Real Time Failure Prediction in Electrical Submersible Pumps

Hayden Schultz
TIBCO Software
hschultz@tibco.com

Richard Tibbetts
TIBCO Software
tibbetts@tibco.com

Ujval Kamath
TIBCO Software
ukamath@tibco.com

Michael O'Connell
TIBCO Software
moconnel@tibco.com

ABSTRACT

As oil fields age, artificial lift systems are required to maintain production levels when reservoir pressures get too low. Failure of an ESP in a well can stop production or even lead to a dangerous event. ESPs are fitted with downhole monitoring units that transmit streams of data back to the surface including: motor temperature, pump intake pressure, intake temperature motor vibration, and motor current.

This paper describes a system to monitor large, geographically diverse arrays of oil wells with ESPs. Sensor measurements are transmitted, normalized, and integrated using a distributed communications network and stream processing system. Analytic models for predicting failure are created offline using historical data analysis, and executed in real time against live sensor data using the stream processing system. When failure is predicted, alerts are dispatched to both a live data operator console and a visual analytic platform. An implementation is described on a major North American oil production system using the TIBCO Fast Data Platform including TIBCO StreamBase CEP, TIBCO Spotfire Analytics, and TIBCO Live Datamart.

Categories and Subject Descriptors

Sensor Networks. Maintainability and Maintenance. Failure prediction. Data Analytics. Information Integration. Stream management. Visual Analytics.

General Terms

Algorithms, Management, Measurement, Performance, Design, Economics, Reliability, Experimentation, Human Factors.

Keywords

TIBCO StreamBase CEP, TIBCO Spotfire, TIBCO Live Datamart, Electric Submersible Pumps.

1. INTRODUCTION

In 95% of global oil wells, artificial lift systems pump the oil from the ground when reservoir pressures get too low. About 10% of these 900,000 artificial lift systems globally are Electric Submersible Pumps (ESPs). The ESPs are supplied by different manufacturers and are operated by different service organizations in very different environments around the globe – from the constant heat of the Middle East to severe winters in the northern US and Canada.

The ESPs are fitted with downhole monitoring units that transmit streams of data back to the surface including: motor temperature, pump intake pressure, intake temperature motor vibration, and motor current. These data can be used to develop leading indicators of failure and shutdown conditions. Some leading indicators are simple rules e.g. if motor temperature increases by a large amount, an electrical short can be induced and the pump automatically shuts down. Other leading indicators are more subtle combinations of changes in pressure, current and other parameters.

The sensor data are typically collected and managed in an historian database; popular commercial products include OSI PI and Honeywell PDF. These systems store process data, as changes (deltas) in readings over time. The data are managed via tags for each sensor, and stored as tag-time-value tuples. These historian systems are designed to store vast amounts of tag data over time, but aren't designed for data analysis, reporting and real-time intervention.

Typical challenges for engineering organizations in analyzing these sensor data include

- Accessing the historian data quickly; with meaningful analyses and reports that show system state and shutdown events; while enabling filtering and drill-down to events of interest across time windows and event frames of interest.
- Developing candidate sets of leading indicators (rules/models) for shutdown events on historian data; and back-testing these across different pumps, manufacturers and field environments.
- Applying the rules/models to real-time data in motion as they appear in the historian, and pushing alerts and notifications to companion systems for monitoring and intervention.
- Monitoring results and iterating rules/models for ongoing performance tuning.

Typical software systems for these tasks are unwieldy and require multiple IT and engineering skillsets; often requiring multiple people to operate. There has been a screaming need for simple-to-configure software systems to address the workflows required.

Note that the value generation from this workflow is significant. For example, across a collection of 1000 pumps producing ~200 BOE/day, shutdowns typically account for at least 200 hours of lost production or ~1,000 BOE/day; and pump failures result in 5-10 days of lost production. For $50 oil this results in ~$50,000/day or ~$20M/year in lost production per 1000 wells. Preventing a conservative 25% of shutdowns results in $5M/year/1000 wells to the bottom line.

In section 2 of the paper we describe the characteristics of data heterogeneity, integration, and normalization. Section 3 details the creation of predictive analytic models for failure. Section 4 lays out the real time event driven operational system, from stream processing architecture through live data mart. Section 5 covers performance of the operational system and section 6 lays out future work.

2. SENSOR NETWORK DATA

2.1 Data Characteristics and Transports

The sensors in the ESP network are in wells in very remote sites, often in other countries. The oil wells that are being managed may

be owned and/or operated by various organizations, so dictating common procedures and achieving a common level of data reliability is usually not possible. The data are often transmitted using older General Purpose Packet Radio Service (GPRS) with various levels of quality of service.

The equipment under test is located in many remote sites, with various degrees of exposure to weather and external effects on telemetry and network components, e.g. birds, so it's not uncommon for errors to occur in the data. This can include missing or incorrect historian entries. When an oil well is initialized, for example, the measurements are often unreliable and can take some minutes to stabilize.

2.2 Data Collection

There are several paths that the remote sensor data can take before they are analyzed. The most straightforward path is for the data to be collected from remote sites and transferred to a centralized location, either traditional relational database, or industry-specific historian database like OSI-PI or Honeywell PHD. This is often not as straightforward as it sounds, as the sensor samples are typically stored and forwarded between remote equipment and local and central historian instances; at varying latencies.

Agile Analytics environments like TIBCO Spotfire, and Event Processing systems like TIBCO StreamBase, connect very easily to any of these external data sources, as there is already a large preexisting library of external data adapters. If there is no existing adapter for a data source, it is straightforward to write a new adapter using the products' APIs. Regardless of whether existing adapters are used or a new adapter is written, a fundamental feature of the agile analytics and event processing systems is that the connectivity to external systems is isolated from the development of the business logic that performs the failure / outage detection.

While a well-developed software system using traditional programming languages will also isolate the code between different layers of the system, this is a fundamental feature of agile analytics and event processing systems which naturally isolate the infrastructure connectivity code, if any, from the algorithm processing steps. Since the connectivity code, the message passing, the control over concurrency is all handled by the analytics and event processing system, the subject matter experts do not need a team of developers. Rather, the responsible engineers can review the data in Spotfire's interactive visual environment to uncover patterns/rules/models in the sensor data leading up to the outage; backtest the models on historical data; and publish the models to Streambase for ongoing monitoring. This can all be achieved in an intuitive point-click environment without the need for traditional coding.

2.3 Data Integration and Normalization

The first step in the detection system is data cleansing, which waits for measurements to stabilize when a well is cycled, eliminates any duplicate samples, filters incorrect values, and imputes missing values from previous samples, as appropriate. When using previous values to substitute missing values, it is important not to obscure failures that cause missing sensor measurements. Sending both raw values that may be null and the latest values accomplishes this.

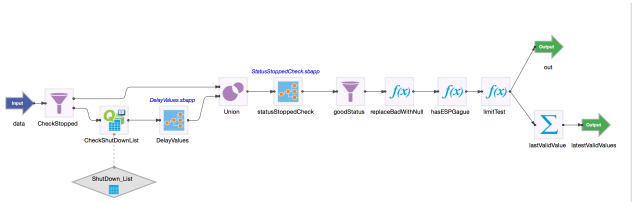

Figure 1. TIBCO StreamBase event flow diagram that replaces anomalous bad data values with the previous valid sensor reading (within time constraint) and stops invalid data from being processed when a well is being restarted.

3. PREDICTIVE ANALYTIC MODELING

While the exact features and algorithms are proprietary, and differ in each implementation, the basic techniques are common across different users, manufacturers, and environments. As such, the feature vectors, and the algorithms in this paper are not identical to the algorithms used in customer implementations.

3.1 Developing Historic Models

Different detection algorithms have been implemented: ad hoc rules, statistical control charts and machine learning. The detection algorithms all implement a common interface making it extremely easy to change between detection algorithms, or even run multiple detection algorithms simultaneously.

A detailed description of TIBCO StreamBase interfaces is given in section 4.1.

The detection algorithms that have been used in this system fall into two broad categories: ad hoc algorithms that subject matter experts have developed to based on their experience, and statistical models and control charts that are trained on collected data.

Training statistical models has been done using the TIBCO Spotfire analytics system either using built-in functionality or by calling R functions to train the model. To improve performance, the R functions were run in the TIBCO Enterprise Runtime for R environment (TERR) environment rather than the open-source R environment, but the functionality is identical. TERR is TIBCO's proprietary, high performance R engine that is embedded in both Spotfire and StreamBase.

Machine learning models are trained using TIBCO Spotfire calling R code running in TERR. Then the trained models may be executed from the TBCO StreamBase system by invoking the R model from an embedded TERR instance (or instances).

3.1.1 Ad Hoc Detection Algorithms

While statistical and machine learning techniques are extremely powerful; ad-hoc, rules-based algorithms are a good way to get started. Organizations that use the ESP failure detection system described herein will have been maintaining oil and gas wells for decades. This organization will have a culture, and have strategies they use to schedule maintenance. A new equipment failure detection system will therefore not be the only way maintenance decisions are made.

The people who schedule site visits for all of the sites need to have confidence in the new system to justify prioritizing suggestions over other maintenance tasks. If the system mimics decisions made in ways they understand, it will build confidence.

For example, if the system reports that a failure is likely due to high pump pressure readings combined with current spikes, it is more compelling than, a logistic regression model that says that the likelihood of failure in the next week is 0.72.

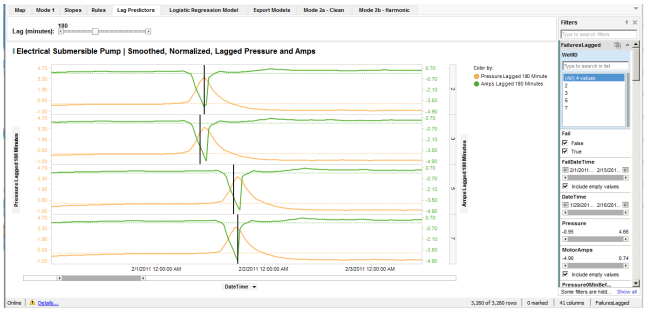

Figure 2. TIBCO SpotFire analysis of sensor data leading to failure. Subject matter experts use this sort of analytics to guide development of ad hoc detection algorithms.

Especially when considering that there will inevitably be issues when a complex new system is initially deployed, the approach we take is to initially deploy with algorithms based on the customer's current detection methods and enhance with statistical, clustering and machine learning models.

During implementation discussions, the customer defined their detection methods using decision flow diagrams like the one below. Their methodology is proprietary so this is not the actual decision diagram, but illustrates the sort of decision logic that is typically used.

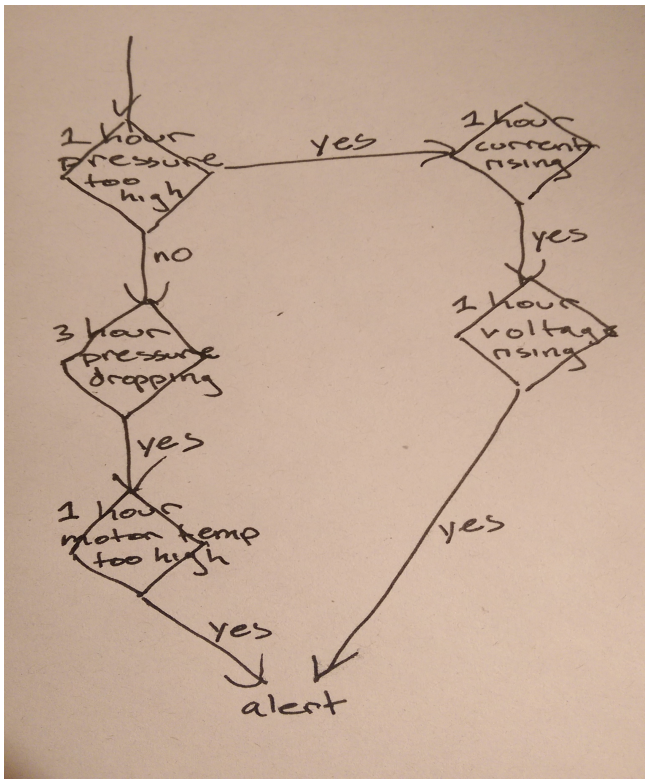

Figure 3. Failure detection flow chart drawn by subject matter export. As this logic is proprietary, some of the logic has been changed to obscure precise details, but the kind of logic the subject matter expert uses is shown.

The graphical event flow language that TIBCO StreamBase uses is particularly well suited to express this kind of algorithm. Here is the equivalent event flow code that automates the previous diagram. This diagram below is the complete source code that is compiled and executed. It's a very easy way for subject matter experts who think of their detection algorithm in terms of flow charts to implement the algorithm.

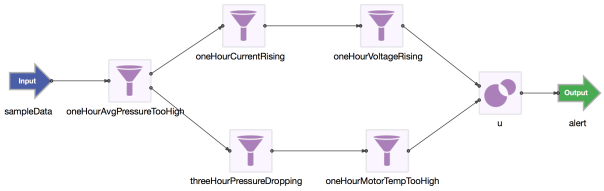

Figure 4. Equivalent TIBCO StreamBase event flow to previous hand drawn flow diagram. This event flow diagram has properties (not shown) for each operator that allows StreamBase to compile and execute this logic.

In addition to event flow diagrams, ad hoc rules have also been implemented. These were simple predicates without any forward or backward chaining. This was not because there were no production systems available—the TIBCO Fast Data platform supports two different rules engines, TIBCO Business Events and the open source Drools rules engine. Rather, the number of rules would not be able to take advantage of performance optimizations from RTTI [arg acronym escapes me while I'm disconnected] so expressing the rules as compiled TIBCO StreamBase predicates expressions is more efficient and simpler to manage.

Rule Name	Rule Predicate Expression
pressure	maxelement(pressure) - minelement(pressure) > 30
pchange	abs(maxelement(pressureROC) - minelement(pressureROC)) > 4
current	avg(current) / maxelement(current) < 0.78

Each of the variables, pressure, pressureROC, and current are lists where the first element is the aggregated value for the latest hour, the second element is the aggregated value for the latest 4 hours, the third element is the aggregated value for the latest 12 hours, and the fourth element is the aggregated value for the latest day. Pressure holds the average pressure values. PressureROC is the rate of change of pressure. Current is the average current value.

3.1.2 Clustering

When failure modes can be described, it is often effective to use clustering algorithms to number the failure modes and label samples with either that failure mode or as normal operation. Then a clustering algorithm can determine the cluster center points. Each sample measurement is compared to a failure cluster center, if the Cartesian distance to the closest cluster center is small, then it is flagged as a likely failure of that type.

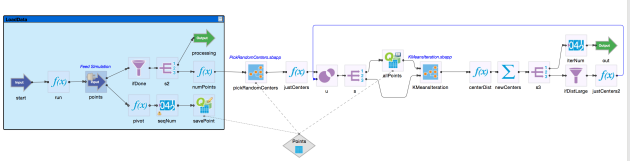

Figure 5. A TIBCO StreamBase event flow program for a k-means clustering algorithm.

The event flow diagram below scores a new list of sample measurements against the cluster centers that have been calculated by the k-means clustering algorithm.

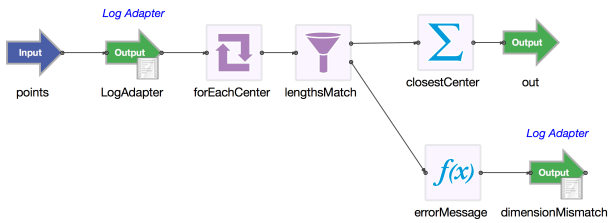

Figure 6. The TIBCO StreamBase event flow code for scoring a sample against a list of cluster centers.

3.1.3 Statistical Detection Algorithms

A binary predictor was built using a logistic regression model. TIBCO SpotFire was used to identify portions of historical data that represented ESP failures and train a logistic regression model. Then the model's coefficients are loaded into TIBCO StreamBase to evaluate the real-time ESP samples.

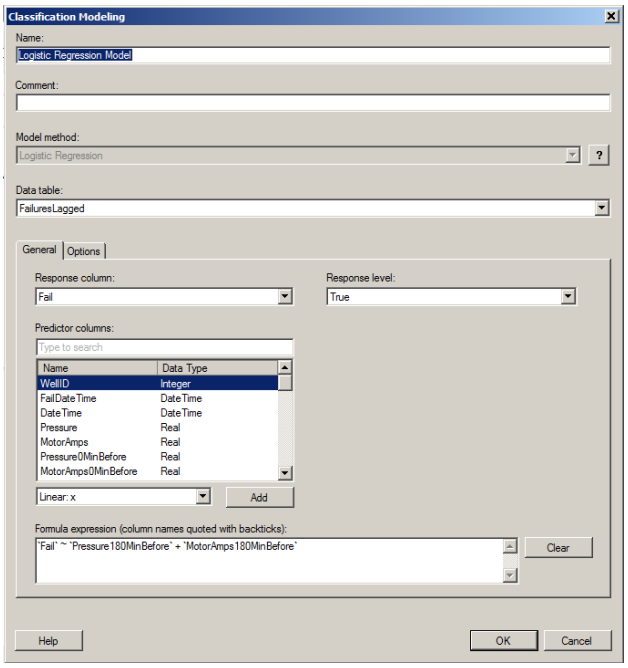

Figure 7. TIBCO SpotFire form to build a regression model

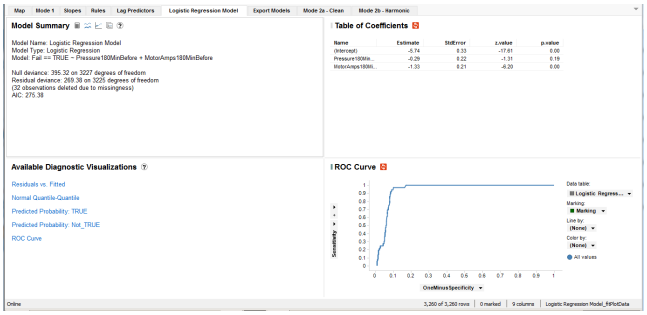

Figure 8. TIBCO SpotFire trained regression model coefficients published to TIBCO StreamBase.

4. REAL TIME OPERATIONS

It is a system requirement that implementing and running different detection algorithms must not be difficult. This is accomplished by separating the stages and defining interfaces so that components may easily be switched when the system is deployed.

4.1 Stream Processing Architecture

To allow different detection algorithms to be plugged in, an interface for an abstract detection algorithm has been defined and each specific detection algorithm implements that interface. One requirement is that all of these detection algorithms operate have the same inputs.

A StreamBase interface is analogous to a Java interface. It is a source code module that only contains definitions. While a Java interface defines method signatures with method returns, a StreamBase interface defines input streams, output streams and shared query tables, specifying the schemas of each. Every StreamBase application (a .sbapp file) that implements the interface must define the exact same input streams, output streams, shared query tables all with the exact same schemas.

In Java methods defined by the interface can be invoked on an Object that implements that interface. In StreamBase event flow programming, an extension point is used to enqueue and dequeue to streams without knowing the actual implementing module. Unlike invoking a method on a Java interface, a StreamBase extension point may specify (either at design time or at runtime) multiple implementing modules so enqueueing a tuple into an input port on that extension point can send the tuple to several implementing modules.

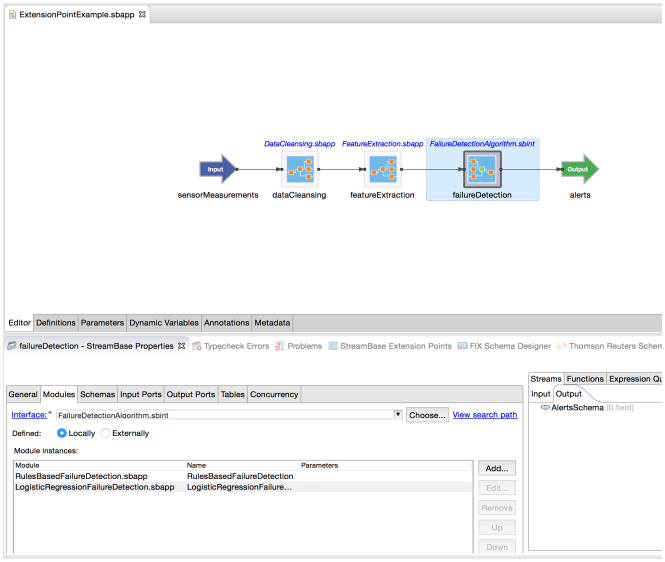

Figure 9. This event flow uses an extension point for the interface FailureDetectionAlgorithm.sbint. The properties view shows that there are two implementations: RulesBasedFailureDetection.sbint and LogisticRegressionFailureDetection.sbint. When a tuple is sent to the extension point, it will be processed by both implementing modules and any output tuples will be sequentially emitted from the output port of the extension point.

Interfaces are used in this system to specify algorithm modules so that different algorithms can be deployed, or several algorithms run simultaneously, without any modifications to the rest of the system. Developers are also required to match the input and output schemas exactly so that any errors are caught at design time rather than causing runtime errors.

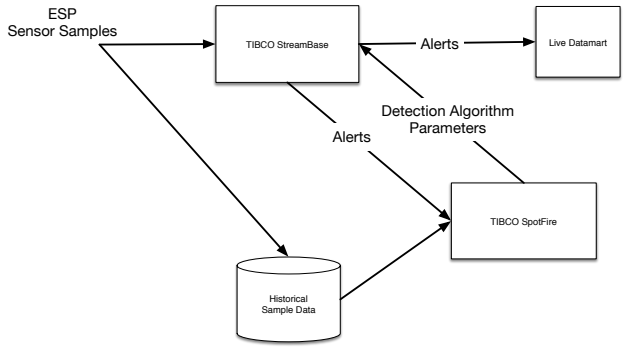

Figure 10. System Architecture Diagram. ESP sensor samples are sent to both a historical database and into TIBCO StreamBase. TIBCO Spotfire uses data from the historical DB and send detection algorithm parameters to StreamBase. StreamBase processes sensor samples and sends alerts to Spotfire (vis Spotfire Automation Services) and into the TIBCO Live Datamart to be displayed on user's dashboards. Users may view the alerts in context using Spotfire and further refine algorithm parameters.

4.2 Stream Processing Implementation

This system was developed entirely with event processing products with no traditional programming of any of the algorithms, analytics, dashboards, and interactive interface to the aggregated feature vectors and alerts.

The only traditional programming language work used to implement the initial system is adapters to external systems, like OSI PI and OSI AF and unit testing. Most of the java code for the junit-based testing framework was autogenerated.

The implementation of this system in TIBCO StreamBase event flow follows the same pattern as most systems. A main event flow module connects to the infrastructure, in the example below, it connects to OSI PI. It passes the data through the stages of the system: data cleansing, feature extraction, and failure detection.

Features and alerts are sent to the Live Datamart where users can interactively query or drill down into the data, display the results on a dashboard textually or graphically, and give simple ad-hoc alerts in the Live Datamart layer.

Alerts are sent to TIBCO SpotFire through SpotFire automation services. This will send an alert email message to a user who will see the frame of data that comprises the alert. The user can decide to schedule maintenance on the well or select the alert which will invoke TIBCO SpotFire on the data frame. Further analytics can be run on the data to determine if it is, in fact, an actual failure or to refine the algorithm parameters if it is a false alarm. If the algorithm parameters are refined, the new parameters and thresholds can be sent to the running TIBCO StreamBase server which will update the parameters in the running server instance.

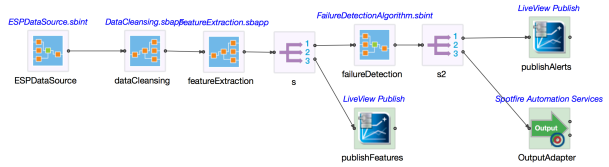

Figure 11. Top level event flow. Connects to external input and output systems and calls each stage of the system.

After data cleansing, a feature extraction stage aggregates the data and calculates common features that are used by all of the detection algorithms. The cleansed data is aggregated using several time windows. The specific features that are calculated are proprietary to each customer, but common statistics like mean values, standard deviation, movement trend are calculated for different time windows, typically daily, hourly, and 10-minute windows.

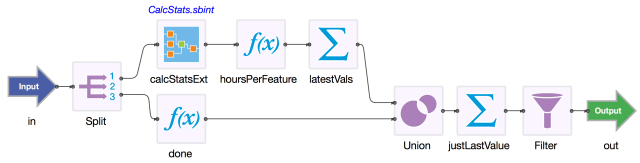

Figure 12. The feature extraction event flow. The CalcStatsExt extension point calculates the statistics for every time window. The latestVals agregate operator outputs the latest value for each windowed statistics as the list of features per time window.

Figure 13. This is the same implementation event flow that is used to calculate every time-windowed feature.

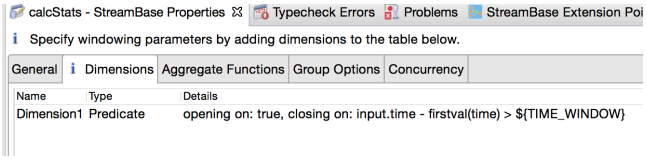

Figure 14. This is the definition of the calcStats aggregate operator's dimension. It closes the window if the delta time in that window is greater than the parameter TIME_WINDOW. The default time window is hours(1).

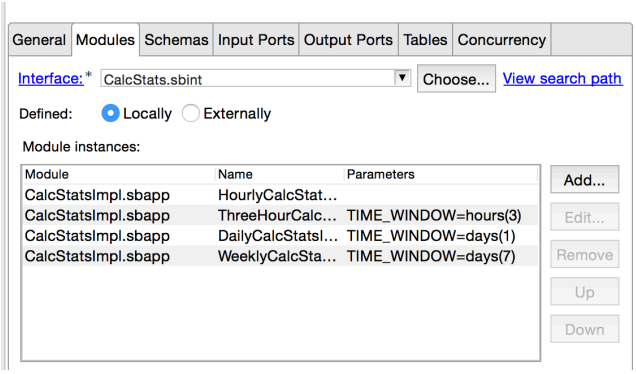

Figure 15. This is the properties of the extension point that declares what implementing modules are used. The same CalcStatsImpl.sbapp module is used each time, but the TIME_WINDOW parameter is overridden to be the desired time window.

5. PERFORMANCE

The performance of different detection algorithms varies widely. The table below shows benchmark timing measurements of the various failure detection algorithms discussed in this paper.

Detection Algorithm Type	Average Performance (messages/second)
Ad Hoc Pure Event Flow	2,311,213
Ad Hoc Rules	569,835
Logistic Regression	45229
K-Means	16060

The ad hoc pure event flow algorithm is extremely fast. The logic is very simple, and very easily optimized. The ad hoc rules are also quite fast because each rule is compiled the first time it is seen. The logistic regression and k-means algorithms do much more complex arithmetic for each tuple which the correspondingly worse performance indicates.

Fortunately, the typical ESP sensor data rate for an entire sensor network is typically in the range 1000 – 10,000 messages/second, depending upon the number of wells in the network. Even the more advanced algorithms preforms well enough.

Additionally, the detection logic is very easily partitioned. All of the algorithms explored in this project operate on the information for only one sensor. The samples for each ESP must be processed sequentially, but different ESP's samples may be processed in parallel.

6. FUTURE WORK

Detecting ESP failure using event processing software has been found to be quite effective for rapidly developing the detection system, allowing subject matter experts to easily deploy and tune detection algorithms, but this is only a qualitative assesment. Measuring the success of this kind of system is not as simple as calculating which alerts were failures and which were false alarms. The system is designed to detect failures before they happen, so when maintenance is performed on the suspect ESP, it will not fail regardless of whether the detection event was correct or not. The only way to quantitatively measure the degree of success is to evaluate the performance is to compare the downtime before and after the system has been in place. However, the system has not been in use for sufficient time to accurately track Return On Investment (ROI) measurements. Over the following months, the system downtime must be tracked.

While the current set of ad hoc and statistical models are felt to be effective, when the system is extended to new types of operation, such as drill head monitoring or other preventive maintenance problem domains, more detection algorithms will be needed.

Training a new detection algorithm can usually be done without significant effort by running the training data through an existing training algorithm. Rich libraries of these exist in R and other publicly available libraries. While the training of these algorithms can be quite time consuming, the scoring of a trained model in our event processing system is almost always extremely efficient. However, even though adding a new detection algorithm is quite straightforward, each new scoring algorithm must be created manually.

The currently set of scoring algorithms with native event processing implementations is small. Many event processing systems, like TIBCO StreamBase can call libraries written in Java, C++, C#, Python, and R, the execution of the scoring is not typically as efficient as a native algorithm.

7. CONCLUSIONS

Using Agile Analytics and Event Processing systems to analyze real-time sensor measurements coming from equipment such as ESPs has proven to be very effective. Two Analytics / Event Procesing developers developed the core of the system in less than a month. The detection algorithms were developed by subject matter experts who occasionally worked with the Analytics and Event Processing experts during development. After a week of training, the subject matter experts were able to take over the development of the system. Subject matter experts who are not software developers and do not have the ability to maintain a comparable system written with traditional software are now able to modify and replace the detection algorithms and deploy them in the real-time system.

8. REFERENCES

[] Hari Balakrishnan, Magdalena Balazinska, Don Carney, Uğur Çetintemel, Mitch Cherniack, Christian Convey, Eddie Galvez, Jon Salz, Michael Stonebraker, Nesime Tatbul, Richard Tibbetts, and Stan Zdonik. 2004. Retrospective on Aurora. The VLDB Journal 13, 4 (December 2004), 370-383. DOI=10.1007/s00778-004-0133-5 http://dx.doi.org/10.1007/s00778-004-0133-5

[] Tibbetts, R., Yang, S., MacNeill, R., Rydzewski, D. 2012. StreamBase LiveView: Push Based Business Intelligence. New England Database Summit (2012).

- Kmeans algo paper
- Logistic regression paper
- Witsml reference
- [perhaps cite GPRS ieee article: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5485637]