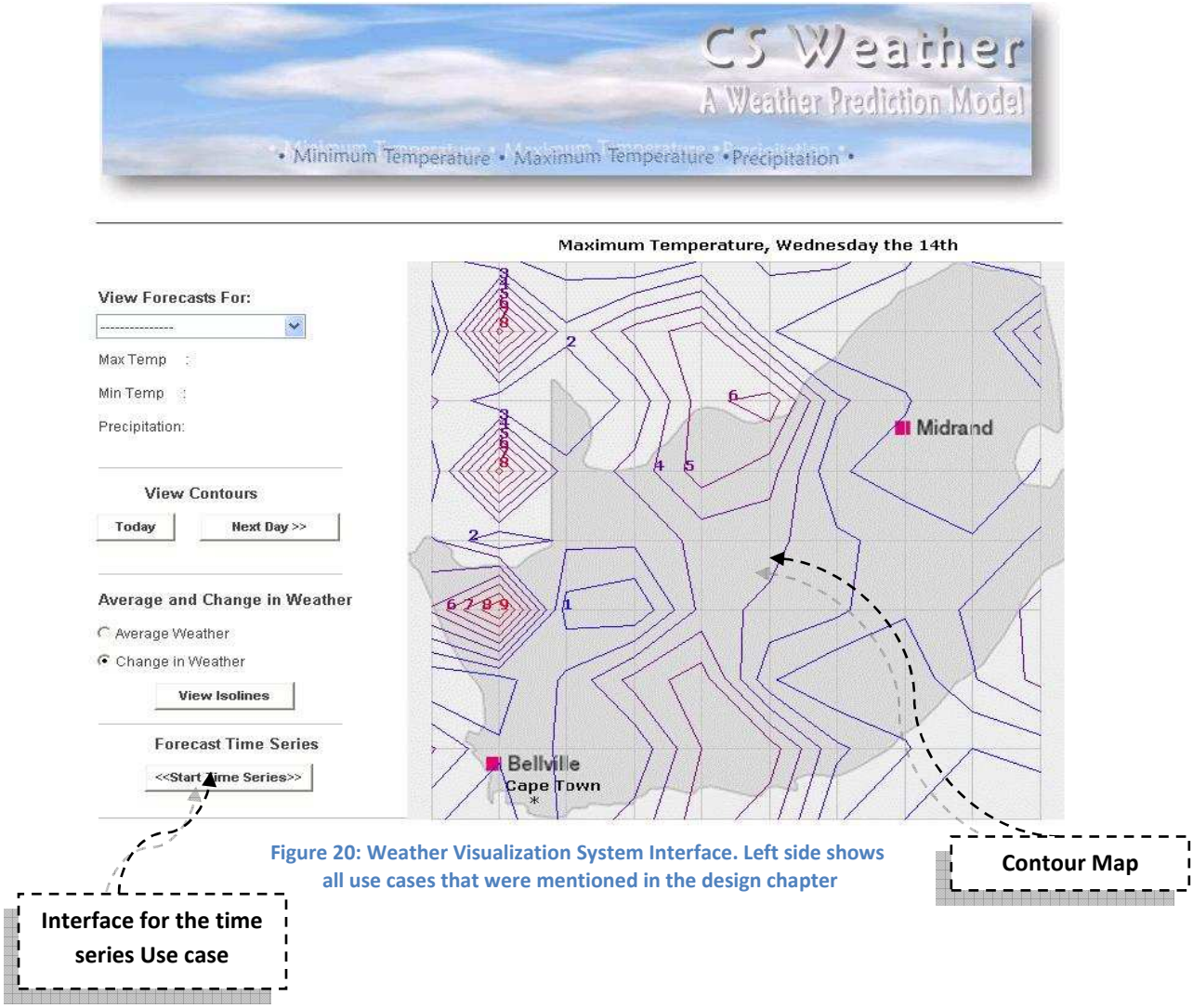

Figure 20: Weather Visualization System Interface. Left side shows all use cases that were mentioned in the design chapter

4.2.7. .NET URL

Java Applets have a host of limitations and some of these came as drawbacks to the implementation process of the Visualization system.

For example, [30] Java applets cannot read from a local disk. This basically means that you will not be able to access any data on your local disk once your applet is running.

As a result of this problem, we had to find alternatives ways to access the weather data for the visualization system.
The best alternative was to use [29] Java .NET URLs. With this system the data is stored on a web server or a client machine on a network. With the URL to the web server known, .NET URL Connection could then be used to create a connection between the applet and the web server or any data host, which could be a local client being used as a server. The visualization system was tested for both cases i.e. the weather data was stored on both web server and local client and the applet was able to connect to both separately (one whose url was specified).

4.3. Java Applets

Java applets are quite useful for developing web applications. As much as they make [30] web application development seem a lot simpler, they have limitations that could come as disadvantages to those employing them. [30]

Some limitations of Java Applets include:

- Unable to read from a local disk (client)
- Unable to write to a local disk (client)
- Unable to call programs running on the browser machine

Reasons for the many limitations are mostly to do with security restrictions, since applets pose a serious threat to security. It is very possible for an external user to access data being accessed by an applet. [30]

5. Evaluation

Several evaluation techniques were employed to test for the effectiveness and correctness of the weather visualization system.
Unit testing has been an on going exercise through out the duration of the project development. It was aimed at eliminating potential bugs at the source code level of the system.
The tests to be discussed in this section, however, are aimed at testing the system against its requirements as was discussed in *system design chapter*.

Another form of evaluation used for the system was usability testing. A usability test was designed. Users divided into two categories i.e. experts and non-experts, were asked to try out the system and give feedback by answering a questionnaire.

5.1. Test Plan

5.1.1. Scope

The tests to be considered in this test plan are aimed at testing if the visualization system meets its requirements as outlined in the design chapter.

5.1.2. Software Requirements

To be able to test the visualization system, the following software will have to be available on the test machine.

- Java JDK and
- A browser with Java plug-ins

5.1.3. Test Requirements

The main aim of these tests is to verify if the visualization system meets its requirements as specified in the *design section.*

System Testing

- Verify if overall, system is still functioning
- Verify if Time Series functionality works as expected
- Verify system meets use case requirements

Volume Testing

- Test system response to large data sets

Data Testing

- Very system response to incorrect data formats

5.2. Test Strategy

5.2.1. System Testing

This test strategy is for verifying if system works according the requirements specification. Check if system retrieves data well and whether the main functionalities of the system work fine.

Test Objective	Test system ability to retrieve data, to visualize contour maps in Time Series
Technique	Put data on web server, and specify server URL to system. Try one of the Time Series functionalities.
Completion Criteria	If system is able to time step through the forecast range, then functionality working well.
Special Considerations	None

Table 6: System Testing Strategy

5.2.2. Volume Testing

Test Objective	This test is to check how the system responds to large data sets
Technique	Increase the size of data sets. Try one of the Contouring functionalities.
Completion Criteria	System should display a contour map with contour lines.
Special Considerations	Maximum number of data sets used in the project 100. Using datasets more than this is primarily just for testing purposes.

Table 7: Volume Testing Strategy

5.2.3. Data Testing

Test Objective	This test is to check how the system responds to incorrectly formatted data sets.
Technique	Test system with data in an incorrect format.
Completion Criteria	System should display an appropriate error message.

Special Considerations	None

Table 8: Data Testing Strategy

5.3. Usability Testing

A usability experiment was designed for the system. Two expert users (in meteorology), six non experts were asked use the system and then asked to give feedback by answering a questionnaire.

The usability questionnaire is attached under ***Appendix B*** and the Ethics Declaration for the questionnaire attached under **Appendix A**

5.4. Results

System Testing
After several runs with all kinds of data sets for the different weather variables available for the project.
We were able to conclude that the overall functionalities of the system were working well. The system was able to access data sets from via a .NET URL and able to use these data to produce contour maps.

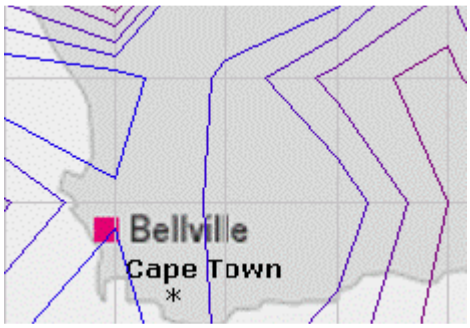

Figure 21: System Testing Output: A section of the contour map from the visualization system

Volume Testing

An increase in the amount of data did not break the system nor did it prevent the system from producing an output. However the output produced is too messy i.e. contour lines are not well spread on the contour map, which makes it difficult to understand and interpret.

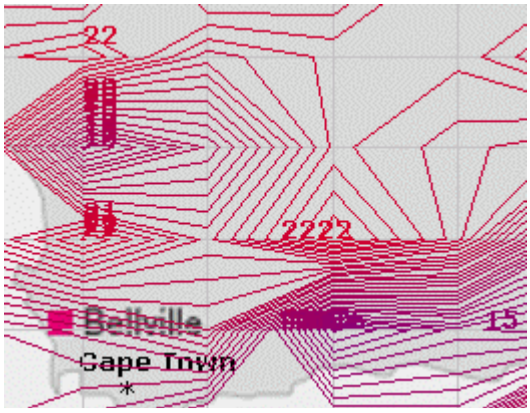

Figure 22: Volume Testing Output: A section of the contour map from the visualization system

Data Testing

The system deals with incorrect data format quite easily. It notices incorrect characters in the data grid and outputs the appropriate message.

Usability Testing

Based on feedback from the usability questionnaire, expert users felt it generally was a good system, reasonably easy to learn. It had potential to be developed in a large weather visualization system.

One in particular mentioned the idea of using smoother contouring algorithms to produce smoother contours.

The non experts on the other hand liked the idea of it being on a website.

Feedback on questionnaire attached under **Appendix B.**

6. Conclusions

6.1. Conclusion

After a detailed background study and work on the project, it became clear that weather visualization is a difficult and sensitive problem.
Making clear and understandable visualizations requires careful consideration and analysis.

Weather information can be represented in all kinds of ways such as weather maps and contour maps.
Most contour plotting algorithms that were investigated are data bound, i.e. they could only work with a specific format of data, proving how experimental implementing contour plotting algorithms is.

Java Applets have several limitations; this includes their inability to write to and read from a disk while active in a browser.
These limitations caused a major delay in our implementation process, since that we only learned about them almost half way in the implementation process. Reading from a web server rather solves the problem with limitations.

An attempt at following the project Timeline/ Gantt chart was made at beginning of the project. But as we got further in the project workload from course modules made it somewhat difficult to follow the project Gantt charts.

Increasing the number data sets makes the output from the visualization system messy and hard to interpret.

6.2. Future Work

The contour plotting algorithm and its sub routines used in the project were not smooth. Using smoother or implementing smoother algorithms could be a valuable improvement to the system.
Another possible improvement could be to shade the regions between contours.
Drawing just contour line may satisfy weather information users, but shading the regions between contours would make it much easier to understand and interpret the information.

The contour map could be made more interactive i.e. for example, allow users to click on a point on the map and view weather information for that point.

The project overall could be extended to include other weather variables such humidity, wind direction, pressure, dew point, and wind speed.
A major improvement could be to design 3D graphic representations of the weather information. For example, precipitation could be represented in 3D cloud graphics.

7. References

1. DRAWING CONTOUR PLOTS, Available from: http://avc.comm.nsdl.org/cgi-bin/wiki.pl; Accessed on the 23 of August 2008.
2. Isabel Harb Manssour, Carla Maria, Dal Sasso Freitas, Dalcidio Moraes Claudio, Flavio Rech Wagner, VISUALIZING AND EXPLORING METEOROLOGICAL DATA USING A TOOL-ORIENTED APPROACH, 1995
3. Kirk Riley, David Ebert, A SYSTEM FOR REALISTIC WEATHER RENDERING
4. Kirk Riley, David Ebert, Charles Hansen, Jason Levit. VISUALLY ACCURATE MULTI-FIELD WEATHER VISUALIZATION
5. M O Dayhoff , A CONTOUR-MAP PROGRAM FOR X-RAY CRYSTALLOGRAPHY, 1963
6. Michael Ferrari, GLOBAL WEATHER VISUALIZATION: UTILIZING SENSOR NETWORKS TO MONETIZE REALTIME DATA, Weather Trends International, 2008
7. Ralf Kunze, Robert Mertens, Oliver Vornberger. DYNAMIC AND INTERACTIVE VISUALISATION OF WEATHER DATA WITH SVG. University of Osnabrück, Germany. 2005
8. T V Papathomas, J A Schiavone, B Julesz, APPLICATIONS OF COMPUTER GRAPHICS TO THE VISUALIZATION OF METEOROLOGICAL DATA, 1998
9. GRIDDING AND CONTOURING, Available on: http://www.geoafrica.co.za/contouring/contour_tutorial_grid_maps.htm, Accessed on 10 October 2008
10. Mike Price, CREATING COOL CONTOURS, Entrada/San Juan, Inc.
11. Nathan Mantua, THE PACIFIC DECADAL OSCILLATION AND CLIMATE FORECASTING FOR NORTH AMERICA, Climate Impacts Group, University of Washington, Seattle, Washington, USA
12. Skotnes H, Hartvigsen G, Johansen D. 3D VISUALIZATION OF WEATHER FORECASTS AND TOPOGRAPHY, Institutt for informatikk, University of Tromsø, September 1994
13. NUMERICAL WEATHER PREDICTION, available at: http://www.metoffice.gov.uk/research/nwp/index.html , Accessed on 19 October 2008
14. STATISTICS OF WEATHER AND CLIMATE EXTREMES, Available at http://www.isse.ucar.edu/extremevalues/extreme.html, Accessed on 03 October 2008
15. Thomas M. Hamill, Andrew T. Church , WEATHER AND FORECASTING
16. Thomas Bengtsson, Doug Nychka , ADAPTIVE METHODS IN NUMERICAL WEATHER PREDICTION, First Spanish Workshop on Spatio– Temporal Modeling of Environmental Processes, Benicassim, 2001
17. Kevin Johnston and Judson Ladd, A CONCEPT OF OPERATIONS FOR AN INTEGRATED WEATHER FORECAST ROCESS TO SUPPORT THE NATIONAL AIRSPACE SYSTEM, National Weather Service
18. The GLOBAL FORECAST SYSTEM, Available at: http://www.emc.ncep.noaa.gov/modelinfo, Accessed on 17 October 2008
19. A. Staniforth, A. White, N. Wood, J. Thuburn, M. Zerroukat, E. Cordero, T. Davies, M. Diamantakis. UNIFIED MODEL DOCUMENTATION – MODEL FORMULATION, Dynamics Research, Meteorology R&D, Met Office. 13th September 2006
20. Antonio S. Cofiño, Rafael Cano, Carmen Sordo, José M. Gutiérrez, BAYESIAN NETWORKS FOR PROBABILISTIC WEATHER PREDICTION, 15th Eureopean Conference on Artificial Intelligence, ECAI'2002

21. Fred Kabul, THE WEATHER PROCESSOR, LBNL Simulation Research Group, April 1999 Available at: http://weather.unisys.com/wxp/, Accessed on 17th October 2008

22. DIGITAL ATMOSPHERE, Available at: http://www.weathergraphics.com/da/, Accessed on 13th October 2008
23. STORM PREDATOR, Available at: http://www.stormpredator.com/about.htm, Accessed on 13th October 2008
24. WEATHERBUG, Available at: http://weather.weatherbug.com/, Accessed on 22 September 2008
25. WEATHER UNDERGROUND, Available at: http://www.wunderground.com/, Accessed on 22 September 2008
26. CONTOURS, Available at: http://en.wikipedia.org/wiki/Contour_line, Accessed on 27 August 2008
27. Fayish A C, Jovanis P P. USABILITY OF STATEWIDE WEB-BASED ROADWAY WEATHER INFORMATION SYSTEM, 2004
28. Kara A. Latorella, James P. Chamberlain, GRAPHICAL WEATHER INFORMATION SYSTEM EVALUATION: USABILITY, PERCEIVED UTILITY, 2002
29. THE SOURCE FOR JAVA DEVELOPERS, Available at: http://java.sun.com/, Accessed on 15 August 2008
30. Marty Hall, Larry Brown. CORE WEB PROGRAMMING: APPLETS AND BASIC GRAPHICS, 2001 – 2003
31. Lutz Prechelt. AN EMPIRICAL COMPARISON OF C, C++, JAVA, PERL, PYTHON, REXX, AND TCL. Universit¨at Karlsruhe, Germany, March 14, 2000
32. Sun Microsystems, The Benefits of Modular Programming, 2007
33. William V. Snyder, ALGORITHM 531 CONTOUR PLOTTING [J6], Jet Propulsion Laboratory, California Institute of Technology, September 1978

8. Appendices

Appendices A: Ethics Declaration Form

Ethics declaration for Usability testing, all participants in the usability test were required to sign this form for ethics and information privacy purposes.

Weather Forecasting Using Dynamic Bayesian Networks

Ethics Declaration

The questionnaire contains information and questions relevant to the visualization interface designed for the project. It is a simple questionnaire designed to evaluate the effectiveness of this interface. Any information you provide will be anonymous and will be completely voluntary and you may withdraw at any time.

Thank you for your participation in our evaluation.

Matthew de Kock	Michael Kent	Pascal Haingura	Participant
_____	_____	_____	_____

Appendices B: Usability Questionnaires