ABSTRACT

DATA FEED MANAGEMENT AUTOMATION : AN EMPIRICAL APPROACH

by
Dibyendu Majumder

This thesis proposes a framework for automation of data feed management and control. The framework can be classified into two broad areas i) building an ontology for a feed system and ii) developing a test suite, that is, a set of statistical models to analyze and diagnose feeds and formalize the results to feedback into the ontology. This thesis concentrates more on the latter and will only outline the course of action required for the former. Detailed descriptions of the statistical models that are part of the test suite are given and their strengths and weaknesses discussed. Practical results of implementing the test suite on a data feed system are analyzed. Various open and unexplored areas, issues and concerns are also discussed.

DATA FEED MANAGEMENT AUTOMATION : AN EMPIRICAL APPROACH

by

Dibyendu Majumder

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

January 2005

DATA FEED MANAGEMENT AUTOMATION :
AN EMPIRICAL APPROACH

Dibyendu Majumder

Dr. Narain Gehani — Date
Chair, Department of Computer Science, NJIT

Dr. Parni Dasu — Date
Member of Technical Staff, AT&T Labs Research, Florham Park, NJ

Dr. Jon Wright — Date
Member of Technical Staff, AT&T Labs Research, Florham Park, NJ

BIOGRAPHICAL SKETCH

Author: Dibyendu Majumder

Degree: Master of Science

Date: January 2005

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2005

- Bachelor of Science in Computer Science,
  Utkal University, Orissa, India, 1998

Major: Computer Science

To my parents who have been inspiring me all throughout my education.

ACKNOWLEDGMENT

Over the two semesters, my thesis advisor Dr. Narain Gehani provided me valuable and countless resources, inspiring me to attend seminars and conferences and introduced me to experts in this field at AT&T Labs Research. His assurances and suggestions had been a guiding factor throughout as I completed my thesis.

Special thanks to Dr. Parni Dasu, primary thesis advisor at AT&T Labs Research whose support from the very beginning till the end was invaluable. Without Dr. Dasu's support as well this thesis could not have been completed. Also to be mentioned is the mentoring received from Dr. Gregg Vesonder, Dr. Jon Wright and Dr. Thomas Kirk. In spite of their busy schedules, I was never denied a single request for a meeting. Their interest reflects AT&T Lab's commitment to academics and the long legacy of academic co-operation.

TABLE OF CONTENTS

Chapter		Page
1	INTRODUCTION	1
2	SURVEY OF EXISTING LITERATURE	5
3	FEED MANAGEMENT FRAMEWORK	7
4	FEED ONTOLOGY AND KNOWLEDGE BASE	10
5	TEST SUITE AND META ANALYSIS	12
	5.1 Introduction to Test Suite	12
	5.2 Introduction to Meta Analysis	13
	5.3 Detailed description of the algorithms	14
6	TEST RESULTS	19
	6.1 Charts for 1st week of June 2004	21
	6.2 Charts for 1st week of July 2004	22
	6.3 Charts for 1st week of August 2004	23
7	CONCLUSION	24
8	FUTURE WORK	25
REFERENCES		27

LIST OF FIGURES

Figure		Page
3.1	The Automation Framework	9
4.1	A Interaction of meta analysis component with knowledge base ...	11
6.1	Hample Identifier analysis chart for 1st week of June 2004.....	21
6.2	Six sigma analysis chart for 1st week of June 2004.....	21
6.3	Hample Identifier analysis chart for 1st week of July 2004.....	22
6.4	Six sigma analysis chart for 1st week of July 2004.....	22
6.5	Hample Identifier analysis chart for 1st week of August 2004.....	23
6.6.	Six sigma analysis chart for 1st week of August 2004.....	23

CHAPTER 1

INTRODUCTION

There is very little formal work on data feed management. Database research has primarily focused on building warehouses and optimizing queries to extract data from these warehouses. Feed management is a difficult problem, due to the complexity and size of feeds, their interpretation, their inter-dependence and the ad hoc nature. A brief introduction to data feeds can be found in [3]. Market demand for feed is very high and increasing. Some typical area where the relevance of feed is very high are trading data for financial systems, billing data for telecom companies, etc. The data feed system being tested receives millions of files every day. File sizes vary from a few kilobytes to hundreds of megabytes. Constant pressure on disk space makes tracking down and removing 'dead files' an important part of the process. There is little knowledge about the nature of feeds, specially feed traffic. Feed traffic varies drastically from no transmission during weekends to very high traffic density during the day hours. Each feed type has its own pattern, some may show no pattern and then there are third party feeds that keep changing and there is very little available knowledge to anticipate these changes. A growing interest now is to build an ontology for data feeds. As first phase of development, feeds are being classified according to the results obtained by the statistical algorithms.

Feeds are very ad hoc in nature. Volume of feed traffic may vary drastically with time. For example, telephone calls may be very high during an emergency or during the

holidays. Comparatively fewer calls are generated during night hours. Such drastic variations may crash a 'billing data' feed system. On the other hand being too cautious may lead to lack of efficiency. Similar examples can be shown for trading data. The volume of trade varies day to day. An efficient data feed handler should be able to anticipate such changes and adapt itself accordingly. When the feed traffic is lower or stable it can allocate more resources to analysis. With time a feed may be termed 'dead' which requires its removal or archival. Bigger feed sizes may lead to a constraint on space. Missing or corrupt feeds may be hard to detect because of their dynamic nature. Feed users vary widely and a good synchronization should exist between the requests and service. For example, the feed handler being tested receives ad hoc analysis requests such as transferring a data set (bundles of files) from one cluster to another where each cluster is devoted to a specific application. There should be a method of defining a data set as well as method for transferring, verifying integrity and validations of these data sets. A good synchronization should lead to efficient handling of such requests.

A typical data feed system consists of three components receiving feeds from various sources, feed management and servicing feed users. While data are being received the feed handler should be able to detect or anticipate traffic volume and pattern and adapt itself accordingly so that all data is received properly. The handler should have the ability to detect missing or corrupt data and to take necessary action. The feed handler should be able to organize, classify and distribute feeds efficiently within soft time constraints, flush out(deleting or archiving) old data, provide feed users with intelligent information about feeds, trace feed movement and feed status and monitor feed life cycle. Feed consumers are applications that require data feeds for execution. Some problems

encountered here include ensuring required files exists, composing a list of files to be transmitted, validating the lists, trace down files that have finished usage, etc. The design patterns talked about in [1] may be relevant here. Meta analysis can be of help here. Meta data is information collected by the statistical algorithms and meta analysis is analysis done on such information. Meta analysis gives us a broader picture of the defect patterns, the severity of a defect and can help co-relate different feed types. Classification based on meta analysis enriches feed knowledge base giving feed users a multi dimensional view of the feeds.

The feed nature and problems explained above indicates the requirement of a common language or methodology that every feed user understands. Effort is on to build an ontology as part of the automation framework. The ontology will help formalize and organize much of the concepts in the feed domain. A suite of statistical models has been developed to analyze feeds and extract patterns and hidden information. At present, capacities of the models are limited but efficient auto regressive models are foreseen in future as experience working with these models increase. Application of the results to real life test scenarios is also discussed.

First section of this thesis will outline the broad framework proposed. Various components of the framework and the most relevant components will be stated. Knowledge base representations and implementation methods will be discussed. Later sections will give a detailed description of the test suite that is being developed. The various algorithms that constitute the test suite will be discussed and their strengths and weaknesses stated.

In later sections the results of application of the algorithms to a data feed system – NINGAUI, will be discussed. The last sections will discuss the conclusions drawn, some of the open areas in this domain and a plan of the future course of action will be outlined.

CHAPTER 2

SURVEY OF EXISTING LITERATURE

Semantic correctness of data feeds is discussed in [Orna2]. Feed handler architectures are discussed in [Roodyn3]. An interesting section in [Orna4] is the description of the technique tool kit. Output of the toolkit is used as part of a specification based system to detect anomalies. It describes various statistical algorithms some of which may be relevant for us. Similarly various statistical algorithms are also discussed in [Agarwal5].

The test suite being developed is very similar to the 'technique tool kit' stated in [Orna4]. But the author doesn't discuss in detail about the standard deviations and its not clear how they are calculated. In contrast, the tool looks at historical data to derive the values. Output of the tool kit in [Orna4] is used as part of a specification based system to detect anomalies whereas ours use the results of test suite not for detecting anomalies only but to garner data for meta analysis. It's the first step of a two step process.

This thesis does not focus on the semantic correctness of data feeds as in [Orna2]. It also differs from [Roodyn3] as it does not talk about building a framework for feed integration but only recognizing that integration may be a part of the total automation process. Therefore the customizable components and Component Object Models discussed in [Roodyn3] there as architecture styles are of particular interest to us.

Though considerable literature exists for data streams most of them are not applicable to feed systems, and that justifies a fresh investigation. In [Agarwal5] some new techniques are discussed to determine trends in evolution of fast data streams. Although one objective is to discuss *data evolution* using visual and diagnostic tools this

thesis specifically targets high speed data streams. A lot of work being done on continuous queries of data streams as in [Widom6] but they fall out of context with reference to feeds. Models and issues discussed in [Motwani et al 7] are applicable for streaming feeds but not feeds in general.

CHAPTER 3

THE FEED MANAGEMENT FRAMEWORK

Figure 3.1 shows the proposed feed framework being developed. The circular entities *test suite, meta data and meta analysis* components are the ones this document will describe in detail. The framework consists of three logical components feed receiving component, feed management component and the feed user interface component. The feed receiving component actually receives the feeds and after initial validation stores the feeds. As shown, the test suite is a part of the feed receiving component. The objective is to detect problems as early in the process as possible. The feed management component is primarily where all feed management logic resides as stated before like tracing and validation of feeds, tracing feed life cycle, archiving old data, etc. Feed user interface is the logical interface to applications that uses the feeds. Every application may need its own interface that customizes the feeds according to individual requirements. This is one place where its hoped building an ontology is going to be most useful.

A good framework is one that allows smooth integration of new users. Feed repository may be any relational data base. Meta data can be stored as simply as a text file. The knowledge base represents knowledge about the data feeds, system resources, applications using the feeds etc. The aim is to capture knowledge about the whole environment related to feeds. The knowledge base may be accessed by any component of the framework. Implementation and subsections of the knowledge base and its underlying ontology will be discussed in the next section. The meta analysis component is plugged

into the framework and primarily used for enriching the knowledge base as results are fed back.

The feed management framework can be classified into two broad categories. The first category consists of the knowledge base, the ontology that represents the knowledge, method of implementation and architecture, reasoning mechanism, etc. The second component consists of the test suite, meta data and the meta analysis component. There is no contribution to the actual framework as noticed. The author is more interested in demonstrating the usefulness of building an ontology for data feeds and it does not matter how the results are used by a framework. The following two sections will describe the categories mentioned above in further detail.

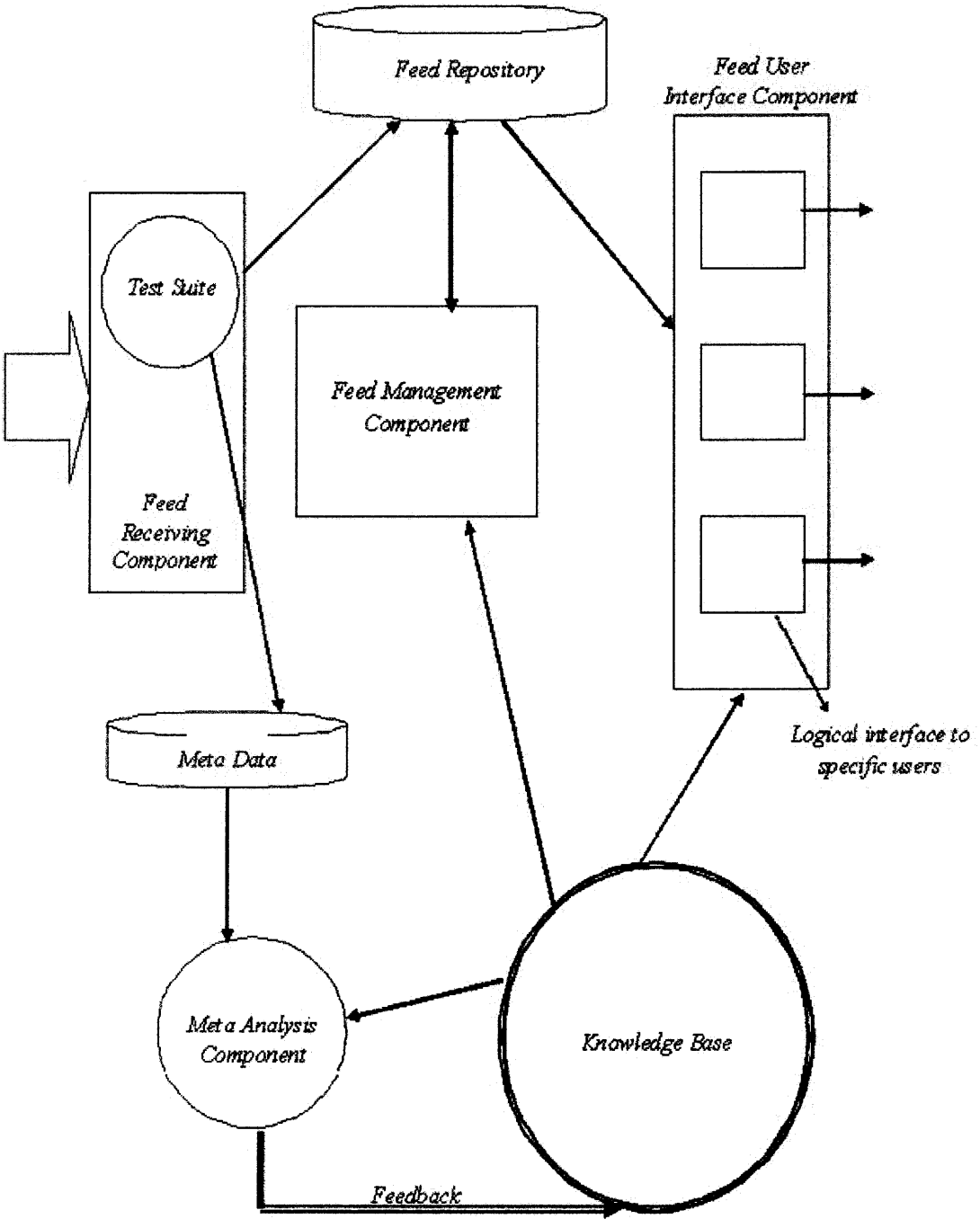

Figure 3.1 The automation framework.

CHAPTER 4

FEED ONTOLOGY AND KNOWLEDGE BASE

There exists several motivations for building an ontology for data feeds. Formalizing the whole feed literature helps in better understanding and a multi dimensional view of feeds. Classification of feeds by analyzing their nature and behavior leads to better understanding of feeds and easier anticipation of changes. An ontology captures the dynamic nature of feeds spontaneously.

Figure 4.1 shows the various components of the knowledge base in detail. There exists ontological elements for various components. An ontology for system resources represents the system and hardware environment such as cpu status, memory status, disk usage, queued resources, processing load, etc. An ontology for applications running represents the requirements for various applications like amount of cpu time requested, memory requirements, etc. The data feed ontology represents all knowledge about data feeds like feed types, feed nature, feed status, etc. A section of the data feed knowledge will be updated by the meta analysis component. A formal representation method will standardize the process of interfacing applications with the feed system. A typical problem faced by feed users is there understanding of the feed nature and behavior is minimum, specifications keep changing and its difficult to maintain specifications simultaneously with changing data feeds.

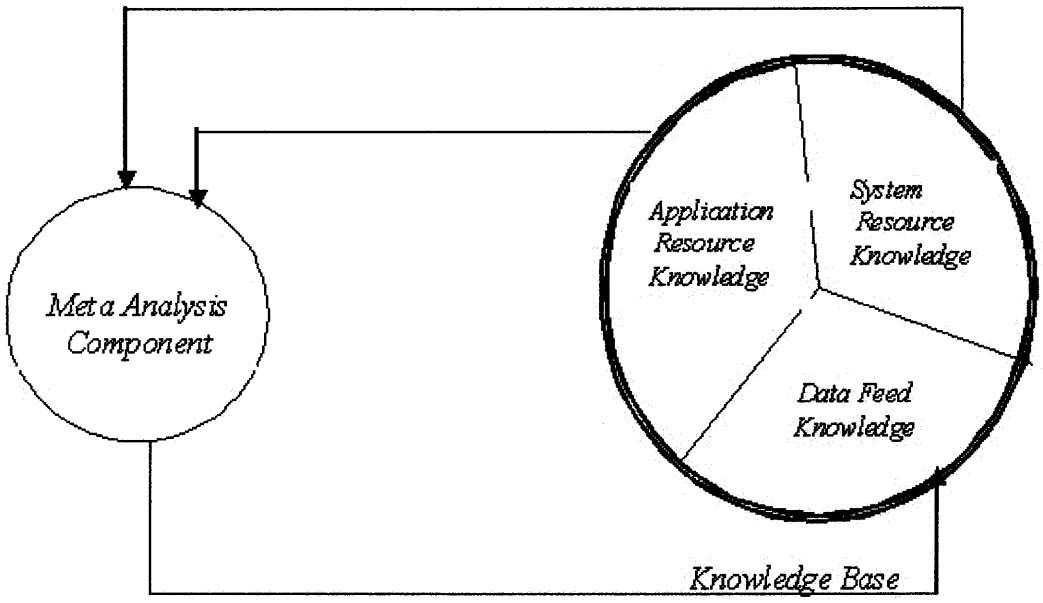

Figure 4.1 A Interaction of meta analysis component with knowledge base.

Though a small survey has been done to choose the methodology [8] for building an ontology, the general idea is to start building from scratch without actually getting tied down to any particular representational structure or implementation methods. At some point of time as a structure em erges a decision should be taken to map it to a particular representation or implementation. Not much progress is made in this direction yet and this document will not discuss any more about this section of the framework. Rest of the document will focus on the test-suite, meta data and meta analysis components.

CHAPTER 5

TEST SUITE AND META ANALYSIS

Several algorithms have been developed to analyze the feeds. The two parameters that are focused on initially are traffic volume and file sizes. The test suite is being optimized by adding more efficient algorithms.

5.1 Introduction to the Test Suite

The algorithms look at past data to understand and evaluate current data. Historical data of the last three months is analyzed and mean and other boundaries evaluated. This then is applied to the current data. All violations to the cut off limits are then marked as 'anomalies' and logged for further analysis later. Its assumed that data traffic varies greatly for weekends and weekdays, so they are analyzed separately. Equal priority given to all the algorithms, i.e., severity of a defect is the same irrespective of the algorithm that has detected it.

A brief description of what is meant by an anomaly or defect is given here. If any measurement at any given point of time violates the anticipated ranges for that time, it is flagged as an 'anomaly'. Every anomaly has a time stamp attached to it. Anomalies are calculated for a given day. This thesis investigates various ways of defining an anomaly other than associating it with a time stamp.

At present the test suite contains the following algorithms

- Hample Identifier analysis
- 5 percentile cut off analysis
- 10 percentile cut off analysis
- 6 sigma analysis
- Quartiles (5^{th} & 95^{th})

A detailed description of the algorithms are given below in another section.

5.2 Introduction to Meta Analysis

The statistical models analyze the data feeds and the results are logged as defects or anomalies, shown in Figure 3.1 as meta-data. These defects are then analyzed again to detect hidden patterns, classification possibilities, etc. This phase is called as meta analysis. Meta analysis is an important part of knowledge extraction, the first level analysis is only an auxiliary to collect data for an exhaustive meta analysis. At this point very simple meta analysis is being done which is termed as 'voting mechanism'. The number of votes for a given defect is being plotted against time, i.e. the number of algorithms that had voted for a defect. The assumption is that severity of an anomaly is dictated by the number of votes. Again, equal priority is being given to all the algorithms but in real life algorithms may be prioritized based on context or scenario. Finally, the test results are evaluated by a human expert and weighed accordingly. In the future, call back functions may be implemented as part of the meta analysis component. Human experts can encode their own validations using first order predicate logic which will be invoked during meta analysis. Experts are notified by automated e-mail messages when

their attention is required. Results of the meta analysis will be published as test results. As understanding of the system increases, the meta analysis module will be enriched by adding several other analysis methods.

As shown earlier in Figure 4.1 results from meta-analysis will be fed back into the ontology enriching the data feed component. At present the focus is on how data feeds can be classified based on meta analysis results. In a complete automation framework, the feed system should be able to adjust itself according to the results of meta analysis. Though the present state is far away from such a goal, this is what is being propose to achieve and hence, meta analysis is one of the most important part of the framework.

5.3 Detailed Description of the Algorithms

Hample Identifier analysis

Applicable for normalized data. Unlike other algorithms deviation is calculated twice. Hence, its more stringent than others in terms of outliers. The maximum, minimum and medians are -3, 3 and 0, respectively.

For a given array, the array is first sorted and the median calculated. Then the absolute deviation of every point from the median is computed. So at this point there are a set of deviations. Then, it calculates the median of the deviations. A correction factor `K = 1.4` is then applied. How the correction factor is chosen will not be described in detail. Let's say a point x_i is an outlier if `abs(x_i-M)/K > 3`. It will illustrated with an example:

given a sorted array(24 elements)

32, 37, 37, 38, 39, 39, 39, 40, 41, 42, 52, 53, 63, 71, 79, 80, 81, 81, 82, 85, 87, 89, 97

compute the median, M

58

compute the absolute deviation of every point from the median

5, 5, 6, 13, 16, 17, 18, 19, 19, 19, 20, 21, 21, 21, 21, 22, 23

23, 24, 26, 27, 29, 31, 39

compute S, the median of the deviations : 21

Apply the correction i.e. K = 1.4826 * S : K = 1.5 * 21 = 31.5

Subtraction factor : K * M = 31.5 * 58 = 1827

A point x_i is an outlier if abs(x_i-K*M) > 3

The six sigma algorithm

The objective of Six Sigma Quality is to reduce process output variation so that on a long term basis, which is the customer's aggregate experience with the current process over time, this will result in no more than 3.4 defect Parts Per Million (PPM) opportunities (or 3.4 Defects Per Million Opportunities – DPMO). A variation of the process is measured in Standard Deviation(Sigma) from the Mean. The normal variation, defined as process width, is +/-3 Sigma about the mean. This thesis uses the latter which is the standard method of calculation.

Notation

0 Minimum value (Same as MIN())

1 1st quartile - 25th percentile

2 2nd quartile - 50th percentile (Same as MEDIAN())

3 3rd quartile - 75th percentile

4 4th quartile - 100th percentile (Same as MAX())

The following is the algorithm used to calculate QUARTILE():

Find the k^{th} smallest member in the array of values, where:

K = ((quart/4)*(n-1)) + 1

Where

quart = value between 0 and 4 (quartile)

n = number of values in the array

Note

If k is not an integer, truncate it but store the fractional portion (f) for use in step 3.

Find the smallest data point in the array of values that is greater than the kth smallest i.e. the $(k+1)^{th}$ smallest member.

Now, interpolate between the k^{th} smallest and the $(k+1)^{th}$ smallest values:

Output = a[k]+(f*(a[k+1]-a[k]))

Where

f = fraction obtained from K noted above

a[k] = the k^{th} smallest

a[k+1] = the $k+1^{th}$ smallest

5% trim analysis

This algorithm is very similar to the six sigma algorithm. In this algorithm the sigma is re-calculated after trimming the upper 5% and the lower 5% of the values. This is equivalent to removing the outliers and analyzing the scenario.

10% trim analysis

It is very similar to 5% trimmed analysis mentioned above except that the trimming occurs at 10%. The advantage of both these algorithms is it gives us an idea about the degree of severity of an outlier. If an outlier persists even after trimming twice then it is marked as of high severity to indicate that it needs to be analyzed. An outlier that persists even after 5% trimming is marked as medium severity. Outliers that are removed by trimming are considered as less severe.

Quartiles (5^{th} & 95^{th})

This algorithm is used to give a clear picture of the distribution of data rather than detecting outliers. Outliers detected by this algorithm is considered not as defects, rather data that is sensitive, prone to errors and possibly flagged as errors by other algorithms like Hample. The quartiles give us a clear picture of the raw data, as it is.

The formula for calculating the quartiles is a modification of the six-sigma formula. The original formula is mostly untouched except for calculating K.

Find the k^{th} smallest member in the array of values, where:

K = ((quart/100)*(n-1)) + 1

Where

quart = 5(5^{th} percentile) or 95(95^{th} percentile)

n = number of values in the array

Note

If k is not an integer, truncate it but store the fractional portion (f) for use in step 3.

Find the smallest data point in the array of values that is greater than the kth smallest i.e. the $(k+1)^{th}$ smallest member.

Now, interpolate between the k^{th} smallest and the $(k+1)^{th}$ smallest values:

Output = a[k]+(f*(a[k+1]-a[k]))

where

f	= fraction obtained from K noted above
a[k]	= the k^{th} smallest
a[k+1]	= the $k+1^{th}$ smallest

CHAPTER 6

TEST RESULTS

This section discusses the result of implementing the tool on a real life data feed system. NINGAUI is a data feed handler built and owned by AT&T for analyzing billing data. NINGAUI receives 30 GBs of compressed data from approximately 100 individual data feeds every day. Because of the nature of applications served by the system, and considering the fact that its analyzing financial data the quality of data feed received by NINGAUI requires very strict monitoring standards. For example, NINGAUI has a requirement that no data is ever lost or corrupted. The aim is to understand these data feeds, how they behave with time and the patterns generated.

A single data feeds is first chosen for analysis. Sum of file sizes of files received every hour is plotted against time. The medians are calculated using three months data- April, May, June. This is termed as the historical sliding window. Plots are generated for the first and second weeks of June, July and August. Here only the charts corresponding to Hample Identifier analysis and 6-sigma analysis are shown as these two are more interesting and show a definite pattern.

Some observations with reference to the charts are as follows :

- Each chart shows seven day data
- The x-axis shows the file size averages plotted against time on y axis
- Outliers are values that crosses limits or bounds
- The seven days are either weekdays or weekend

- Most of the curves are normalized. This is based on the underlying assumption that record volume increase as day progresses and decreases at the end of the day

For the first week of June, Hample Identifier (Figure 6A) detects a couple of outliers in the mid-region. This indicates that outlier have occurred at the peak period when the record volume is high. The 6-sigma analysis for the corresponding period (Figure 6B) does not show any outliers. However the points corresponding to the outliers in the earlier diagram does approach very near to the medians. In all data for the first week of July looks sound and well within control.

Charts for the first week of July leads to some interesting observations. The data is nowhere within the boundaries. The actual data has gone drastically below the lower limit for both the charts. Again the deviation is maximum during the peak hours. It means the traffic is much less than expected during this period. This continues for four days. An outlier of this degree requires further verification by experts. Having talked to the experts our first hand observation is that its not a known case and the experts suspect some problem with file logging or tracking down data movement.

Things improve in August. The Hample identifier detects (Figure 6E) only one outlier though its not supported by 6-sigma (Figure 6F). Otherwise the data is very well contained in limits. Plotting for the month of August indicates that the system has recovered from its aberration in the month of July. Further investigation is required at this state from human experts.

6.1 Charts for 1st week of June 2004

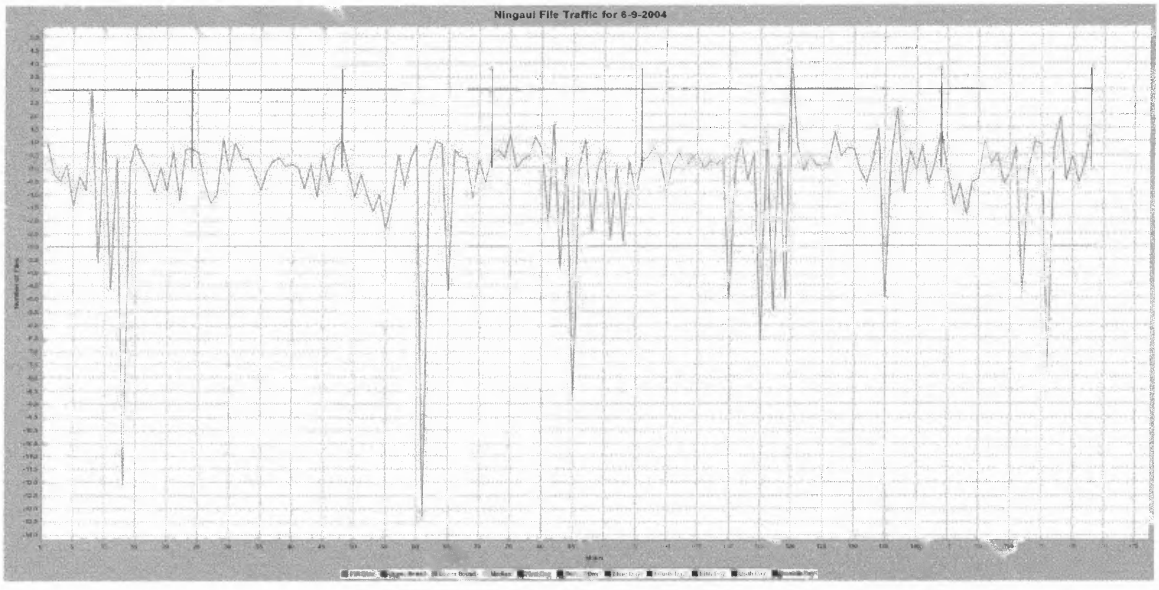

Figure 6.1 Hample Identifier analysis chart for 1st week of June 2004.

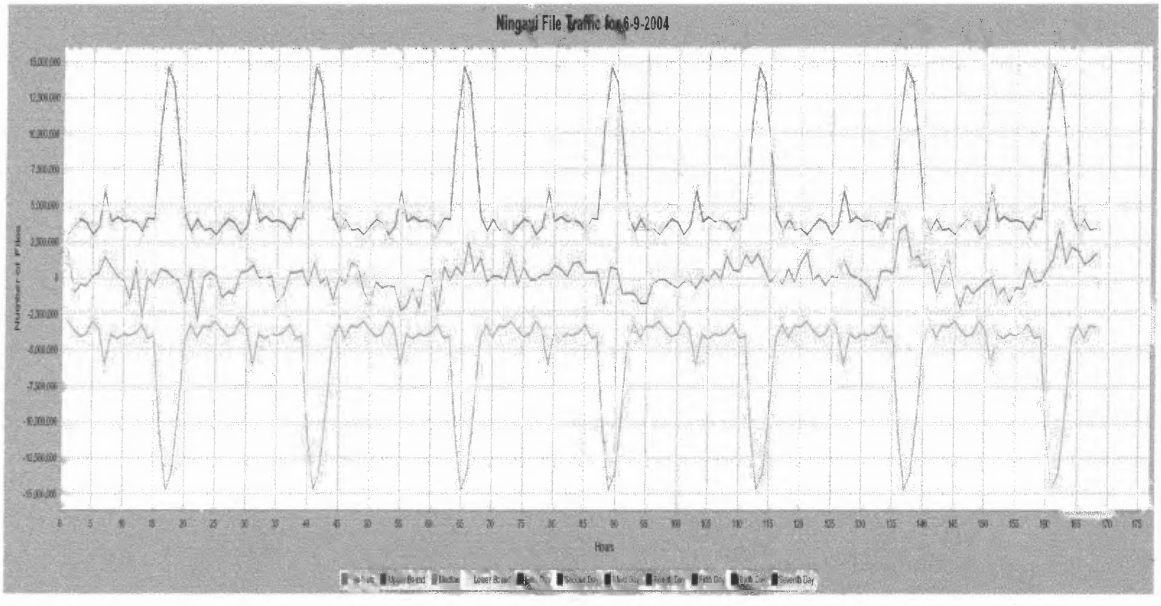

Figure 6.2 Six sigma analysis chart for 1st week of June 2004.

6.2 Charts for 1st week of July 2004

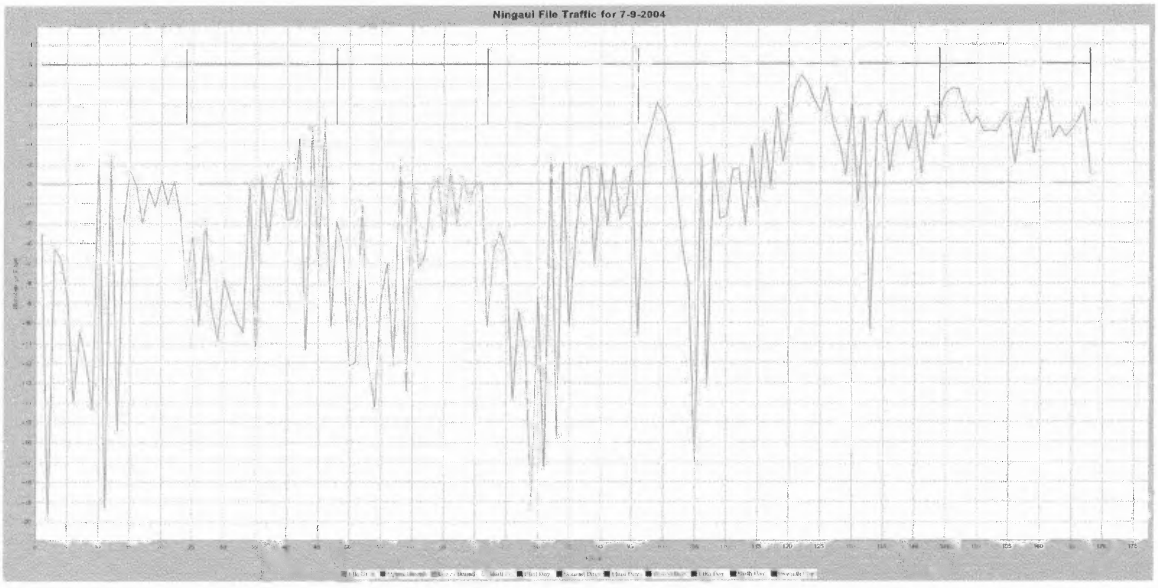

Figure 6.3 Hample Identifier analysis chart for 1st week of July 2004.

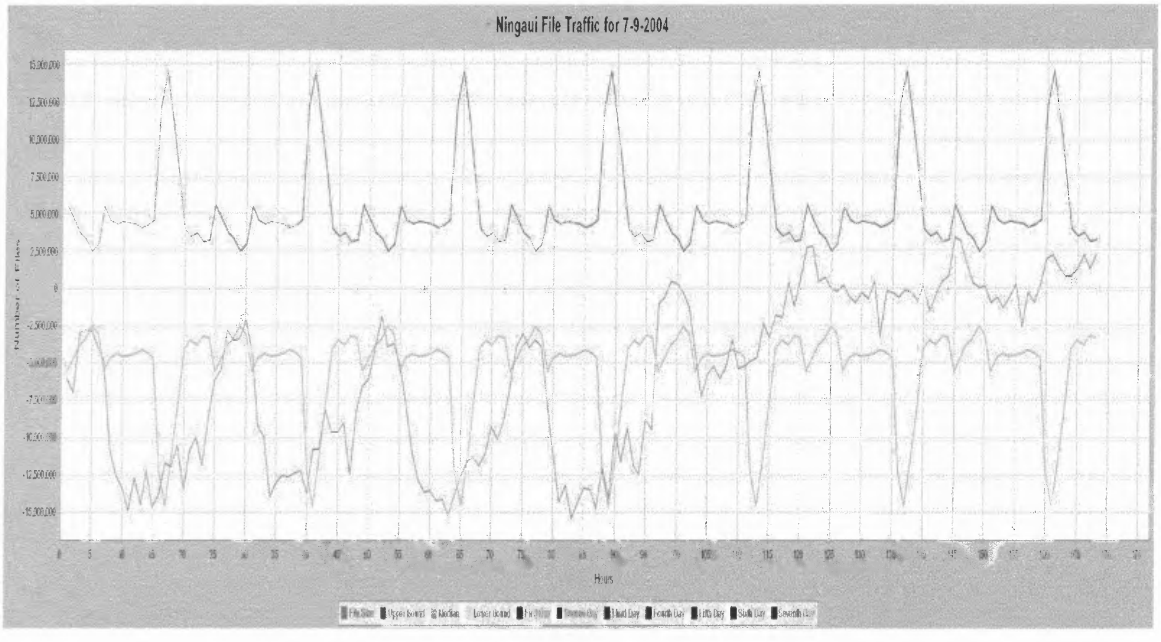

Figure 6.4 Six sigma analysis chart for 1st week of July 2004.

6.3 Charts for 1st week of August 2004

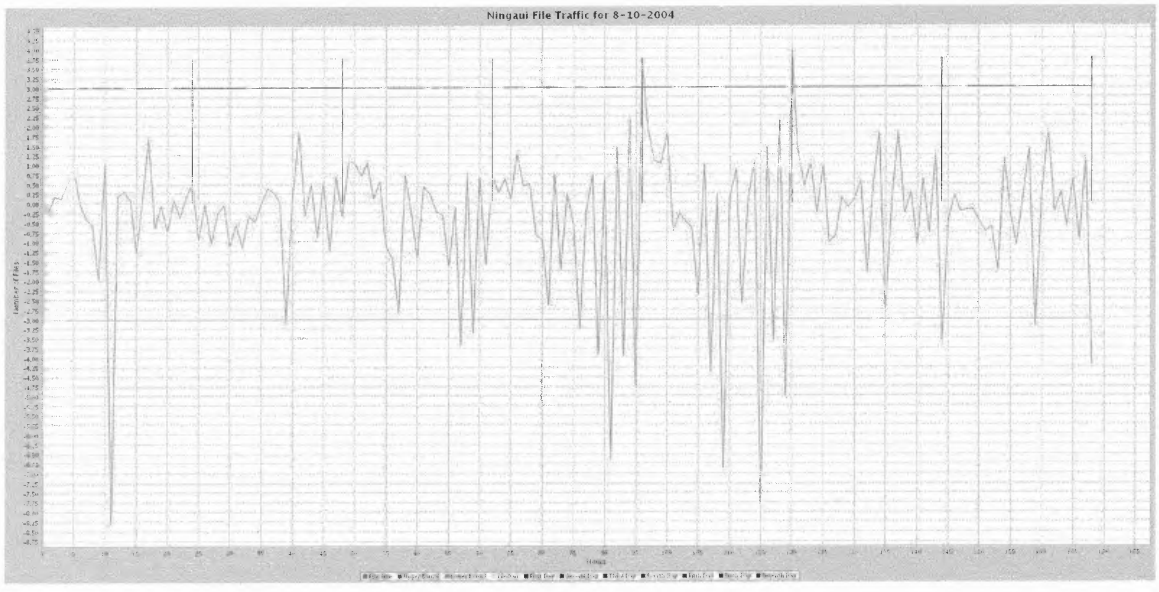

Figure 2.5 Hample Identifier analysis chart for 1st week of August 2004.

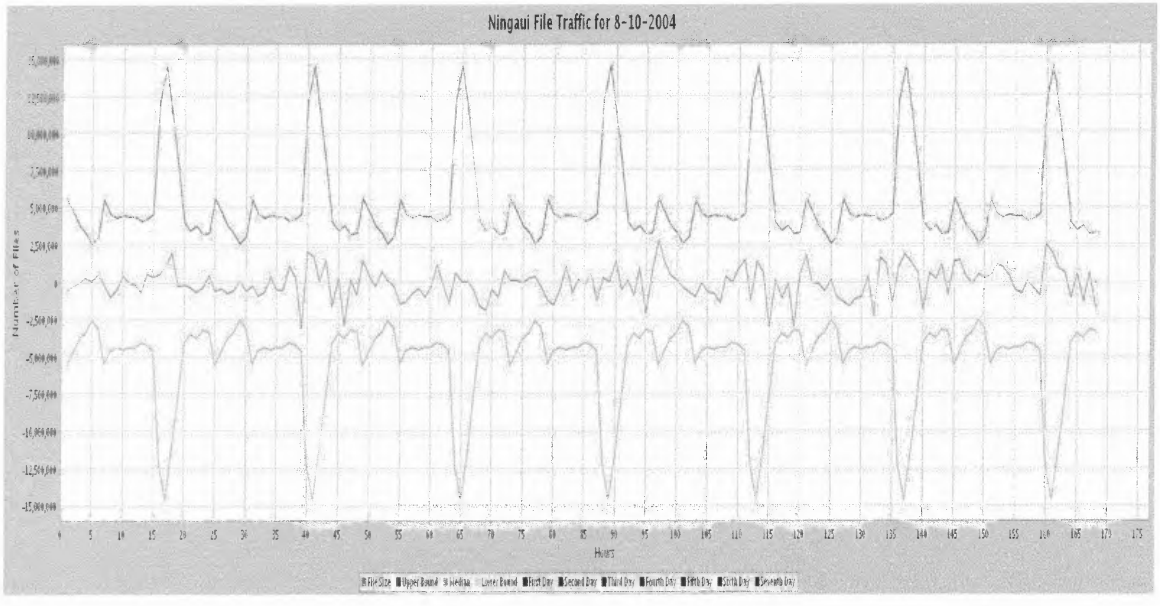

Figure 6.6 Six sigma analysis chart for 1st week of August 2004.

CHAPTER 7

CONCLUSION

An outline of the proposed feed management framework is first given. A very brief introduction to the knowledge base and related components is given. The thesis then focused on the test suite and meta analysis. It then discussed outlier detection and how it helps with meta analysis. A detailed description of the statistical algorithms are given. The tests shown are simple but effective. Defects detected early in the process gives ample time to control before they are passed on to the feed consumers. Human experts are automatically notified in case problems arise which leads to further investigation of the feed thus detecting problems before they enter into downstream user applications. Moreover, outlier detection is a step towards meta analysis which is supposed to be the primary strength of the automation framework. The results of implementing the test suite on a real life feed handler is shown. Test results showed how problems are detected in feed traffic during one month.

CHAPTER 8

FUTURE WORK

This section discusses various areas that remain to be explored. Only a very small section of the framework has been worked upon. Specifically the focus was on the first part of the framework but for second part, developing the ontology, most of the questions are yet to be answered.

Current work in meta analysis is just a proof of concept. In particular the effort is on searching for criteria that can help classify data feeds across various dimensions. Such knowledge will help the feed user visualize the feed data from different contexts. This in turn require a lot of communication with the ontology that deals with systems resources, etc. It is yet to be explored.

It is being investigated how to classify or prioritize historical data to generate even more accurate results. In other words, studying the learning mechanisms of the models and how to improve them. These models are supposed to be intelligent with feedback and auto regressive features and they are supposed to replace human experts with time. Every such model should have a set of information associated with them like features, characteristics, past results, degree of accuracy, how well suited to a particular environment etc. Such knowledge will only be accumulated over time. It's a part of the learning process. One aim of this work is also to explore the possibility of formalizing such acquired knowledge and make it a part of the ontology. Another aspect is the

implementation architecture for this learning process. The possibility of interfacing the statistical models with an agent architecture is being explored.

A couple of new algorithms may be added to the suite. But more important is the understanding of the algorithms. First it is needed to know the behavior of these algorithms under particular scenarios and test their consistency. Second, to be able to correlate the results of various algorithms against each other.

Looking at the past data to calculate the limits, its termed the sliding historical window. The behavior of the medians against various sliding windows is calculated to come up with the most efficient one. Further work required to define how to chose the best sliding window. It is believed prioritizing the historical data will lead to better results. Every day/week will be associated with a weight. The older the day/week, lesser will be its weightage. Thus older the day, the lesser role it plays in dictating the value of the medians. It is not yet implemented that.

This document discusses about the statistical algorithms included in the test suite. However the test suite itself is only a part of the framework whose main objective is to enrich the ontology by analyzing the feeds. Primary focus is not on detecting defects, rather enriching the knowledge base. Very little work has been done in that direction and in future It is hoped to come up with a specific knowledge base representation architecture.

REFERENCES

1. Raman Kannan. *Managing Continuous Data Feed with Subscriber/Publisher Pattern.* OOPSLA '95.

2. Orna Raz, Philip Koopman, Mary Shaw. *Benchmarking Semantic Availability of Dynamic Data Feeds.*

3. Neil Roodyn, Wolfgang Emmerich. *An Architectural Style for Multiple Real-Time Data Feeds.* 1999.

4. Orna Raz. *Helping everyday users find anomalies in data feeds.* 2004.

5. Charu Agarwal. *A framework for Diagnosing Changes in Evolving Data Streams.* 2003.

6. Babu & Widom. *Continuous Queries over Data Streams.* 2001.

7. Babcock, Babu, Datar, Motwani & Widom. *Models and Issues in Data Stream Systems.* 2002.

8. Lopez. *Overview of methodologies for building ontologies* 2000.