Reducing Business Surprises through Proactive, Real-Time Sensing and Alert Management

Mitchell A. Cohen, Jakka Sairamesh, Mao Chen
IBM T. J. Watson Research Center
Hawthorne, NY, 10532
{macohen, jramesh, maochen}@us.ibm.com

Abstract

OEMs need to transform the way they do business in order to ensure better quality of products and services. Crucial failure symptom information is lost between the end consumers of products and the manufacturers. Manufacturers have access to this information but are typically unable to handle its volume in a timely fashion. However, using this data properly can result in diminished labor time in issue resolution, decreased warranty costs for manufacturers and improved customer retention. In this paper, we present a novel system for Proactive Real-Time Event and Alert Processing, which enhances an enterprise's ability to monitor, analyze and detect critical business events and situations. This capability can help improve operational efficiencies, reduce costs, streamline processing of business alerts, and enable the enterprise to react in a more timely fashion. The system can enable monitoring of near real-time, low-level, industrial sensor and controller events, and high-level events from underlying structured and semi-structured data. The alert system uses domain knowledge to enable processing the events in real-time, performing the appropriate analytics and evaluation on the events and alerting the right set of users. The alert and event processing system has been deployed and validated in real pilots with industry specific data. We are currently validating the scalability and performance of the alert system with many different information sources and high volume sensor information.

1. Introduction

Over the last ten years we have been witnessing a transformation in the structure and daily operations of large and small enterprises because of the drive to lower costs, leverage the Internet for business operations, and integrate better with their supply chains. Increasingly, enterprises are dependent on their supply and value chains to build products and deliver services respectively. This has a profound impact as the streamlined flow of information and services in the supply and demand chains is crucial to manufacturers and enterprises wishing to capture the product feedback needed to improve product quality and increase customer satisfaction. For example, in the automotive industry, information disconnection in the demand chain is causing slower product feedback, rising warranty costs and declining customer satisfaction in spite of the advances in vehicle technology. Sensing and alert management in such demand chains can provide accurate and faster feedback, improve quality and reduce costs. In the following, we present the motivation for scalable sensing and alert management.

Rising warranty costs: It is estimated that in the US alone, the warranty costs in the automotive industry are around 3% of the OEMs (manufacturers) revenue, and the total costs for 2003 are estimated at 12 Billion USD[i], and rising. The same issues are being grappled with in the Electronics, Aerospace and Heavy Engineering industries. Manufacturers are extremely concerned about the rising warranty costs, and are determined to reduce them as much as possible through sensing product information (e.g., vehicles) and their status in real time.

Lack of visibility into operations: Most enterprises do not have deep visibility (at a fine grained level), controllable operational models and flexible and easily modifiable business operations. With sensors and actuators the right capabilities for monitoring operational performance, integration and control can be enabled for businesses to sense and adapt their daily operations in a timely manner. In the world of industrial sensors, the model of top-down design becomes critical as the right sensing of the operations can provide crucial information for optimizing business operations. The business metrics for enterprise operations can include improving the current cycle times of the core business processes in manufacturing, production, parts, inventory and sales. Shrinking the cycle times necessitates deeper and better visibility in

the enterprise operations. In this paper, we discuss various business scenarios, challenges and technology components that can enable such monitoring and evaluation in a scalable fashion.

Several papers and articles in the literature ([1], [2], [3], [4], and [5]) present novel event stream processing work. However, they have primarily focused on ad hoc communication domains or domain agnostic systems, and very few have focused on using semantics for event processing. The fundamental difference in our work is combining domain knowledge structures with sensor information to extract meaningful information in order to generate real alerts and take actions through complex decision processes or activate additional monitoring processes.

1.1. Business Metrics and Challenges

In order to enable monitoring of business operations, specific business metrics need to be defined for capturing the state of the business operations, current performance and trends. Tracking these metrics enables gathering of intelligence and predicting future trends in order to make timely decisions. For example, metrics are defined for managing the life-cycle of products from the production phase to the delivery and service phases. The following are typical examples of business metrics: a) Production and product life-cycle process times; b) Down time of assets in production and assembly lines; and c) Cost of monitoring production processes through automation. The challenging problems include identifying the metrics (assets or processes), sensing the data for the metrics in a scalable fashion, controlling the sampling rate dynamically, and evaluating the data to extract critical alert information.

1.2. Contributions

Summary of the Alert-Event System: In this paper, we describe a unified semantic event stream system that continuously monitors diverse data sources and generates alerts based on domain specific rules. This system can enable manufacturers to closely monitor critical business events (reducing surprises) and gather business intelligence from information such as product failures, warranty intelligence, field events, sales transactions, asset performance and others. Our solution is currently undergoing field trials and pilot deployments. This paper discusses the system design, methodology, reliability and scalability. A mathematical foundation is being worked on, but is beyond the scope of this paper.

2. Business Scenarios

In this section, we present two business processes and then describe the business and technical challenges.

2.1. Production, Engineering and Service

Production processes and asset-management: Enterprises are investing in sensor systems to enable the monitoring of the production processes, production assets and product lifecycles in the factory environment. The deployed sensors in the production environment capture daily production data and quality information in near real-time for the production engineers and specialists to keep a close watch on the manufacturing processes and assets involved in the process (e.g., robots on an assembly line).

Industrial Sector Demand Chain: In the Automotive Sector, manufacturers must manage rich relationships with dealers, fleets, and independent repair shops to support vehicle service lifecycles and build aftermarket revenue streams. Optimizing these aftermarket relationships requires OEMs to effectively support a wide range of critical business activities including real-time sensing of inventory replenishment, logistics, parts tracking, product behavior, retail performance management and others.

3. Technical Challenges

Real-time integration of diverse sensors and data types: Integration of a multitude of events and data from various sensors in various formats is a complex challenge. Sensors of various kinds are being deployed at every operational manufacturing site. These sensors are built by third-party vendors who have custom ways of sensing, sampling and generating the events. The data rates could range from 1-50 megabytes per second. The monitoring, integration, analysis and distribution of event information based on the data are critical for enterprises to streamline their manufacturing processes.

Programmability of controllers and actuators: Current pervasive systems are focused towards light-weight middleware layers over mobile devices with programming frameworks such as *MIDPs, SMF, OSGi, J2ME* and others. For a service oriented model, where sensor controllers and actuators offer services to higher level applications access (e.g., sampling rate), new models of programmability with messaging controller interfaces (e.g., Web Services) are needed.

Service-Oriented abstraction: With heterogeneity of various applications, information sources, sensors, and

sensor controllers, there is a need to abstract away the underlying sensor components and interfaces into a collection of services for higher level applications to use and configure. Each device controller can be defined by a service interface to enable regional or local computing servers to access and integrate with the controller.

4. Event Stream Processor Overview

At the center of the system solution is an Event Stream Processor which handles all events, ultimately deciding on the actions that need to be taken. Every kind of message coming from external systems, sensors and devices are treated as events. When running, the Event Stream Processor steps through following for each event: a) Event message receipt; b) Event transformation; c) Metric calculation; d) Metric evaluation; and e) Action invocation. The first 2 steps are performed at a message adapter layer.

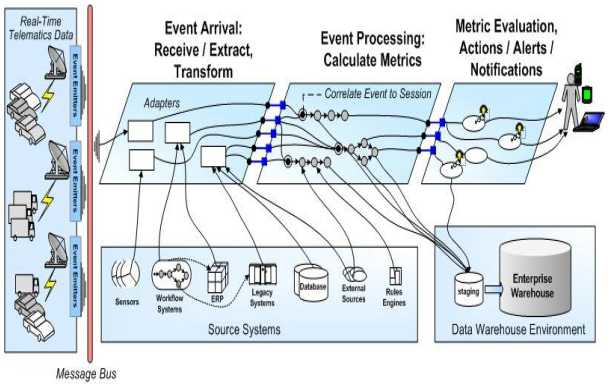

Figure 1. Flow of Event Stream Processing

4.1. Event Message Receipt

The Event Stream Processor has adapters for receiving messages. Adapters determine the transport mechanism for the group of senders (HTTP, JMS, etc). We have created adapters which have web services that can be called from clients. We also have adapters which can pull events by polling a potential event source. Here web services can be used to.

4.2. Event Message Transformation

Each adapter is capable of handling different formats of event messages. During transformation the messages are converted into a common format. Many different transformations are supported. The preferred method of transformation is via a web service call. The actual transformation can be done via any of the commonly accepted transformation techniques including XSLT, translator packages, and straight Java code.

4.3. Metric Calculation

Metrics are calculated measures based on incoming events. Metric calculations can go beyond the scope of events, potentially accessing other systems for additional data input or for additional calculations. Rules engine integration falls into the calculation category. Computations can range from mathematical calculations to text analytics including syntax and semantic analysis. For the automotive industry, we have created metrics for: a) Fleet management or individual vehicle monitoring, e.g., "running average of vehicle speed"; b) Complaint management, e.g., "warranty claim symptom"; and c) Class of vehicle performance, e.g., "percentage of vehicles within a class (Make, Model, Year, Trim combination) with warranty claims on a particular subcomponent."[ii]

Metrics can be processed on both structured and unstructured information. Telematics systems in automobiles collect many (anywhere from thirty to over 100) different parameters from vehicles. The telematics systems take snapshots of how the vehicle is being used and is performing. Values such as engine speed, vehicle speed, and tire pressure are collected multiple times per second. These values are well-structured real numbers. Metrics based on telematics data can be as simple as the value itself, such as "tire pressure" or can be calculations based on current and previous values, such as "running average of the vehicle speed to engine speed ratio." Calculations can include multiple different types of values, common with ratios, or even values from external sources.

Unstructured information may come in on events which contain text fields. For instance, automobile warranty claims come from dealers containing service technician write-ups. The text fields contain brief descriptions of the symptom and cause of a problem as well as the action taken. Each of symptom, cause and action can be metrics that are "calculated." In this case the calculation consists of syntactic and semantic analysis of the text using glossaries of categorized terms, abbreviations, and common misspellings.

4.4. Metric Evaluation

Prior to the Event Stream Processor being run, an expert (or a group of experts) with a great knowledge of the data, expected event arrivals, and normal overall system behavior creates rules based on the metrics. These rules are used in both the Metric Calculation and

Metric Evaluation phases of the processing. An example of such a rule on (structured) telemetric data in written form is "the twenty minute running average of vehicle speed should not exceed 75 miles per hour." Rules are actually stored in XML form. An example of a rule on the (unstructured) text of warranty claims using metrics described previously could be "there should never be a warranty claim where the symptom was fire" which will enable a quality analyst to be immediately alerted if such an unfortunate incident should occur.

Figure 2 shows each of the steps taken by the Event Stream Processor. Each step has a well-defined interface that is implemented by a local web service. Using such a Service Oriented Architecture allows easy modification of the processor's behavior through simple plug-and-play.

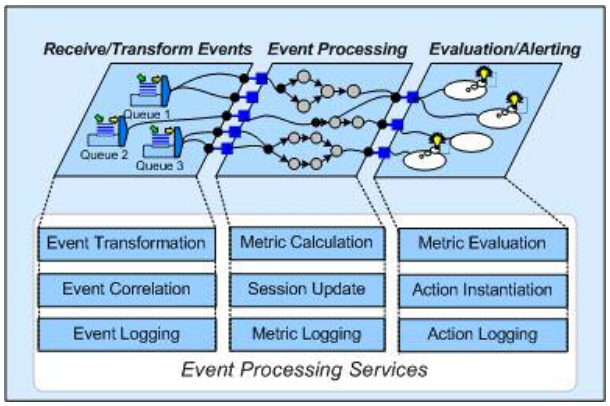

Figure 2. Services Oriented Architecture of the Event Stream Processor

Upon arrival, events are passed to the *Event Transformation Service* to be converted into the standard format used by the rest of the event processor. Based on the incoming event, the *Event Correlation Service* retrieves a list of metrics which need to get calculated. For a metric involving past values, the service also retrieves the relevant session. The values contained in the event are then used in calls to the *Metric Calculation Service*. The Event Stream Processor invokes the *Session Update Service* to update the session to ensure the current event is included in future metric calculations. The *Metric Evaluation Service* determines what actions if any need to be taken based on the newly calculated metric. These actions are taken with calls to the *Actions Instantiation Service*.

5. Scalability

Scalability is a key design point for the Event Stream Processor. The ability to handle large numbers of events has become paramount in the new world of sensors and actuators. For scalability, we need to look at the Event Stream Processor as well as the systems it uses. For the processor itself, there are two issues preventing simple replication of the server to allow any number of instances of it working in parallel, i.e., direct scalability. The preventative issues are session management and event listening.

The Event Stream Processor continuously calculates metrics, which get updated as events arrive. Calculations of metrics often depend on having sessions. That is, we need to store information about past events to calculate metrics based on new incoming events. A common example is with the calculation of running averages. Consider the metric of a running average of engine speed, a common value supplied by telemetry, for a particular vehicle. If the running average is over a ten minute period, then all data coming in over the last 10 minutes must be stored. All the data for a particular vehicle is needed together in the session to be able to do the calculation. As each new triplet of vehicle, timestamp, and engine speed arrives for a particular vehicle:

1. The session for the engine speeds for that vehicle is retrieved.
2. Data no longer needed (older than ten minutes in this case) is removed from the session.
3. The incoming timestamp and value are stored. (Steps 2 and 3 can be implemented with a circular array of session data.)
4. The running average metric is calculated.
5. Rules are applied to the calculated metric to determine if a warning needs to be indicated.

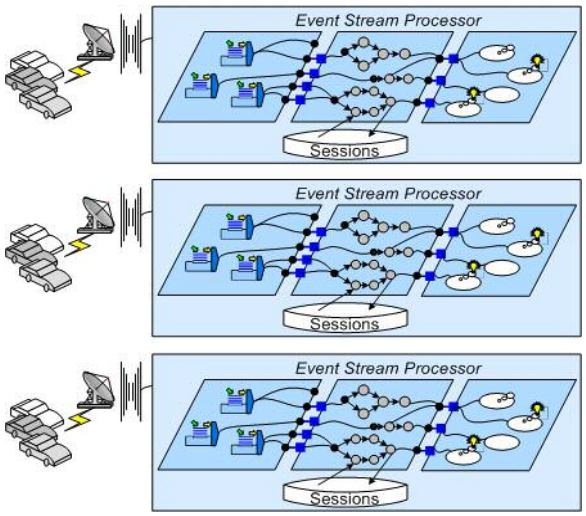

Figure 3. Assigning Event Creators with Their Own Processors with Local Sessions

Scalability can only be enabled by ensuring the current version of the session is available at the instance processing the event. One option for solving this problem is to have a common store for the sessions. As shown in Figure 4, the sessions can be stored in a Database Management System or a data grid. Each instance of the Event Stream Processor would retrieve and update the sessions as needed. Both database managers and data grids are often wrapped in web services. Another option for ensuring session availability, shown in Figure 3, is to assign each session to a particular Event Stream Processor instance. Sessions can then be stored and managed locally by each processor.

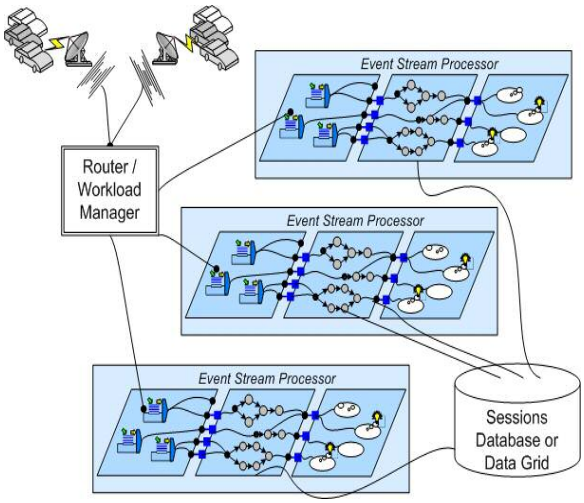

Figure 4. Using a Router to Pass Events to Processors with Shared Sessions

Message receipt remains separate from event processing allowing multiple nodes to handle the processing. In one instantiation of our system, we route incoming events based on the session to which they belong. Each event processing node handles the events for a group of sessions. There are two main factors in deciding how to partition the sessions across the event processing nodes:

1. Can we separate the sessions in a way so that all the metric calculation can be done locally? That is, can the data existing within the Event Stream Processor be collocated within nodes so that metric calculation can be done locally?
2. Can we separate the sessions so they are split up so that messages of the same type go to different event processing nodes? Messages are typically skewed by type. For instance, if metric calculations are mostly at the vehicle level, "change in transmission gear" events are much more common than "engine on" messages. Routing messages by vehicle will avoid workload problems caused by the difference in the volume of messages by message type.[iii]

6. Routing

There are two methods for handling the message routing:

1. Using a conventional message router
2. Assigning target web services for each of the different sessions

Message routers are well understood, but we'll explain further on the use of web services. Each different session can have its own web service. Each event creator can then be assigned a specific web service to call based on the session needed for its events. This type of mapping of event to sessions only works when sessions are only needed at the creator granularity. The metrics being calculated dictate the granularity. As long as a group of sessions rely only a specific set of event creators, those creators can use the same web service.

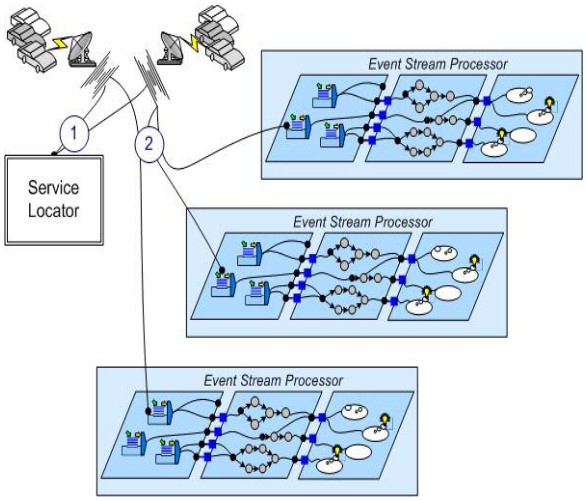

Figure 5. Using a Service Locator to Allow Event Creators to Send to a Processor

A more elaborate scheme to allow creators to send different types of events to different nodes consists of two steps. Prior to sending the event, a service is called to determine where to send the event. This routing decision can be based on the message itself. This mechanism pushes some of the routing processing onto the event creator. This approach has been described extensively in The Integrated Building Design Environment by Fenves et al [6]. A key ability of the Event Stream Processor is how at each step in its event

processing, it can integrate with external systems. In one particular instance we integrate with a semantic text analytics engine. Every time a warranty claim event is processed, it is passed along to the semantic text analytics engine to pull out symptoms, causes, and actions from text entered by the service technician at the automotive dealership. The integration can be done synchronously or asynchronously as shown in the figures below.

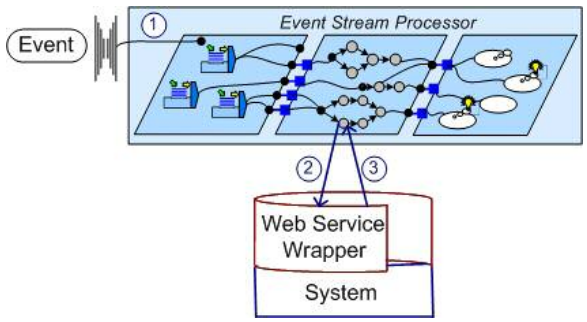

Figure 6. Synchronous Integration of External System

With synchronous integration, as shown in Figure 6, during event processing the call to the external system is made and the processor thread making the call awaits a response. Choosing synchronous integration makes most sense when integrating with external systems that respond quickly relative the needed response time.

The asynchronous integration shown in Figure 7 allows processing to continue prior to getting the result of the call. When the processing of the call is completed, the system creates a new event which allows processing to continue.

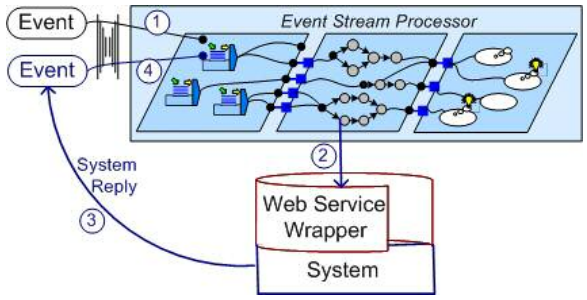

Figure 7. Asynchronous Integration with an External System

7. Conclusion

In this paper, we described a novel unified semantic event stream processing system for general critical alerts that enhances manufacturer process efficiency in quality, service, and warranty management. This system improves visibility into critical alerts and reduces surprises on warranty costs. It allows business early access to information and trends. Our alert and event system is undergoing field trials. A version of the solution has already been deployed and functional on real industrial data. The system is written in Java with XML and Web Services as the support for semantic data structures, rules, configuration and evaluation. The performance has been excellent on real-time information streams. Flexibility allows multiple deployment scenarios enabling the system to be optimized for various event evaluation mechanisms. Various techniques are deployed to allow for scalability with the metric types and data patterns determining which technique is to be used. Further validation on the scalability and routing is being done in real pilots and engagements.

References

[1] Daniel J Abadi et. al., *The Design of the Borealis Stream Processing Engine*, Second Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, January 2005.

[2] C. McGregor and Josef Schiefer, *A Web-Service based framework for analyzing and measuring business performance*, Information Systems and E-Business Management, Volume 2, Issue 1, Apr 2004, pp. 89-110.

[3] Tivoli Risk Manager, Event Correlation Engine, 2003. http://www.ibm.com/tivoli.

[4] David Luckham and Mark Palmer, *Separating the Wheat from the Chaff,* RFID Journal, 2004.

[5] Rajit Manohar and K. Mani Chandy, *Dataflow Networks for Event Stream Processing*, 16th IASTED International Conference on Parallel and Distributed Computing and Systems, November 2004.

[6] S. Fenves, U. Flemming, C. Hendrickson, M. Maher, R. Quadrel, M. Terk, and R. Woodbury. *Concurrent Computer-Integrated Building Design.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 07632, 1998.

[i] Estimate is from AMR Research – on the web at http://www.advmfg.com/content/resourcecenter.asp?id=440.

[ii] Automotive *components* include high level systems such as Engine, Transmission, and Body. *Subcomponents* are subsystems within the components. For instance, a subcomponent within the Engine component is the High Pressure Oil System.

[iii] Of course, there can be skew in the messages created for different vehicles. A whole paper (or perhaps even a text book) can be written on dealing with message skew.