5-1-2012

Dynamic Data Extraction and Data Visualization with Application to the Kentucky Mesonet

Anoop Rao Paidipally
Western Kentucky University, anooprao.paidipally443@topper.wku.edu

DYNAMIC DATA EXTRACTION AND DATA VISUALIZATION WITH APPLICATION TO THE KENTUCKY MESONET

A Thesis
Presented To
The Faculty of the Department of Mathematics and Computer Science
Western Kentucky University
Bowling Green, Kentucky

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

By
Anoop Rao Paidipally

May 2012

DYNAMIC DATA EXTRACTION AND DATA VISUALIZATION WITH APPLICATION TO THE KENTUCKY MESONET

Date Recommended: 4/30/2012

Dr. Jonathan Quiton, Director of Thesis

Dr. Guangming Xing

Dr. Qi.Li

21-May-2012

Dean, Graduate Studies and Research Date

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor and mentor, Dr. Jonathan Quiton, for his overwhelming encouragement and continuous support of my thesis study and research, for his patience, motivation, enthusiasm, and immense knowledge. Dr. Quiton has exceptionally inspired and enriched my growth as both a student and person. His supervision and guidance is unlike anything I have ever experienced. This thesis would not have been possible without the dedication of Dr. Quiton.

I gratefully acknowledge Dr. Guangming Xing and Dr. Qi Li for their supervision and precious time invested to read and provide correction to this thesis. I am thankful that in the midst of their busy schedules, they accepted to be members of my reading committee.

I would like to express my deep and sincere gratitude to Dr.Stuart Foster, Kentucky State Climatologist and Director of the Kentucky Mesonet and Dr. Rezaul Mahmood, Associate Director of the Kentucky Mesonet for the data and research direction. I would also like to thank Andrew Quilligan, Systems Administration at Kentucky Mesonet for providing us the access to the data needed for this project.

I would also like to thank those closest to me, whose presence helped make the completion of my thesis work possible. These are Venkata Aditya Korada - my best friend for the past 6 years, Mohnish Thallavajhula(great friend and Knowledgeable expert in Web development who has helped me many times when I needed the most), Gopichand Natunki, Sampath kumar Pasupunuri and Craig Dickson. Most of all, I am greatly thankful to Soujanya Siddavaram Ananta, who shared my happiness, and made me happy. Chinni, thanks for the love, patience and understanding. Most of all, I would like to thank my family, and especially my parents, for their absolute confidence in me. The knowledge that they will always be there to pick up the pieces is what allows me to repeatedly risk getting shattered.

Finally, I acknowledge research support from Kentucky Science and Engineering Foundation (KSEF-2013-RDE-012), Western Kentucky University Junior Faculty Scholarship

(No. 223149), and computing support from the Kentucky NSF- EPSCoR Research Startup Fund (RSF-031-06).

Anoop Rao Paidipally

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DYNAMIC DATA EXTRACTION AND DATA VISUALIZATION WITH APPLICATION TO THE KENTUCKY MESONET

Anoop Rao Paidipally May 2012 47 Pages

Directed by: Dr. Jonathan Quiton, Dr. Guangming Xing and Dr. Qi.Li

Department of Mathematics and Computer Science Western Kentucky University

There is a need to integrate large-scale database, high-performance computing engines and geographical information system technologies into a user-friendly web interface as a platform for data visualization and customized statistical analysis. We present some concepts and design ideas regarding dynamic data storage and extraction by making use of open-source computing and mapping technologies. We implemented our methods to the Kentucky Mesonet automated weather mapping workflow. The main components of the work flow includes a web based interface, a robust database and computing infrastructure designed for both general users and power users such as modelers and researchers.

INTRODUCTION

There are a lot of applications that benefit from dynamic data extraction and data visualization. Some of these areas include Business Intelligence (BI), Customer Relationship Management (CRM) and Enterprise Information portals (EIP) etc. Specifically, system integration of various components aims at combining selected systems so that they form a unified new whole and give users the ability to interact with all the modules on a single information portal.

This thesis deals with the design and implementation of a central information system that joins several modules such as a database, statistical tools along with other system software into one cohesive unit[20]. Our goal is to develop a fully functioning and flexible integrated information system that can be used for automated administration useful for providing timely critical information while reducing inconsistency and work overload.

1.1 Background

Our system is designed to serve both research and commercial goals of the Kentucky Mesonet. The Kentucky Mesonet is a network of automated weather and climate monitoring stations being developed by the Kentucky Climate Center at Western Kentucky University[14].

The Mesonet data is mainly used by researchers for making mesoscale modeling and predictions. In particular we are interested in upgrading the capability to power meaningful data visualization tools in real time if possible. For instance news channels on the television may show some kind of mesoscale data visualizations, but they may not be automated and therefore it may need someone to update it. Some national government organizations that employ automated systems use expensive tools which may not be feasible for small organizations such as State Mesonet to employ them.

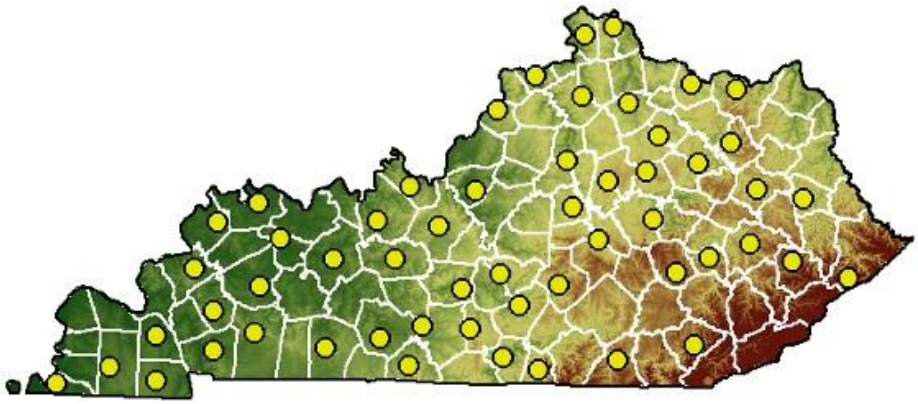

Figure 1.1: Mesonet Locations as of January 2011 [9]

Other mesonets such as Oklahoma and Texas have developed high end static graphs and images, they are yet to come up with real time updates and animations. This thesis attempts to move further into high-end animations.

Figure 1.1 shows the locations of all active Kentucky Mesonet sites as of January, 2011. These Kentucky Mesonet remote weather stations record data that are stored locally in the station datalogger until queried. The datalogger scans all the sensors every three seconds and stores the value in memory. Every five minutes, the station datalogger computes the average of the three-second values, providing 12 five-minute averages for each sensor per hour. These observations are used to derive the official record for the specified five-minute period. Measured parameters are values brought back to the Kentucky Mesonet data ingest machine directly from the station datalogger space and is further processed. Data ingest systems perform initial collection of data from network sites and populate tables in a relational database. This data is provided through the Kentucky Mesonet's public webserver. Data and metadata are provided in a rough, preliminary XML format [18]. Currently, selected users have access to their data repositories.

We started developing our own database infrastructure in order to have a more flexible system for different levels of users and to enable multiple system integration such as analytics and web/front-end interface. The specific advantages are enumerated as follows :

1.1.1 Availability

Due to limited bandwidth, the Kentucky Mesonet does not recommend bulk downloads. We can overcome this problem by downloading data in small chunks and storing them in our database. Thus, all our needed data resides in our system where we have full control over the required data.

1.1.2 Reliability

Reliability considerations indicate that this database has been redundantly maintained. The accessibility of critical data can be increased by redundantly maintaining the copy of the database. Furthermore, maintaining a database on its own gives us the capability to compute a larger number of requests rather than relying on the public server which has to be shared by many people.

1.1.3 Efficiency

We are fetching flat files from the Kentucky Mesonet and indexing them in our database resulting in an increase in data fetching efficiency which is critical to our automated weather mapping workflow.

1.2 Open-Source Software

Open Source Software are computer programs and applications whose source code is freely available to the developers. Developers using open source Software are generally allowed to modify the source code and redistribute. A software is considered open source software if it satisfies all of the conditions under the general headings listed below [20]:

1. Free Redistribution
2. Source Code

3. Derived Works
4. Integrity of the Author's Source Code
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Fields of Endeavor
7. Distribution of License
8. License Must Not Be Specific to a Product
9. License Must Not Restrict Other Software
10. License Must Be Technology-Neutral

More information about open-source software can be learned from the open source initiative website which is available online [20]. The advantages of open-source software are as follows [8]:

1. Stability

   With open-source software, you have full access to the source code which enables businesses to choose to upgrade as they see fit and not because of the pressure from the software vendor.

2. Auditability

   With the source code being open to the public, users can independently test and verify the usefulness of the software and it's flexibility in the face of future changes. Closed source software on the other hand forces the user to trust their claims.

3. Experimentation

   With open-source software we have the opportunity to try new technologies without the hurdles of commercial or non-disclosure licence agreements

4. Flexibility and freedom

   Software flexibility is about being able to choose solutions suitable for the needs of the users. The availability of the source code and the right to modify it is a very

important feature of the open source technologies. It enables the unlimited tuning and improvement of the software product.

On the other hand, there are downsides to businesses using open-source software. They must also be aware of the risks and the downsides of using them which were identified [4] as follows:

1. Code reliability

   Open-source software are usually contributed by many thus, there may be risks of unequal quality of the codes performance. In most cases, there is limited funds for quality control.

2. Lack of customer support

   Once we decide to use open-source software we are on our own. That is, we have to figure out how to install and use applications without sabotaging our data and hardware. Very few or no help documents or manuals may be available since the software may be changed very frequently. Though lots of help would usually be available on the Internet and there are many self-motivated forums that can help you install and run open-source software, there is no qualified support available.

3. Maintenance

   Another major drawback is maintenance: though most of the time issues are quickly addressed, some may take longer time, especially if the issues at hand are not 'real' problems but local ones in the context of few users.

1.3 Objectives

The main purpose of this thesis is to develop an integrated and flexible database and computing system for Kentucky Mesonet with application to climate modeling and high-resolution map generation. I started my research by identifying the general model problem

and gathering the problem parameters and requirements from my advisor and from Dr. Stuart Foster and Dr. Rezaul Mahmood from the Kentucky Mesonet. From this information we identified the critical components, designed and implemented our system using the Kentucky Mesonet data.

The specific objectives are as follows :

1.3.1 Replicate and design the database

If you design your solutions for multiple users, then database replication, in the appropriate circumstances, can improve the way the users share data. Database replication and design enables us to take a new approach to building this solution by creating a single database that contains all the data and related information. A complete set of data is still contained in the repository of the Mesonet, but each replica handles only a subset of that data that is required for that specific application. We designed the database such that we can integrate the analytics and the front-end components.

1.3.2 Propose a model

In this thesis, we have proposed a two-level integrated model for dynamic data extraction and data visualization. Visualization is critical to the effective analysis, and assessment of data generated by numerical weather prediction.

1.3.3 Apply to Kentucky Mesonet data

Finally, we aim to apply our model to Kentucky Mesonet data. Specifically, we will develop a web-based visualization portal with a back-end database that can allow users of the Portal to data visualization such as static images or animations.

This thesis is organized as follows: Chapter 2 describes the various components that are required for the model and have explained how the components exactly fit into our model;

Chapter 3 provides the details of the database/analytics/user-interface integration; Chapter 4 shows the results and outcomes of our design as it is implemented in the Kentucky Mesonet automated weather mapping workflow, and Chapter 5 provides the summary, conclusion, and ideas for further research and projects.

REVIEW OF LITERATURE, CONCEPTS AND METHODS

This chapter mainly focuses on the literature, concepts and methods that are required to understand this thesis. Section 1 of this chapter introduces the term dynamic data extraction and the design tradeoffs involved with it. Section 2 of this chapter address all the aspects of system integration; Section 3 describes data visualization concepts; Section 4 explains about Infographics, and finally Section 5 illustrates and describes each of the components that was used as the part of framework.

2.1 Dynamic Data Extraction

Data extraction is the act or process of retrieving data out of (usually unstructured or poorly structured) data sources for further data processing or data storage (data migration). It starts with an extracting system importing the data and then followed by data transformation or the addition of metadata prior to export to another stage in the data workflow. Usually, the term data extraction is applied when data is first imported into a computer from primary sources, like measuring or recording devices [23].

When we are using data extraction, we usually have to consider the following important design tradeoffs:

2.1.1 Latency tolerance

Some forms of data extraction imply a delay between updates to the data that is used by multiple applications. For example, in the Kentucky Mesonet data context, this could mean that sometimes a site may become offline due to several reasons, so the data at a particular time may not be readily available.

2.1.2 Push versus pull

When accessing a data source's database, a system can either pull the data from the database or let the database itself push the data when a change occurs. Pull approaches are generally less intrusive, while push approaches minimize latency. In our context we use the "pull" approach by repeatedly requesting from the main database to which we do not have the administrative privileges.

2.1.3 Granularity

If possible, getting a larger chunk of information at one time is generally more efficient than propagating each small change by itself. This requires an understanding of the cohesion between multiple data entities. That is, if one entity changes, other entities may also be effected. However, since we are fetching from a server that has multiple users, and so to avoid hogging the server's resources, we opt to download small chunks of data instead of large, bulk fetching.

2.1.4 Master/subordinate relationships

If updates are made only to one application's data, propagating these changes is relatively simple. However, if multiple applications are allowed to update the information, you can run into difficult synchronization issues. For a more detailed description of synchronization issues, see the Master-Master Replication pattern [7]. In our system, at least for the first version, we have restricted users to only browse information and potentially analyze data for superusers but they are not allowed to modify database entries.

2.1.5 Synchronization logic versus latency

For geographically dispersed applications, sharing a single database may cause excessive network latency. To overcome this problem, you can use distributed databases that

contain copies of the same data. However, distributed databases add the additional complexity of synchronization and replication logic. In our system, this may be an issue to address, if multiple users are regularly downloading or browsing the data.

2.2 System Integration

It is common that different organizations run different but co-existing applications. The problem they face is that maintaining these systems require considerable human resources which may not be cost efficient in the current highly competitive markets. The alternative approach is to integrate systems, While integration may have a considerable initial cost in human and computing resources, the long term benefits would potentially reduce the cost of staying with the integrated systems.

Integration of multiple software components aims at combining selected components of the software so that they form a unified new whole system and it gives users an illusion of interacting with a single information system. The reason for integration has two primary advantages. First, given a set of existing tools, an integrated view can be created to facilitate information access and reuse through a single information access point. Second, given a certain information need, data from different complementing information systems is to be combined to gain a more comprehensive basis to satisfy the need.

Organizations have always been concerned with data quality and integration. But the interest in improving data and content management is clearly on the rise, as companies are increasingly focusing on unifying their organizational wide data and on designing architectures to maximize the usefulness and accessibility of that data.

2.3 Data Visualization

Data visualization is the graphical display of abstract information for two purposes: sense-making (also called data analysis) and communication [5]. There isn't a universal

approach to data visualization, and many of the techniques IT experts use are still evolving. Some data visualization approaches might work better than others for an organization, depending upon that organization's needs. It is said that effective visualization tools come from knowing the goals and objectives of the clients.

Generally speaking, there are two categories of data visualization: exploration and explanation. The two serve different purposes, and so there are tools and approaches that may be more appropriate for one and not the other. For this reason, it is important to understand the distinction, so that we can be sure of using tools and approaches appropriate to the task at hand.

2.3.1 Exploration

Exploratory data visualizations are usually a first attempt to describe a dataset. When you need to get a sense of what's inside your data set, translating it into a visual medium can help you quickly identify its features, including interesting curves, lines, trends, or anomalous outliers, which is helpful in developing graphical tools.

2.3.2 Explanation

By contrast, explanatory data visualization is appropriate when you already know what the data has to say, and you are trying to tell that story to somebody else. In the context of our project, images of Kentucky map are an example of an exploration visualization used for a variety of reasons such as detecting trends or detecting critical weather events.

If exploratory data visualization is part of the data analysis phase, then explanatory data visualization is part of the presentation phase. Such a visualization may stand on its own, or may be part of a larger presentation, such as a animations that are being generated in our context.

2.4 Infographics

Information graphics or infographics are visual representations of information, data or knowledge [17]. These graphics are used anywhere where information needs to be explained quickly or simply, such as in signs, maps, journalism, technical writing, and education. They are also used extensively as tools by computer scientists, mathematicians, and statisticians to ease the process of developing and communicating conceptual information. They are applied in all aspects of scientific visualization.

The basic material of an information graphic is the data, information or knowledge that the graphic presents. In the case of data, the creator may make use of automated tools such as graphing software to represent the data in the form of lines, boxes, arrows, and various symbols and pictograms.

2.5 Components of our Automated Kentucky Mesonet Weather Mapping System

The first step of the design process was to identify the various open source components that would best fit into our model. Also since each and every component provide a wide range of services, the identification also required us to know exactly what services are needed. We seek products that provide not only the necessary functionality but also ease of use, flexibility, reliability, low total cost of ownership , and high return on investment.

We evaluate products similar to how others evaluate them which are as follows:

1. Technology

   Smaller organizations should look for products that execute on open source platforms to reduce the total cost of ownership. The integration platform should also be scalable. Organizations that initially implement their integration platform on Windows platforms should consider the ease with which they can migrate to a more robust platform (e.g., Unix).

2. Testing and development

   The platform infrastructure should include robust, isolated testing and development partitions that support both online and offline testing.

3. Documentation

   Technical documentation should be complete. Integrated, contextual documentation facilitates interface development, especially for the casual or less experienced user. In this regard, this manuscript should provide the technical description and details in sufficient detail such that others can understand and facilitate seamless future upgrades.

4. Ease of implementation

   The product should facilitate development across all types of interfaces and should maximize a developer's ability to reuse components (objects) previously developed for other interfaces. Ease of use is enhanced by a graphical user interface that enables users with minimal programming skills to develop message mappings of data to be shared between systems that use different formatting standards.

5. Flexibility

   The integration platform should support a wide variety of data exchange standards and message formats. The important software engineering principle of maximizing cohesion and minimizing dependencies of code applies here as well. Furthermore, components are meant as units of composition which can be used independently.

6. Application experience

   Developers should look for a vendor that has prior experience with their installed applications. This will facilitate the efficient development of interfaces and reduce support costs.

7. Performance

   Simulated load testing should be an essential part of product evaluation. Candidate products should exhibit satisfactory performance benchmarks not only at anticipated transaction volumes but also at significantly higher throughputs to ensure that future growth can be accommodated at acceptable performance levels.

8. Reliability

   Developers should seek documentation of the system availability experiences of current customers. They should also expect vendors to provide service-level agreements that guarantee acceptable levels of system availability and service response times for technical support when required.

9. Audit, security, error management, and recovery

   Audit and security features are important. Automated recovery and restore capabilities provide rapid recovery efforts in the event of a transaction corruption or system failure.

10. Monitoring tools

    An integration platform is one of the most vital systems, and if problems arise, early warnings and a rapid response are mandatory. Developers should carefully evaluate product features that allow IT operations to monitor system performance in real-time and trigger automated alerts when failures occur or performance falls below user-defined thresholds. Reporting features that allow support staff to analyze key workload statistics and other system management functions are also important to ensure optimal performance.

11. Ongoing support and maintenance

    Lots of help would usually be available on the Internet and there are many self-motivated forums that can help you install and run open source software, with onsite support guaranteed within acceptable time frames.

12. Total cost of ownership

    A number of factors combine to optimize the cost of ownership. An accessible technology platform such as an integrated information system reduces acquisition costs. Ease-of-use features and comprehensive support tools help to lower support costs. Release cycles and understanding what is involved in upgrading to the latest version of the software should also be taken into consideration when estimating the total cost of ownership

Additional considerations include the organizations financial viability, which impacts its ability to enhance and support the product for the long term. Small and non-profit organizations such as Kentucky Mesonet cannot afford to repurchase an integration platform because the selected vendor was not truly committed to the product or market and have stopped supporting the product.

I have filtered the components that were identified by the most popular components that are available by communication effectiveness and its capability to get integrated into a wide range of applications. Table 2.1 summarizes all the components that were used in this application. This section briefly describes the list of components and describes the services that are used.

Table 2.1: A List of Components used in the Automated Kentucky Mesonet Weather Mapping System

Name	Description
XML	Markup language for transport/storage of data.
LAMP Server	Acronym for Linux, Apache, MySQL, and PHP systems
Linux	is a open source unix based operating system
Apache Server	Which acts as Web Server
MySQL	relational database management system (RDBMS)
PHP	a server-side scripting language for Web development
Crontab	a time-based job scheduler for Unix-type operating systems
R language	an open source computing software
JavaScript	is a prototype-based client side scripting language
JQuery	is a cross-browser JavaScript library to simplify the client-side scripting

2.5.1 XML

XML stands for extensible markup language very much like HTML [22]. XML was primarily designed to carry data not display data. In XML, the author of the document would be able to create tags whose syntax and semantics are specific to the target application. Because XML syntax consists of text-based mark-up that describes the data being tagged, it is both application-independent and human readable. This simplicity and inter operability have helped XML achieve widespread acceptance and adoption as the standard for exchanging information between heterogeneous systems in a wide variety of applications, including Web services. Also the semantics of a tag is predefined or fixed but is

instead dependent on the context of the application that processes the document. Although XML was primarily designed to mark up content its main advantage lies in the way XML is used to describe structured data which makes it important both as a data storage and interchange format [18].

In a data-centric application such as Kentucky Mesonet where data is stored in a relational database or similar repository; we want to extract data from a database as XML, distribute the data to its partners, store XML into a database or both. By doing so it makes it easier for the partners to keep track of the data. This is a very convenient way of staying up to date with the content of a large number of sites. Also, it makes it easier for other partners to directly link the data. Because XML data can easily be read by computers, it's also easy for web designers to configure their sites so that the latest data is visible on their web page [15].

According to world wide web consortium [3] the XML specification defines an XML document as a text that is well formed if it satisfies a list of syntax rules that are provided in the specification. Some of these syntax rules include:

1. XML documents should have one and only one root element.
2. Each and every XML element must have a corresponding closing tag
3. XML tags are case-sensitive.
4. All the elements in the XML document must be properly nested.
5. All XML attribute values must be quoted.

From the programmer's perspective, it is very important to check the value of verifiable conditions on the data. One major advantage of XML is its ability to place the required preconditions on the data, and to do this in a very simple declarative way. We can define preconditions such that the root element should be 'observations' , the sub-root element should be 'observation group' and each observation group must consist of only one attribute

i.e data status and list of elements in an order. These kind of conditions can be defined by using a document type definition [16] or XML Schema Definition [21]. Checking an XML document against these conditions by using a document type definition or XML Schema Definition is called valid.

A DTD corresponding to an XML document is an optional part of an XML document that defines the document layout and structure. Even though its not required, there are many advantages of using a DTD. In order to validate an XML document, it has to pass through a processor which reads and specifies DTD, then verifies the XML structure to ensure that the elements of the XML appear in the order required, also that the require elements, the attributes and their values are in place, that no other elements or attributes are in place, that no other elements or attributes have been inserted where they shouldn't have been, and so on. If the XML document is known to be valid, then it is completely predictable. Let us consider the above example, As a developer we can write the code to read each piece of data from a validated document. Given the document being validated there would be no need to clutter the code with error checks or error handling; only one error check around the parsing code would be required. That piece of code would check for an error and the programmer can be confident that no other errors would exist or there would be no reading errors.

The motivation behind building an XML based workflow is to support multi - organizational workflow processes, as well as to support reusability, adaptability and survivability of both intra- and inter-organizational workflows. Multiple organizations on the Web can post their services as workflow steps, and these steps can be incorporated into other organizations' workflow processes using querying and browsing capabilities. Finally, the survivability is supported by replacing failed workflow components with functionally equivalent components at run-time, thus changing workflow schemas on the fly.

2.5.2 LAMP Server

Open source softwares are providing developers with a range of alternatives to commercial softwares that are very low in cost and are as efficient as their counterparts. LAMP is a wonderful preexisting integrated packaged example of such kind. LAMP is an acronym for the combination Linux, Apache, MySQL and one or more of Perl, PHP and Python. Tool supports integration with software configuration management tools, testing tools, application servers, and so on, and integration process is moderate and requires manual settings.

2.5.3 Linux

Linux is an open-source unix-like operating system which is a part of open source software development and distribution. The core and the most important component of the Linux is the kernel. Various research studies and experiments prove that LAMP architecture performs significantly well over other similar architectures [13]. One main advantage of using Linux is that any operation can be performed in a variety of ways. There are many tools that are available and the choice that we make should result in our ability to function within that tool, ease of use, desired functionality and overall 'feel', amongst other things. Besides the cost, the security of the Linux operating system is much more powerful than most of the commercial tools that are available [10]. So for sensitive projects such as Kentucky Mesonet, we prefer to choose Linux-based machines.

2.5.4 Apache Server

The Apache [6] in LAMP is the Apache HTTP server, commonly referred to as Apache web server. The primary function of a web server is to deliver web pages on the request of clients.

Apache web server is the most widely used open source software. The open design and the code allows the developers to create their own custom enhancements on the top of

core Apache program. Apart from Apache core, there are also a lot of custom extensions that are available for free. If a custom feature that corresponds to an application does not exist, then the developer can build their own. Many developers around the globe constantly contribute to its growth, which is available to any one using this server.

Apache has built-in support for a wide range of web programming languages, including Perl, PHP and Python. These languages are easy to learn and can be used to create powerful online applications. Apache also includes "SSL" and "TLS" support, which are protocols for sending encrypted data over the Internet, and are important in the development of safe online stores and other applications requiring privacy.

As described earlier, one of the main disadvantages of using open source software is that there is very little customer support available. However, the Apache Web Server features a large user support community. Unlike many software companies that handle all program support from one location, Apache technical support is spread throughout multiple locations, companies, and forums. This distributed model of support allows users to obtain answers to technical questions at any point of time, no matter where they are located. By being open source, Apache is connected to many users who are able to create technical patches and bug fixes very rapidly. As soon as a problem is found, users around the world communicate and contribute solutions. The result of this community support is software that is very stable and well maintained.

2.5.5 Database

The database that we have used in the project is MySql. The reason for using this is that it is a highly capable system that is usually used for running enterprise level sites with varying degrees of database complexities. MySQL is characterized as a free, fast, reliable open source relational database. At certain times there will be a trade off between speed and capabilities, and the MySQL team intends to keep their database engine fast and reliable. Aside from being the open-source database, MySQL is extensible, offering

multiple variations in engine types. Furthermore MySQL's performance is much superior than many of its counterparts, mainly due to the format of its default engine which is MyIsam. In addition, MySQL tends to perform much better on Linux and Unix types of operating systems when compared to commercial operating systems such as Windows. In terms of security, MySQL supports most security mechanisms currently used in the market and also supports custom development of security mechanisms.

2.5.6 PHP

PHP is the programming platform that holds together all the other components of the LAMP system. It's a language that helps write all the dynamic content that can access the entire data stored in the MySQL database. PHP is the most popular and widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML [source:php.net]. PHP is fast, stable, secure, open source and most importantly easy to use. PHP doesn't use a lot of the system's resources so it runs fast and doesn't tend to slow other processes down. It works well with other software and can be quite fast. PHP is also fairly stable. Another key advantage of PHP is its connective abilities. PHP uses a modular system of extensions to interface with a variety of libraries such as graphics, XML, encryption, etc.

To date the LAMP server is considered as the best example of integration of server components. The huge advantage of using LAMP is that as a developer we can build an app locally and then deploy it onto the web. Other advantages include ease of development and ease of maintenance.

2.5.7 Crontab

A system based scheduler is an application that is responsible for performing a given task in the background at any given point of time. The basic feature expected out of job

Figure 2.1: Pictorial representation of a crontab

scheduler software is the automatic submission of the executions. The system based scheduler that is available in Ubuntu Linux is CronTab. CronTab is another important component of this model.

A cron is a utility that allows tasks to automatically run in background of the system at regular intervals by use of the cron deamon. Crontab is the date and time based execution of background tasks based on a defined period during which resources are available for processing. Crontab is a file which contains the schedule of cron entries to be run at what times they are to be run. To see what crontabs are currently running on our system the following command can be used in the terminal.

sudo crontab -l

To edit the list of crontabs the following command can be used at the terminal

sudo crontab -e

Figure 3: Pictorial representation of a crontab

2.5.8 R language (Statistical tool)

The R software is an open source object oriented computing platform [11] which we have chosen as our computing component of our system. R has built-in GIS shapefiles

for Kentucky that we have used to create our climate contour plots. The scripts in the R language is being developed by Dr. Jonathan Quiton in collaboration with Kentucky Mesonet.

2.5.9 Java Script

JavaScript is a scripting language designed primarily for adding interactivity to Web pages and creating Web applications. Client-side JavaScript programs, or scripts, can be embedded directly in HTML source of Web pages. Depending on the Web developer's intent, script code may run when the user opens the Web page, clicks or drags some page element with the mouse, types something on the keyboard, submits a form, or leaves the page.

JavaScript can function as both a procedural and an object oriented language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in compiled languages like C++ and Java. JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval), object introspection (via for ... in), and source code recovery (JavaScript programs can decompile function bodies back into their source text). Javascript is supported by all major browsers.

jQuery [2] is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML. jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development [12]. It aims at making the developers write less maintenance code and concentrate on the actual behavior logic. It helps the developers to access the HTML DOM elements and manipulate them in a very easy way.

Also, the code itself is very robust and reliable. That is the reason why jQuery is the most popular JavaScript library/framework. jQuery is used by more than 49% of the

websites. Other popular competing frameworks to jQuery are YUI (Yahoo User Interface), Scriptaculous, etc.

jQuery's design helps implementing animations very easy. Handling browser events is a breeze using jQuery. The major success of jQuery comes from its ability to extend. It is very easy to develop plugins for jQuery. As a result, it has a very dedicated and helpful community of users. The development of jQuery is collaborated by the use of the revision control system, Git. jQuery is hosted on Github and can be accessed at [2].

jQuery is hosted on multiple Content Delivery Networks (CDN) which include Google and Microsoft. So, using jQuery is simple. Just include a script tag pointing to the required version of the jQuery library from one of these CDNs and you're good to go.

For any given version of jQuery, there are two library modes available.

1. Development version
2. Production version

The Development version of the library contains all the comments, debugging code (ex: console.log etc) and the original code in its original form along with all the proper indentation. This, in another way is nothing but the human readable form. It is generally named jquery-<version_number>.js

The Production version of the library is also known as the minified version. It means that the code is not in a human readable form. It serves two purposes.

1. Provides security to the code,
2. Minifies the code heavily resulting in faster load times.

PROJECT DESCRIPTION

3.1 Data Extraction Workflow

In this chapter, we present our database component that provides selective extraction of data objects from XML documents, store these documents in an object-relational database, and retrieve/reconstruct data into CSV files which acts as an input to or statistical component which is responsible for generating the maps required for animation.

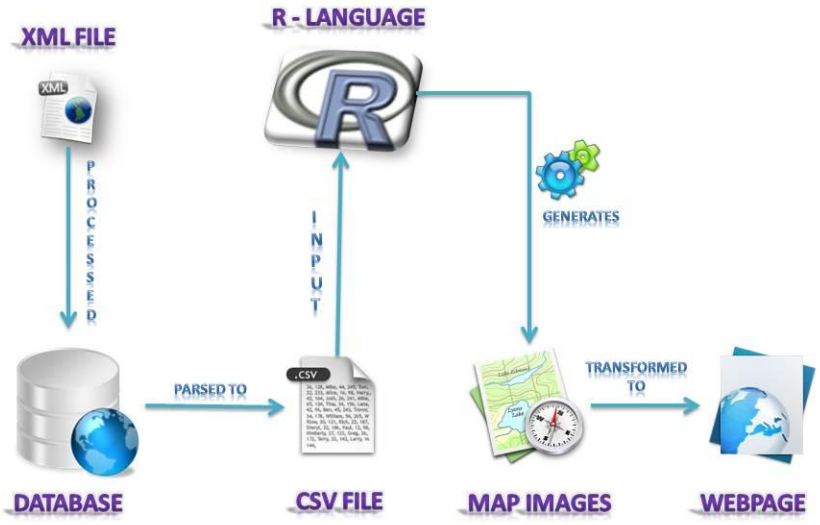

Figure 3.1: Architecture model of Dynamic Data Extraction and Visualization

A model of the business requirements would be necessary to ensure that there is a clear understanding between components of what is needed. Only then can we ensure that the requirements in terms of sources, transformations, and targets that are needed to move data be clearly communicated via a common, consistent approach.

The first step of the above process involves data extraction and storing it to the relational

```
<NetworkSites>
        ...
        <NetworkSite>
                <Network>KYMN</Network>
                <STID>FARM</STID>
                <NetSiteName>Bowling Green 5 S</NetSiteName>
                <County>Warren</County>
                <State>KY</State>
                <Lat source="NAD83 GPS" units="decimal_degrees">36.92669</Lat>
                <Lon source="NAD83 GPS" units="decimal_degrees">-86.46519</Lon>
                <Elevation source="GPS" units="feet">559</Elevation>
                <SiteClass>COLLECTION</SiteClass>
        </NetworkSite>
        ...
</NetworkSites>
```

Figure 3.2: Sample XML data format

database management systems.

As already mentioned in chapter 1 our source of information was from the Kentucky Mesonet in the form of XML files, which were regularly being uploaded to the available online links.

One available solution to extract the data from XML files would be that the DBMS essentially links a given table of a schema with a XML file and, during query processing, parses data from the XML file on-the-fly. Oracle, for instance, offers an option to have external tables while MySQL enables the CSV engine. As a result, data can be queried without having to explicitly load the raw data into the DBMS. In practice, however, XML files are still outside the DBMS as there is no support for indices, materialized views or any other advanced DBMS optimization. Query processing performance is therefore lower when compared to the the performance of queries running on internal tables, so the systems mostly offer external XML files as an alternative way for the user to load/copy data into normal DBMS tables rather than for query processing. Since the data is queried quite regularly say every 5 minutes or so it would be efficient to use the later approach.

An XML-relational mapping scheme is used to create a relational schema corresponding the "filtered" hierarchy of an XML document. Actually, both an XML document and a relational database can be viewed as trees as shown in Figure 4.1.

There are two important links provided by the Mesonet, providing the information about the location information and data regarding the temperature, radius etc., respectively. The first link points to a XML file named sitelist_xml that provides the information about the location details such as latitude and longitude coordinates, time zone, address of the location, etc. A PHP script was written that extracts the data from the XML and adds this data to the master table.The data that is inserted into this table is used to generate tables (if none exists) with respect to the location name dynamically. Any new addition to this table would allow PHP script to create a new table.

Similar to the above another link was provided that points to a XML file called site_latest_data _xml that gets updated every 5 minutes with the information about the temperature and other information for every corresponding locations. This data is extracted from the XML using the PHP script and is being used to insert them into the corresponding tables that are already created. Each table corresponds to information regarding a particular site. For example all the information corresponding to ALBN site is stored on ALBN table and so on. The data is extracted as a record and it is inserted by finding the location on it and inserting it to the respective table. This record consists of information of values of data such as air temperature, relative humidity, wind speed etc. at that particular location.

Data is observed by the PHP script for every 5 minutes. This PHP script is assigned as a job to crontab to schedule it for every 5 minutes. Five minute collection sites are scheduled for collection at: 00, : 05,: 10 and so forth, minutes past the hour. During the course of our testing we have identified small collection delays. So as to avoid these schedule conflicts, we have attempted to retrieve data at times slightly offset from the project's collection schedule, such as 2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52 and 57-minutes past the hour for data collected on a 15-minute schedule. This would extract the information and insert them

into the table every second minute.

The database consists of the same number of tables as that of a number of sites. Each time a new site is added then the PHP script creates a table dynamically to the database. Each table saves all the corresponding data pertaining to a particular site. Any kind of database failures such as table crashes and inappropriate format would be handled by the PHP script. Also a frequent backup of database is performed to avoid the loss of data under any circumstances. The database engine used here is InnoDB.

3.2 Analytics component

The R software is an open source object oriented computing platform [11] which we have chosen as our computing component of our system. R has built-in GIS shapefiles for Kentucky that we have used to create our climate contour plots. Dr. Jonathan Quiton, my advisor, created the function(climatemap.ky()) which is used to interpolate the climate data over the Kentucky map.

3.3 Front-end interface

The final component is to add a front-end interface that shows static and animated weather maps. With simple tools such as an editor, and a bit of scripting, we can develop any kind of animations with vibrant colors and consistent style. Generating animations by the rapid display of many images, or frames, each with a slight change from the previous one is one of the oldest ways to generate animations. In the early days cartoons were made of literally thousands of images, stitched together in exactly this way. With modern scripting tools simulation of this motion is made possible. Also with the help of scripting even interactions are possible although a bit complex.

In the case of our weather data images each and every image represents the weather data corresponding to a single point of time. So when the above images are set to animation it

represents the flow of data of weather related events and helps users predict the weather in the nearby locations.

There are lot of open source tools available in the market for generating or displaying these kind of animations. The most important attribute for making effective animations with scripting is the keyframe feature. Typically, a keyframe is a period or a point on the time line where certain key events can occur. Suppose we think of a movie with parts of scenes where a character carries out one motion and then another. Then we may say that there is a keyframe where the first motion stops and the next motion begins. When exported and displayed in a browser, the generated animations run at average speed. That is, the frame rate is an average of the minimum and the maximum allowed frame rate for the client web browser. The faster it is the faster the animations run. The best alternative is to insert a time control between frames, thereby changing the frame display rate.

APPLICATION

The chapter illustrates the application of the above said concepts to the Kentucky mesonet data.

4.1 Overview

I have followed a simple phased approach for completion of my thesis. Each phase was considered as the milestone and was derived from the list of components [19]. I haven't followed any standard development life cycle, but I have followed the software engineering principles from the beginning.

Though a complete detailing of the numerous resources used in implementation of this model is beyond the scope of this thesis paper, a few examples are reviewed below.

As already stated the data is available in the raw XML formats from the Mesonet servers at predefined locations which is accessible only to partners. Users who wish to access the data can request it from [9]

The following steps are required to produce the visualizations from the given data set.

1. Use of XML parser to read the data from the XML file and store in the database.
2. Creation and maintenance of database.
3. Generating a metadata file which acts as an input to statistical softwares.
4. Processing the metadata file to generate the required images.
5. Creation of animations from the generated images.
6. Finally embedding the animations to the webpage.

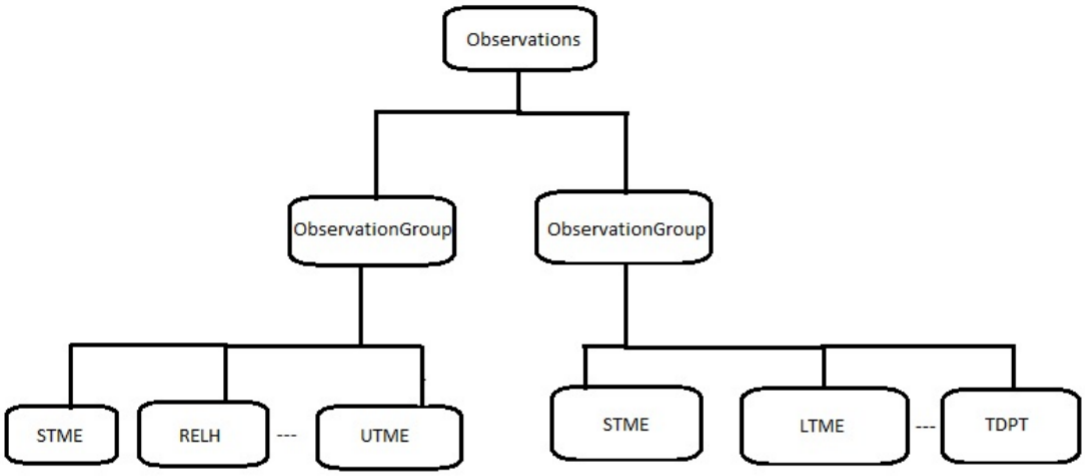

Figure 4.1: Pictorial representation of a database

4.2 Use of XML parser to read the data from the XML file and store in the database

The easiest way to read a well-formed XML file is to use the Document Object Model (DOM) library compiled into some installations of PHP. The DOM library reads the entire XML document into memory and represents it as a tree of nodes, as illustrated in the figure 4.1.

The script starts by creating a new DOM document object and loading the Observations XML into that object using the load method. After that, the script uses the getElementsBy-Name method to get a list of all of the elements with the given name. Within the loop of the Observation nodes, the script uses the getElementsByName method to get the nodeValue for the Network Name, AWIPS tag, and all other tag information and those values are stored into their respective arrays. The nodeValue is the text within the node. The script then uses these attribute values to insert into the database. To do so a prior connection is established with the database with required credentials. Then the script uses mysql_query() function to run the query. If the script is not able to make the connection, then script terminates further execution and tries to establish the connection later.

During the course of execution, if a new record appears for which the database table does not exists, then the script to update the location info is called that creates the new

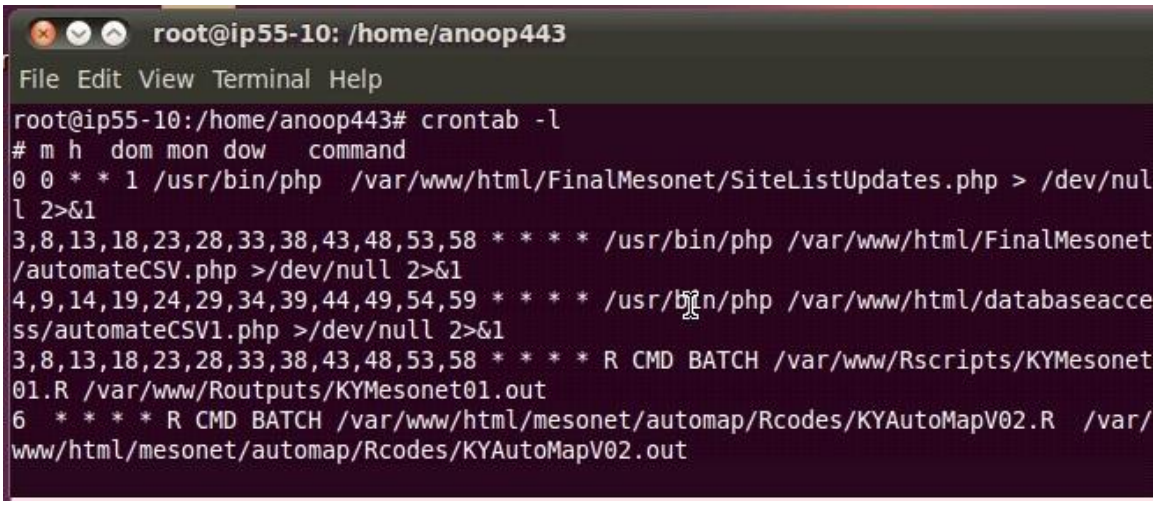

Figure 4.2: Pictorial representation of a crontab in the server

table, while retrieving the corresponding data from the location Info file. In case of table crashes, a query is called to automatically repair the tables.

By using the recursive function, I've managed to substantially reduce the number of "if" conditional statements in my script; the code is now easier to read, and also structured more logically.

Because of the heavy load on the database, I have tried to increase the response time of the script by preprocessing the data. When we try to insert data to our database, our application or the database itself (using triggers) does additional calculations or data insertions. This adds some overhead when data is being inserted, but according to our requirements, it is necessary. This is a form of event processing. In other words, if we preprocess our data ahead of time, we can reduce the time it takes to complete a request and also increase the efficiency.

4.3 Creation and Maintenance of database

The database consists of the same number of tables as that of a number of sites. Each time a new site is added then the PHP script creates a table dynamically to the database. Each table saves all the corresponding data pertaining to a particular site. Any kind of

database failures such a table crashes and inappropriate format would be handled by the PHP script. Also, a frequent backup of database is performed to avoid the loss of data under any circumstances. There also exists a table to store the data related to the list of sites in the database. This table stores the latitude and longitude positions of the sites along with the relevant location information. The primary key of the tables would be the site name which would be the foreign key to each of the tables. Therefore, the database engine used here is InnoDB.

4.4 Generating a metadata file which acts as an input to statistical software

The data that is inserted into the tables every five minutes from the various sources now has to be retrieved into a single metadata file which is accepted by the statistical softwares. Some of the most common formats are CSV, TXT, XLS etc. Among these we have used CSV since it is considered as the best among the available formats. Since the required information has to be retrieved from multiple tables it would be inefficient to use query from individual tables. Therefore a stored procedure was written which used the union operator to combine the results from all the tables and return the result set. Also, joins were not used to join the location information with the corresponding table values. Instead, the join operation was performed by using the script since the scripting tool has a constant performance that cannot improve over time as the data in the database gets increased. So similar to above a PHP script was written which would read the data from the database and generate the csv file. The PHP script matches the site location information with the values at any particular point of time and writes into the CSV file. The PHP script to generate the CSV file is scheduled as a job in crontab for every 5 minutes. This file contains the information shown in the Table 4.1 . The generated CSV file is stored at a particular location on a hard drive which acts as an input to statistical software. This CSV file is used for further processing. The CSV file consists of information as shown in Table 4.1.

Table 4.1: Contents of AutomateCSV

Variable	Description
Network	Network abbreviation
STID	Network-site abbreviation
NetSiteName	Relative site location
County	County
State	State
Lat	Absolute location: Latitude
Lon	Absolute location: Longitude
UTME	UTC time of observation
UTME_AWIPS	UTC time of observation
LTME	Local time of observation
STME	Standard time of observation
TAIR	Air Temperature 1.5m
RELH	Relative Humidity
TDPT	Dewpoint Temperature
WSPD	Wind Speed
WDIR	Wind Direction
WSMX	Maximum 3-second wind speed
SRAD	Solar Radiation
DPRC	Daily Precipitation Total

The key idea is that data output in flat file. This file gets updated at regular intervals. When queries arrive, the statistical software will take care of bringing the proper data from the flat file, and it will store it and evaluate it in and appropriate way.

4.5 Processing the metadata file to generate the required images

The following algorithm is taken from Dr. Jonathan Quiton's interpolation algorithm as implemented in climatemap.ky() function:

1. We generate a vector distance D from each Kentucky Mesonet site to any given point $P = (x, y)$.

2. We take the nearest k sites. The default is $k = 3$.

3. For each of the sites selected, we created a weighting function $w_i = \frac{f_i}{\sum_{j=1}^{k} f_j}$ where $f_j = \frac{1}{d_j^r}$, d_j is the distance of the j^{th} site to point P and r is a constant that determines how much influence a site gets if it is closer to point P. The default value of $r = 2$ which corresponds to Newton's law of universal gravitation. Other weighting algorithms and incorporation of location elevation will be considered in the future.

4. Finally, we generate the interpolated climate by $\hat{C}_P = \sum_{i=1}^{k} w_i C_i$ where C_i is the climate information at the i^{th} site.

4.6 Creation of animations from the generated images

jQuery's design helps implement animations very easily. Handling browser events is a breeze using jQuery. The major success of jQuery comes from its ability to extend. It is very easy to develop plugins for jQuery. As a result, it has a very dedicated and helpful community of users. The development of jQuery is collaborated by the use of the revision control system, Git.

jQuery is hosted on multiple Content Delivery Networks (CDN) which include Google and Microsoft. So, using jQuery is simple. Just include a script tag pointing to the required version of the jQuery library from one of these CDNs and you're good to go.

Including a script tag is as follows:

Table 4.2: A sample script tag for jQuery

```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.0
      /jquery.js">
</script>
```

jQuery is an open source library which is used to improve the performance of JavaScript. The Closure Compiler compiles JavaScript into compact, high-performance code. The compiler removes dead code and rewrites and minimizes what's left so that it downloads and runs quickly. It also checks syntax, variable references, and types, and warns about common JavaScript pitfalls. These checks and optimizations help you write apps that are less buggy and easier to maintain.

By default, jQuery uses "$" as a shortcut for "jQuery". So, using $("#id") or jQuery("#id") is the same. jQuery in the code is an object. A JavaScript object.

Extending jQuery with plugins and methods is very powerful and can save you and your peers a lot of development time by abstracting your most clever functions into plugins. In my work, I've made use of a plugin called SCIAnimator [1], which helps in animating the images using the jQuery library. Also we have extended this plugin to suit our needs.

This plugin makes it so simple that everything boils down to a simple method call onto the jQuery object after using the CSS selectors. The name of the method is 'scianimator()'. This method call takes a JavaScript Object as its input (a JavaScript hash). This hash contains an array of images to be supplied, in our case, the dynamically generated weather images, the configuration relating to the display of the animation such as width,

defaultFrame, theme etc. Following is a sample method call on the jQuery object ($).

Table 4.3: A sample method call on the jQuery object on the KY Mesonet images

```
$('#scianimator').scianimator({
    'images': ['images/relh1.jpg', 'images/relh2.jpg',
                'images/relh3.jpg', 'images/relh4.jpg'],
    'width': 600,
    'theme': 'dark',
    'defaultFrame': 'last'
});
```

4.7 Embedding the animations to the webpage

The web portal that we discuss here supports any dataset in which values can be accessed through dates and processes the dataset from the database, and produces the visualization of the dataset.

This means that a layout or background needs to be stored just once, with the object to be moved and places another layer on top of the background. The application follows simple programmed instructions on how the object is to be moved. The result is a far more economical animation. JavaScript can make animations by either method or a combination.

You can adjust the frame rate to achieve the level of smoothness you want. Frame rate is the frequency (rate) at which the browser produces the unique consecutive images called frames. Although the standard frame rate in Scianimator is 330 frames per minutes (fps), which comes to 5.5 frames per second it can easily be set for up to 9 frames per second. In many cases the cost is very little more depending on the net speeds. The total number of frames has been adjusted so that the total running time for one iteration is about 5.5 seconds. You will notice a considerable difference in smoothness of the animation at the higher frame rate. However, please notice also that the file size is not significantly different. The button itself is an animation, while the motion is another animation. Scianimator has

the added feature of code-reuse. The same animation can be used over-and-over again.

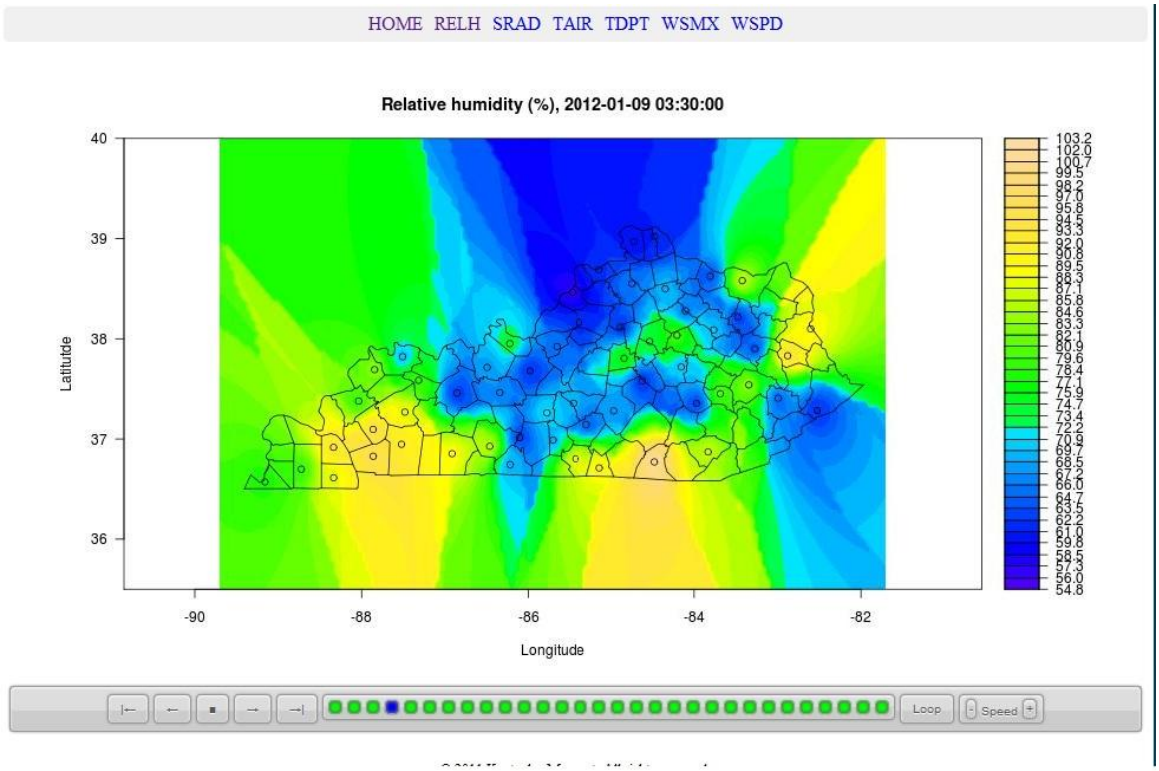

Figure 4.3: Pictorial representation of a animation in the WebInterface

Here is a piece of the Flash stage and controls for the animation above. In it you can see two keyframes indicated as dots from frame 1 and 30 (the green dots). The blue dot between them indicates the current frame in the motion created between the states at frame 1 and frame 30. To the rightmost we have a speed control bar which is used to increase or decrease the frame rate.

As you can see the timer value is the time between frames. This is input using a form variable. Hence it can be shown dynamically. Notice how the animation is accomplished. The animate function basically calls itself for each redraw. The reason for this is that every JavaScript function must complete execution before the screen is updated. Therefore, it is not possible to create the animation in a loop within a single call to the function. The animation begins and can be stopped with the stop button (third button from the left). When the animation is stopped the left most button can be used to show the foremost frame,

similarly the right most button is used to show the last or the most recent frame. Also, the second and fourth buttons from the left can be used to move the frame by one position.

This example goes a bit further in that it works in almost all the popular available browsers. The problem with that is that the positioning of layers is somewhat different in IE and mozilla, thus requiring a lot of branching to different command sets depending on the browser.

Finally, we present a simple interactive animation of the weather data. This example was created for a Kentucky Mesonet on math/computer science modelling to show electronically the experiments that led us to the above described models of dynamic data extraction and visualization. The idea is to demonstrate how the images can be set in motion for visualization from the plain raw data. It also allows the speed of animations to change. This is available online at 161.6.10.55/mesonet

4.8 Studies and Observations

We have produced different types of visualization requiring computation-intensive calculations. One of the visualizations is the effect of relative humidity shown on the Kentucky Map.

We have performed the calculations to produce the above visualizations on a single machine with 13GB random access memory, 1.6GHz processor speed, and 600GB of free hard disk space. The single machine on an average took about 47 seconds to finish the computations to produce the visualization, whereas the time taken to update the database and producing the required CSV file has taken an average of 30 seconds.

To store the weather data of all the 100 locations at any point of time it would consume 5MB of data, Which makes it 1.4 GB for a single day. The total size of the dataset if the data gets stored continuously for about one year would approximately be about 500 GB.

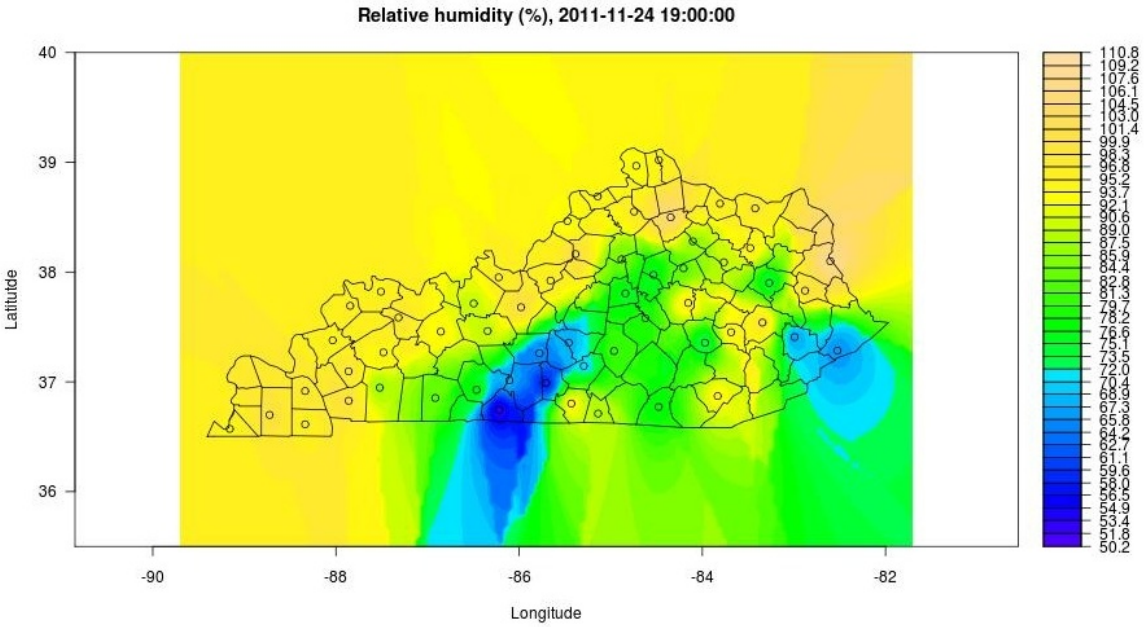

Figure 4.4: Pictorial representation of Relative Humidity

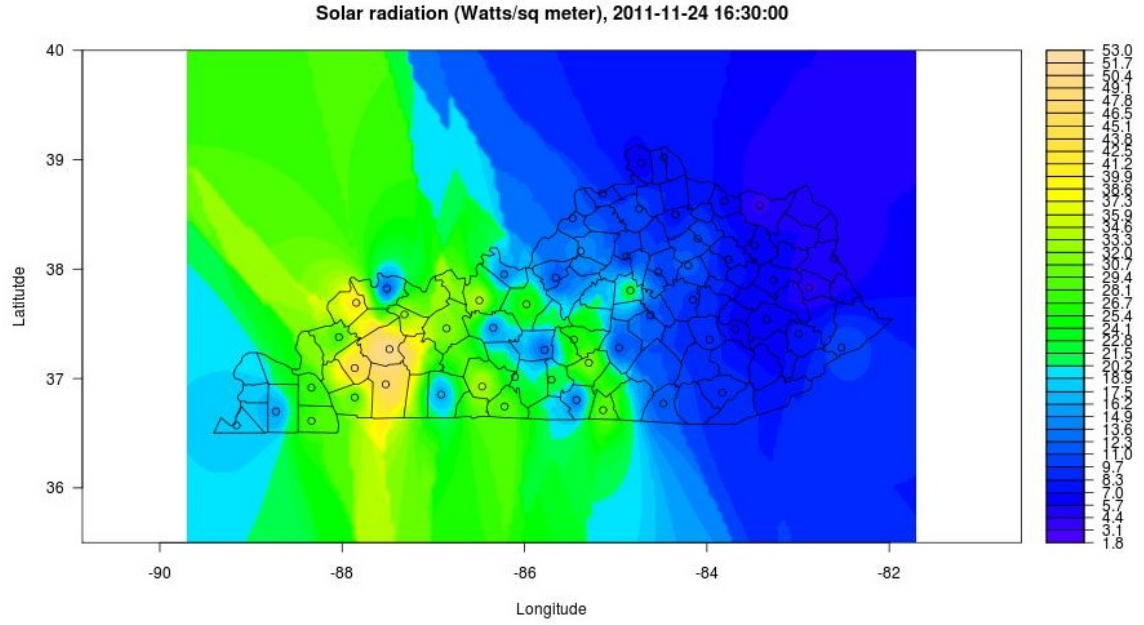

Figure 4.5: Pictorial representation of a Solar Radiation

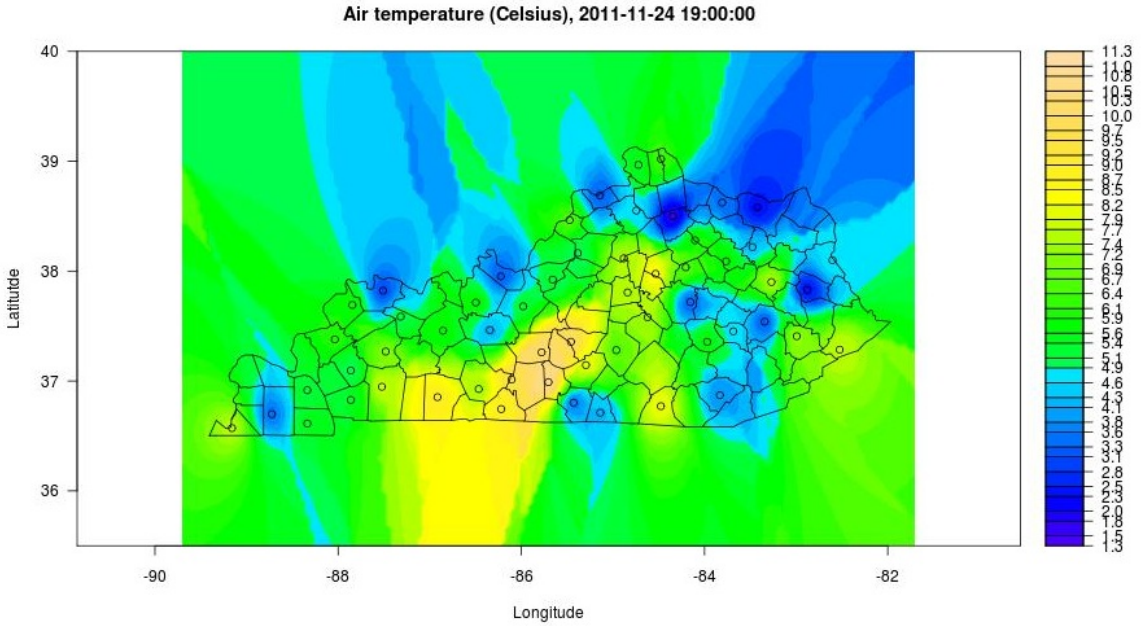

Figure 4.6: Pictorial representation of a air temperature

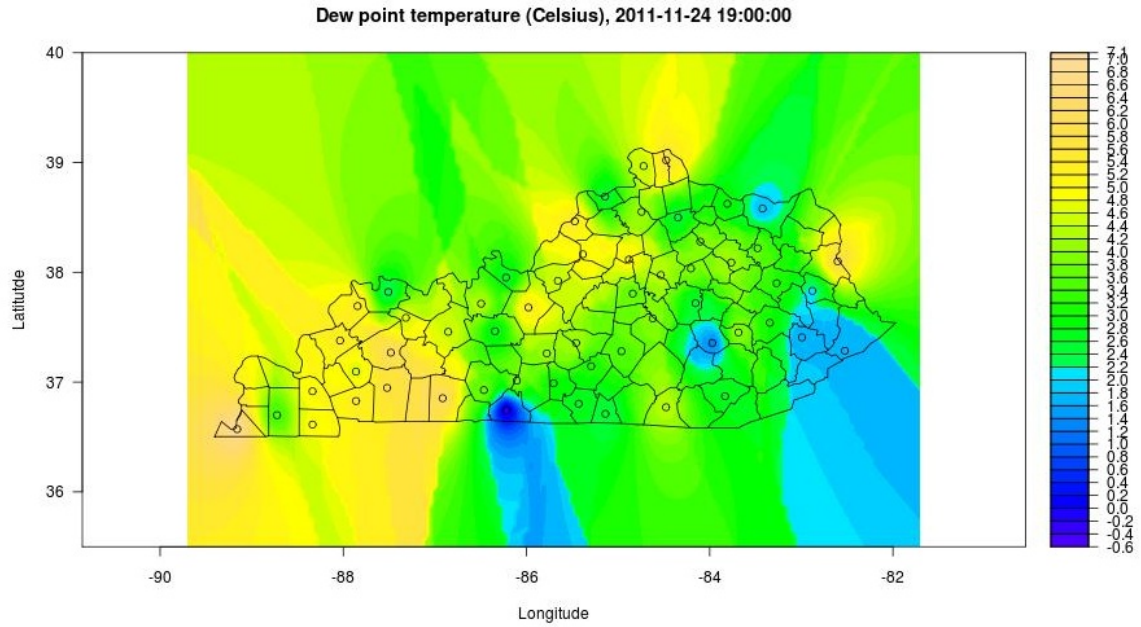

Figure 4.7: Pictorial representation of a dew point temperature

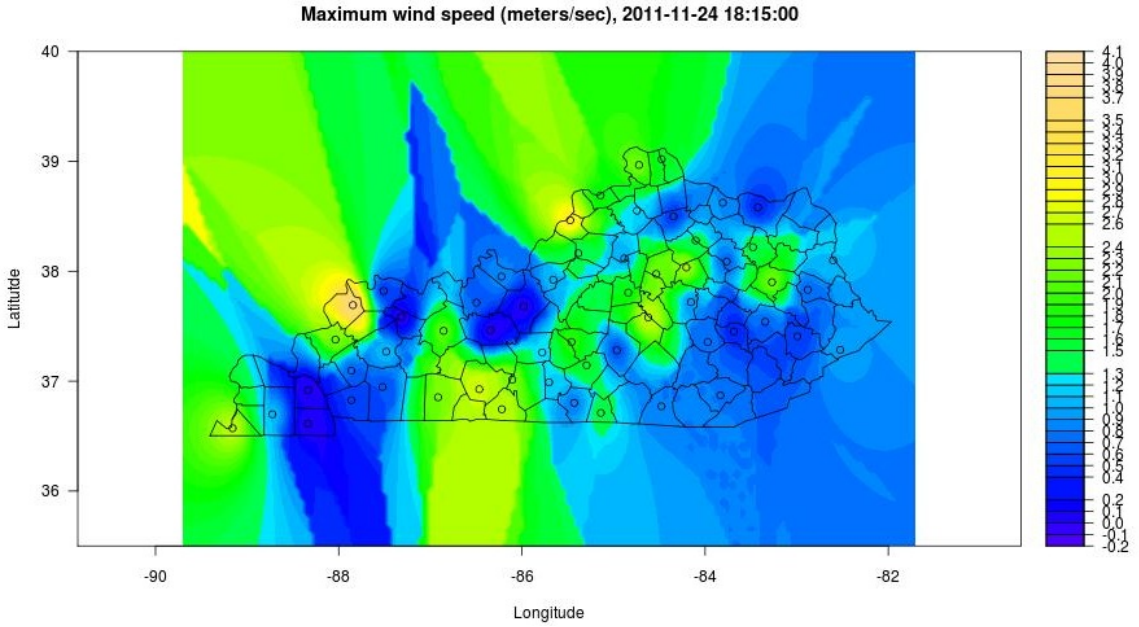

Figure 4.8: Pictorial representation of maximum wind speed

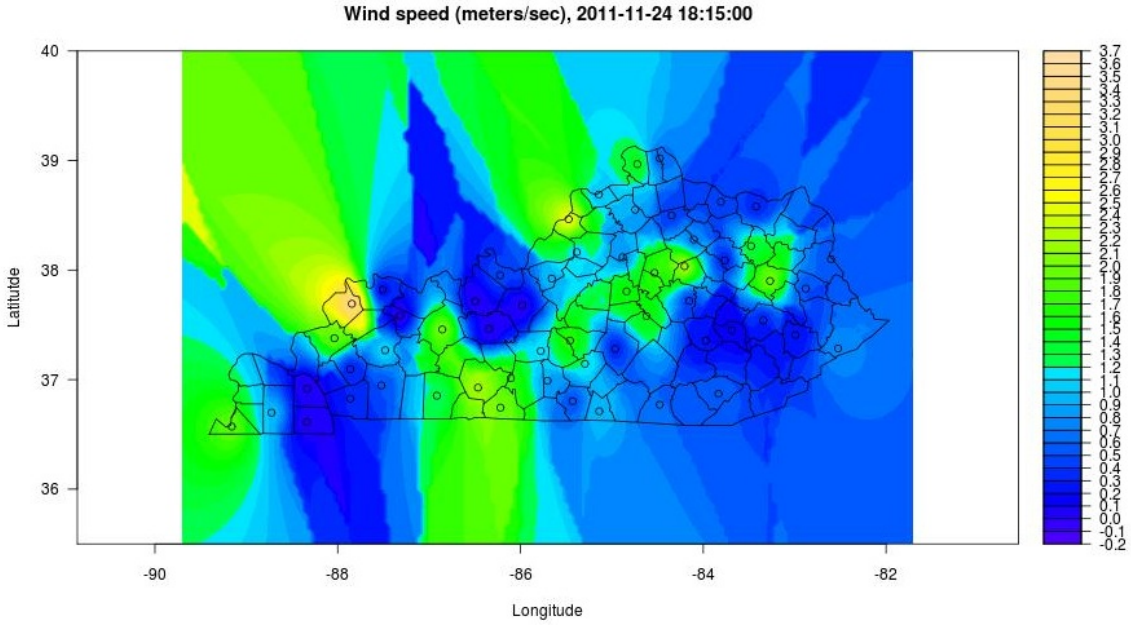

Figure 4.9: Pictorial representation of wind speed

A single machine takes considerable time to perform complex calculations on this 500 GB of data and to produce the visualization. This dataset has 60 files currently; processing all these files to produce a simple visualization on a single machine takes approximately 2 minutes in total which includes the process of database update and metadata file generation, Processing the metadata file to generate images and to produce the animations.

Since we are trying to develop a visualization web portal, there is a high probability that multiple requests to perform different calculations on this dataset can be received by the server in the same period. If one machine takes approximately 2 minutes to produce the results for one complex calculation, every user has to wait for at least 2 minutes to get the result once the job is submitted to the web server. Given the limited number of resources, user wait time might increase. If a single user submits multiple requests to produce animated visualization effects, then the response time would increase even more. If the dataset is bigger, the response time of a single machine would be prohibitively high unless appropriate measures are taken.

Therefore, to support the above-specified situations, either finding a faster machine with huge amounts of memory or combining all machines available and utilizing those resources to their fullest are the solutions.

Finally, the above application was tested in the following browsers.

1. Internet Explorer 9
2. Chrome 8
3. Safari 5
4. iPhone - Mobile Safari

CONCLUSION

5.1 Outcome

The main outcome of this thesis includes developing a general framework or work flow for dynamic data extraction, statistical analysis and visual representation of the data with application to Kentucky Mesonet data. An automated model, which contains multi-threading-capable code for intelligently and quickly performing numerous simultaneous checks, has been developed and is currently undergoing operational testing before being fully integrated into the network data flow. This system allows users to access and visualize the data regardless of the geographic location of the user and without having to install specialized software on their systems.

While the development of the evaluation framework took more time than expected, we believe that the result is worthwhile. Standards and best practices have been referenced throughout the development of the model and its application, including the design of its supporting computing, which includes a system which collects, processes, stores, and distributes data to statistical tools on a mission-critical basis. The framework, without change, can be used for a significant number of similar evaluations and, with minor change, could be used for a wider range of problems. I believe that our approach provides for a flexible and practical solution until the process of "database extraction and visualization" is improved and standardized. Also, the XML based work flow provides easy exchange of work flow process definitions between organization, and an integration tool to enable coordination of the organization's tasks.

5.2 Future Scope

As of now the system has been designed to be used only for academic and research purposes and our server may have limitations on concurrent users. If the system has to be used by many, a lot of performance considerations have to be taken care of in order to avoid the load which would result in crash. The main objectives of this project would be fulfilled when the end users would be able to see the changes of the weather in the form of animations and act accordingly. In the near future we would make sure that all the objectives of the project would be fulfilled.

Dr. Jonathan Quiton, Dr. Stuart Foster, Dr. Qi Li and the Kentucky Mesonet Modeling and Visualization Team are currently developing the second-generation weather maps which is designed for more accurate and smoother interpolations using the database and the integration tools developed in this thesis.

BIBLIOGRAPHY

[1] Brentertz. scianimator, 2010.

[2] Software Freedom Conservancy. jquery - write less do more, 2009-2011.

[3] World Wide Web Consortium. Wenning, rigo, 2003.

[4] ECOMsolutions. Disadvantages of open source software, December 2007.

[5] Stephen Few. Patterns of enterprise application architecture, Addison-Wesley,2003.

[6] The Apache Software Foundation. Apache web server, 2011.

[7] Martin Fowler. Data visualization for human perception., 2011.

[8] GBDirect. The benefits of using open source software, October 2006.

[9] D. Michael Grogan, Stuart A. Foster, and Mahmood Razaul. The Kentucky Mesonet: Perspectives on data access, distribution, and use for a mesoscale surface network. The Kentucky Mesonet, pages 1–4, 2010.

[10] I. Hadad. Establishing an open source software strategy: Key considerations and tactical recommendations. Technical report, The Linux Foundation and Contributing Editor for the Linux Journal, Nov-2010.

[11] R. Ihaka and R Gentleman. R-language, 1993 - 2011.

[12] JavaScripter.net. What is javascript?, 1999-2011.

[13] UV Ramana IIT Kanpur. Some experiments with the performance of lamp architecture. Technical report, Veritas Software, 2004.

[14] Western Kentucky University Kentucky Mesonet. Kentucky Mesonet, 2004-2011.

[15] Dr. Sham N, Kimbro S., D. Alperovitch, S. Collins, O. Gazitt, and L. Dennis. An exploration of xml in database management systems, 2001.

[16] D. Raggett, A. Le Hors, and I. Ian Jacobs. Document type definition, 1999 - Present.

[17] L. Robert Harris. Information graphics. *A Comprehensive Illustrated Reference. Oxford University Press.*, pages 1–4, 1999.

[18] E. Rusty Harold. Xml for data, May 2002 - Present.

[19] Klaas-Jan S., A. Muhammad, and Avgeriou P. Paris. The importance of architectural knowledge in integrating open source software. *Springer*, (2):142–158, 2010.

[20] W. Scacchi. Open acquisition: Combining open source software development with system acquisition. Technical report, Institute of Software Research University of California, Irvine, 2002.

[21] C. M. Sperberg-McQueen and H. Thompson. Xml schema definition, 2001 - Present.

[22] World Wide Web. Extensible markup language. *Extensible Markup Language XML 1.0 Fifth Edition*, pages 1–4, 1996 - 2003.

[23] Wiki. Data extraction, December 2007.