

**Abstract** – The goal of this whitepaper is to demonstrate how an enterprise could leverage the concept of “situational awareness” as an alternate mechanism to cope with changing business conditions. Business event correlation when combined with user defined business rules leads to the definition of execution pathways that invoke business logic in a manner that was not previously envisioned. This enables the enterprise to *deal* with new business events that would normally have resulted in business exceptions.

The pattern described in this paper promotes the concept of “management by exception” by introducing a service provider component that insulates the business event consumer from disruptive business event changes. The proposed architectural construct attempts to bring about a level of stability to the business event consumers. In addition, this paper helps point out that embedding the architectural principle of separation of the volatile sequencing of calls in a business process from the more stable business behavior (encapsulated within the business services) that form the steps of a business process provides the foundation for incorporating “situational awareness”.

## **Containers: Architecture and Design, Analytics and Business Process (April 2010)**

### ***Introduction –***

#### ***Achieving Situational Awareness while preserving durability of business services***

Situational awareness is the ability to alter or replace business behavior dynamically in response to the "situations" such as business events and/or changing business conditions or operating conditions.

A previous article elaborates on how [Situational Awareness](#) can help achieve reusability within the context of SOA. In that article, a context-aware business rules framework was introduced. The business rules framework was a critical foundation component that was needed to incorporate alternative business context into an existing business service (for instance, localization context or regulatory policy). Taking the concept of business rules framework one step further we look at Event Driven Architecture (EDA) and Complex Event Processing (CEP) and how they enable Situational Awareness.

EDA leverages business events as the mechanisms for invoking the right business behavior. This article shows how separating business service sequencing from that of business behavior (that is encapsulated in the business service) provides the architectural foundation for achieving situational awareness. Here we show that business events can be analyzed to find the appropriate service for handling the event. Not tying business events to a predetermined business process sequence or pre-configured business service provides a level of flexibility that is key to making a business solution “situationally aware”.

Further, we show that utilization of CEP provides the additional functionality of identifying event patterns from event streams (temporal and/or geo-spatial) thus helping the enterprise to find the appropriate business service to respond to the event. For known event patterns deductive reasoning is used while inductive reasoning is employed to discover newer event patterns.

This goal of this article is to prescribe a solution from a practitioners’ perspective to demonstrate the benefits of adopting [Event Driven Architecture](#) (EDA) as a foundational architecture paradigm along the incorporation of [Complex Event Processing](#) (CEP) and [Event Steam Processing](#) (ESP).

### ***What is EDA?***

EDA deals with linkages between the systems, processes, value chains of the enterprise through the event publication and event subscription model so as to loosely weave together connectivity across disparate systems.

These links can be strung together to keep the enterprise information flow fluid, flexible and extensible thus positioning the enterprise to accommodate changing business needs and changing market conditions.

## ***What is CEP?***

CEP deals with contextualizing business events with the correlations of temporal and geographic events. At the heart of CEP is [ontology](#) and [semantic reasoning engine](#). Ontology makes CEP business context-aware while reasoning engine assists in the decision making process. Based on the outcomes of decision, CEP can also learn from past decisions that are enacted on these events that are then transformed into rules for usage in the next round. This process of self-learning and self-healing makes the CEP engine powerful especially so, as decisions by the human actor are used as the basis for making the business logic more proactive without the need for adding code. This self-learning process also aids in automating the logic embedded in the human work-flow and thus makes the overall process more robust, scalable and predictable.

## ***Problem Domain Definition – Situationally Aware Business Services operate based on business event streams with contextual information***

When a business service is equipped with event streaming processes, its operations rely on the observation of business event patterns, event hierarchies and event correlations to subsequently trigger event rule evaluations. Here the pattern being sought is the binding of a verb/ object pair with that of a subject. This pattern matching allows one to establish causal relationships or a temporal relationship (time-based) which could be further aggregated with other events as well. These aggregations and correlations of events are performed with the explicit purpose of applying a behavioral rule.

Incoming events are first augmented by combining temporal and geographic events by applying runtime pattern matching rules. This allows alternate alerts to be triggered or new events to be published. One essential implementation technique using event stream processing is that of [Complex Event Processing](#) (CEP). The focus of this paper is to establish mechanisms to achieve situationally-aware behavior by leveraging CEP based rule engines to discover novel business execution pathways.

The power of CEP is its scalability in adapting to enterprise wide business services, and in correlating business events through [semantic relationship](#). The event correlation technique provides a crucial input to deductive and inductive reasoning logic, which is embedded in the CEP engine. CEP enables the business community to react to new business trends or new operational conditions by identifying new rules or by wiring in alternate operational processes to handle these changing market conditions in real time.

## ***Brief note on CEP based Reasoning Engines (Deductive Reasoning vs. Inductive Reasoning)***

Reasoning engines are a critical component of CEP and they rely on the concept of deductive and inductive reasoning. Deductive reasoning is based on a set of theories and a proven hypothesis, allowing these patterns to be applied to observed events. The theories must be true in order to make deductive reasoning valid.

On the other hand, people rely on inductive reasoning to derive hypothesis and axioms when they find repeating patterns when observing events. The exploration of these repeating patterns in time leads to the discovery of a hypothesis and finally evolves into a theory strengthened by an increasing set of observations. Event streaming patterns and correlations are not completely established and are non-deterministic in inductive reasoning as they

are based on a study of the frequency with which the streams of related events occur in succession. Greater the probability of occurrence of event correlation patterns (as measured by sample data sets) the greater the degree predictability in identifying the successive event.

## ***The Business Scenario***

Given all the background of EDA and CEP in previous sections, we will use a business scenario to depict how to apply EDA and CEP to business services. This section highlights the three elements that we will elaborate in the business scenarios:

1. How external events drive business processes
2. How business processes generate business events
3. Capturing how the human actor uses a manual work flow process to handle a business exception

Assume a stock trade customer has executed multiple trades (events) in the past 8 hours, all of which are trading requests for the same stock. These trade requests could all be collated resulting in an alert being triggered for any stock trade over a certain volume based on a pre-established trade volume threshold defined by the company.

With the above scenario in mind, we outline what a human actor does when a mass sell or buy request is made or when a multi-part buy or sell request is made that is above the normal stock trade volume range. Here the CEP engine establishes the fact that this is an out of range trade request by a customer and initiates a manual workflow to a human actor i.e. the broker. The scenario plays out like this:

- a) Broker checks if the customer is a company official (they can go online and check the prospectus of the company which can be also obtained via RSS or ATOM feeds).
- b) Broker checks if there was a major announcement with significant financial impact on the company that would affect the stock price in the past 8 hour period (they go online and check the business news such as Reuters or Bloomberg both of which are available via RSS or ATOM feeds).
- c) Broker notifies the clearing house that there should be a stop order on the processing of the customer request due to suspicious behavior (Broker sends an SMTP email request).
- d) Broker initiates “seek approval” notifications to clear this request from Brokerage and the financial organization whose Stock is in question (Broker send an SMTP email request).
- e) Broker waits for the “approval” notification prior to resubmitting this withdrawn customer buy/ sell request (Broker awaits an SMTP email notification).

If a CEP engine is coupled with deductive reasoning engine then the attributes that are captured for the rules engine to process the set of events are as follows (attributes are in bold):

- a) **Stock symbol** and **buy** or **sell** attribute for **N shares** (that is greater than X number pattern) and RSS feed on company official name equals customer name or title
- b) RSS feed on **Stock symbol** that puts a **business or financial alert** on the **Stock Symbol**
- c) SMTP subject includes the **Stock symbol** and the Financial Process “**hold**” workflow to the Clearing house
- d) SMTP subject includes the **Stock symbol** and the Financial Process “**approval**” attribute from Brokerage and the financial organization whose Stock is in question
- e) SMTP subject reply includes the **Stock symbol** for an “**approval**” attribute on the Stock in question

## ***Augmenting Behaviors at Runtime***

The behavior that we are augmenting in the service could allow the service to initiate a "new" alternate flow which could add a call to another more “appropriate service” to handle the event. The CEP engine, based on the event correlation pattern identifies the “appropriate service”. In the event that the system is able to find a previously identified event pattern a predetermined rule can be executed whereby the event content parameters get plugged into the rule algorithm. The event pattern is then mapped to an existing rule but the rule could in turn trigger a “new

“service for handling the situation. Thus, the CEP engine can be used to handle an existing situation in a “new” manner. Likewise, new business rules could also be introduced to map to an existing event pattern. One could also add newer business operations via the mechanism of service chaining to handle the situation. It must be noted that new event handler services need not always result in the replacement of an existing business service. In extended enterprise scenarios, these new event handler services could add new event subscriber notifications to third party syndicated business services and/or could trigger alternate B2B business processes.

Regardless, of whether the event correlation pattern results in a new set of rules being executed or whether the rule execution results in new services being invoked they all lead to runtime behaviors that would be classified as “situationally aware behaviors”. In addition, this facility of initiating a new alternate flow or re-routing of the handling of an event to a new business service allows business services to be wired in gradually without massive configuration management challenges thus allowing new behaviors and new execution pathways to be brought on-line in phases.

Alternatively, one could redirect the in-aberration event to an interceptor flow that then "deals" with the “new situation” event information by logging of the event information for later analysis so as to not lose this information allowing inductive logic/ data mining engines to work on this event payload information. Following this the standard portion of the event message payload can be forwarded on to the standard event consumer – thus protecting this event consumer from a previously unknown situation that the standard event consumer would have deemed an exception. If the pattern cannot be identified and/or a matching rule cannot be found then this condition could lead to the execution of an exception flow. Without access to these event heuristics the CEP engine could initiate a manual workflow.

Thus, it can be seen that in addition to protecting the event consumer and stabilizing the event flow logic, use of this dynamic augmentation mechanism to trap unforeseen events actually augments the enterprises' event handling capacity and provides an alternative to invoking an exception flow or even to failing the use case more gracefully as opposed to with an unrecoverable error condition.

### **Basic Flow – Event – Business Process**

1. Trade request occurs (Initial Event)
2. Customer not a Company Official (System check)
3. Trade volume is not too large or is within the trade volume threshold (System check)
4. No company announcement is available to cause any major drop or hike in the stock price (System check)
5. Trade request is processed (System Final Action)

### **Alternate Flow – Event - Business Process**

1. Trade request occurs (Initiation Event)
2. Customer is a Company Official (System check)
3. Trade volume is too large (System invokes a trading service to determine the average volume of trades for that week, for the year, for this time of year, for this industry and for this company) – trade is placed on hold either in response to a regulatory rule or corporate rule.
4. Company announcement is identified that causes a major drop or hike in the stock price following the trade request (Temporal Event of company announcement is correlated with the Customer trade request)
5. Resulting event following temporal event correlations identifies a pattern of suspicious insider trading when the temporal event correlation and event hierarchy is combined with a business rule of the company's financial announcement as non-trivial (System Action per rule)
6. Additional querying to check if the company official's options trade was pre-published and if this is part of the timeline. (System check)
7. Trade is suspended and reported (System Final Action)

## **Alternate Flow – Event – New Sub Flow Enhancement Options (after #4 above)**

*An alternate sub-flow and/or service chaining call could be inserted between steps 4 and 5 above to handle the situation if the company official status cannot be established which could be a disruptive event with no pattern heuristics. The alternative sub-flow could be invoked prior to suspending the trade with specialized analysis being performed for the company stock when the price hike occurs vs. when it drops; especially if the customer is not a direct company official but is related to the company indirectly through a partnership or has other influence such as being an influential commodities trader. Also, this large volume trade could also be turned into another event that could be used to analyze influence over commodities trade or could be analyzed to see if this was in response to another commodities trade event. If the large volume trade is deemed to be in response to another commodities trade as a result of further analysis patterns being established this “suspicious” trade could be released from a suspended status and/ or new events could be published to other relevant parties and event consumers.*

## **Exception Flow – Business Process – Alert**

No Heuristics available and so the trade has to be routed to a broker to execute a manual escalation for more approvals or for extra-scrutiny

1. Trade request occurs (Initiation Event)
2. Customer is a Company Official (System check)
3. Trade volume is too large (System check to see what the average volume trades have been during that week for the year and for this industry and for this company)
4. Company announcement DOES NOT cause any major drop or hike in the stock price following the trade request (Temporal Event of company announcement is correlated with the Customer trade request)
5. Resulting event following temporal event correlations identifies a possible pattern of suspicious insider trading when the temporal event correlation and event hierarchy is combined with a business rule – but the pattern is not completely applicable as the announcement is trivial and does not cause any out of the ordinary dips or hikes in the stock price (System Action per rule)
6. System cannot confirm suspicion of insider trading as announcement is not directly related to the stock prices and not enough heuristics are available for identifying and executing the pre-determinate rule (System Action cannot be determined per rule)
7. Trade request is redirected to a manual work-flow process to escalate the trade approval (System Final Action)

## **Post Condition – Event**

Customer trade is either processed or a manual escalation process is executed or the trade is suspended but the customer account is updated with the status of the trade with information about the trade settlement being sent to the customer as an email notification.

## ***Service Patterns that can be used to infuse Situationally-aware behavior***

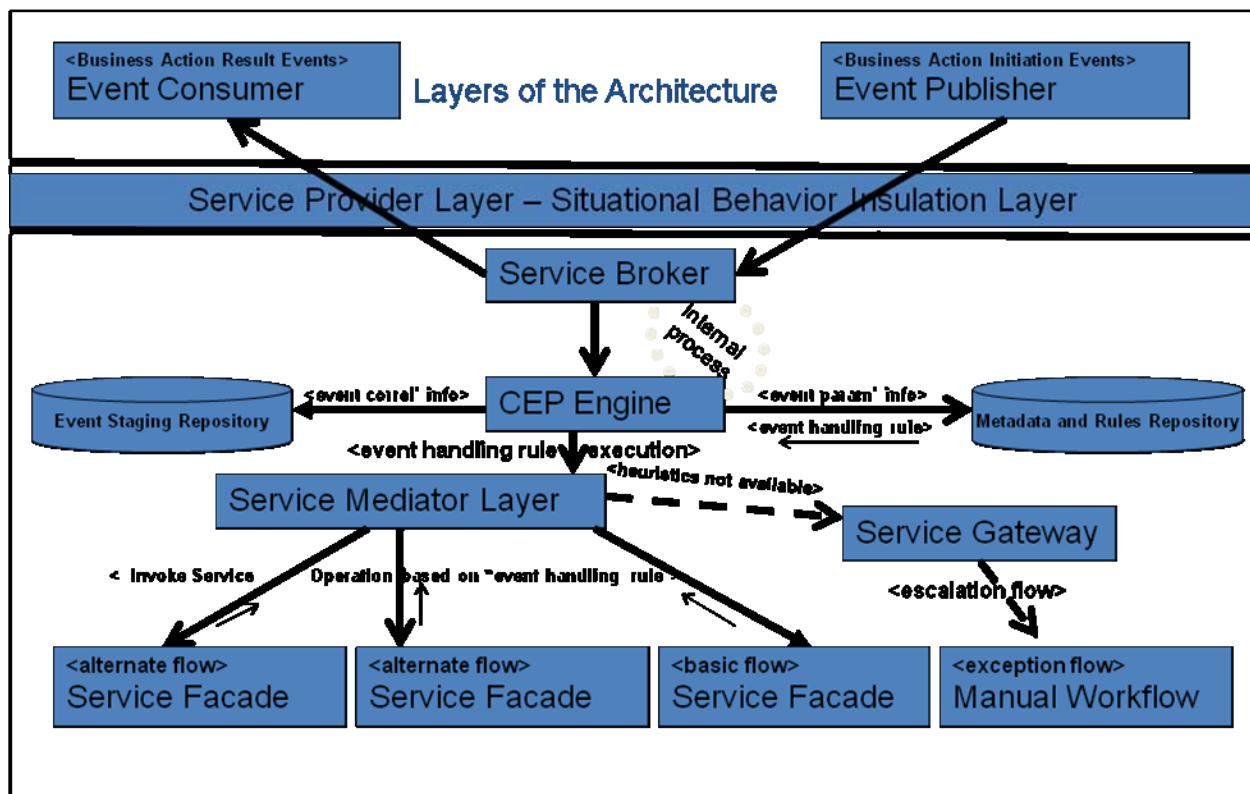
This section outlines a few key design patterns that might be useful in enabling situational awareness.

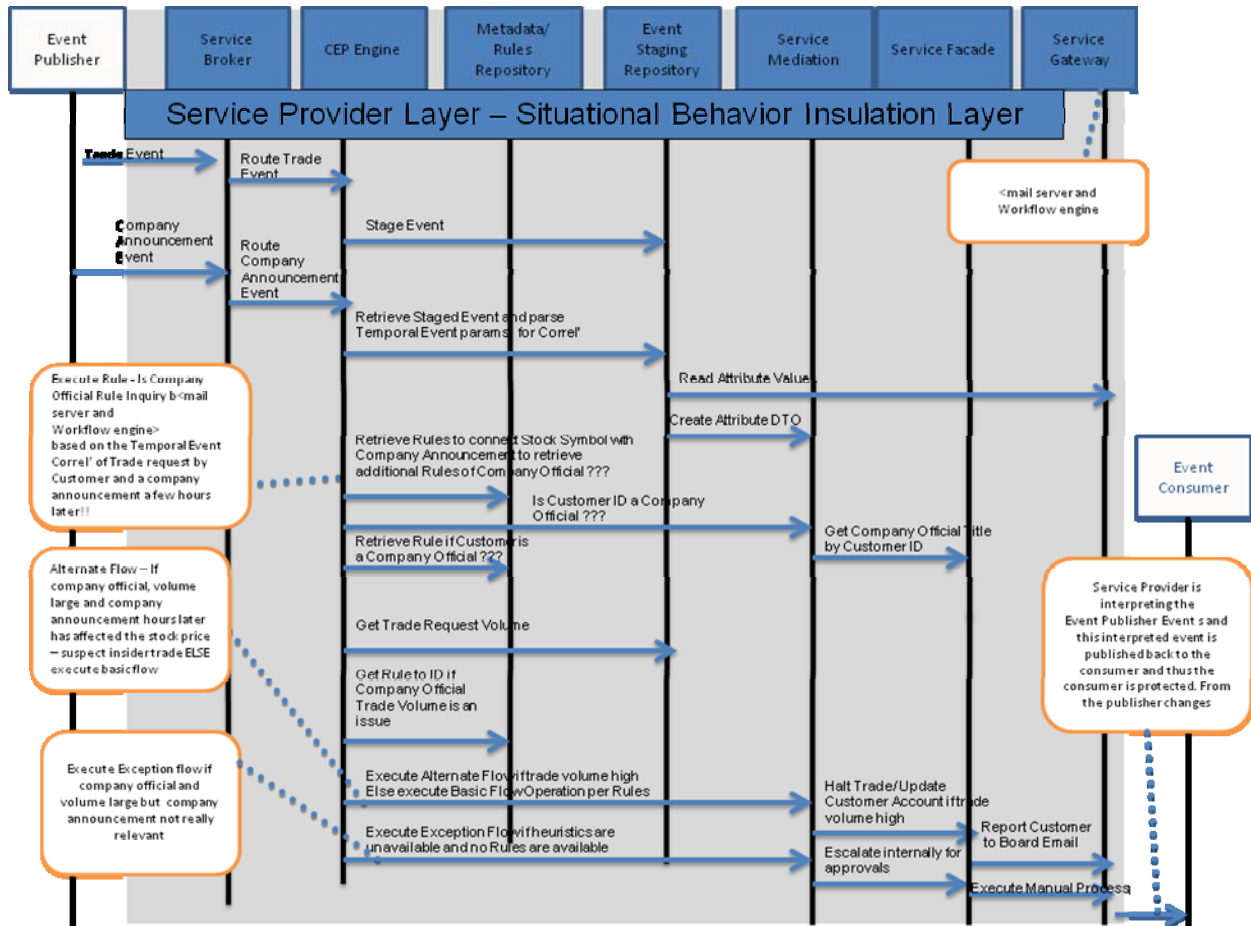
- **Service Provider Layer** – represents all of the components that enable the event publisher changes to be intercepted and re-interpreted into an actionable business event which can be consumed by the event consumer. The goal is to provide a layer of indirection between the event publisher and the event consumer thus insulating the event consumer from new business events that might result from changing business situations or disruptive market conditions
- **Service Broker** - has the ability to broker between the event publishers, various components of the service provider layer and the event consumers
- **Service Gateway** – is the common point of exit from the service provider layer and is responsible for communications between the various process boundaries of an enterprise or between the various process boundaries in an extended enterprise scenario such as in a B2B process

- **Service Façade** – has the ability to create a uniform interface hides details of the event producers or partial event publishers. This layer is capable of aggregating granular notifications and modifying partial events. In addition, this component is able to transform partial event notifications by combining other relevant enterprise contextual information into a complete business event. Here the partial or granular event notifications could be coming from one or more business processes and/or legacy business applications. The service façade is able to provide a single more business relevant point of entry into these more complex business applications and legacy applications
- **Service Mediator** – has the ability to abstract away all granular interactions and cause these to be moved into one common point of mediation that enables altering of the interaction partners, re-wiring of the execution paths more dynamically. This component allows configurable protocol or interaction paths to be defined to prevent hard-coded point to point interactions.

Some of these behaviors may be provided in various degrees in a CEP engine but the concept of layer of indirection as described in the above design patterns are key to insure separation of concerns whether it be between the event publishers and event consumer or between process within or across enterprise boundaries. The constructs of service broker, service gateway and service mediator could also provide points of interception, points to perform audit, security and other common enterprise functions.

Service Provider layer is the highest level of abstraction that insulates the event publishers from the event consumers and the internal components of the enterprise and hence is singularly the most important layer of indirection in this architecture. Not having this layer the common point of entry and interception could make the enterprise services legacy business applications and business processes brittle and prone to disruptions in operations.





\*\*\*\*\*  
 \*\*\*\*\*  
 \*\*\*\*\*

## Conclusion

Again the goal of the paper is to demonstrate that the platform enabling components of EDA, CEP and ESP drive “situational awareness” and hence need to be embedded into the enterprise to intercept external events to find the right event consumers (business process and/ or business applications) to trigger based on the external event correlation patterns.. We hope to have provided prescriptive guidance and the supporting models to attain “situational awareness”. We have tried to demonstrate that by externalizing service and business process sequencing and loosely coupling the services that execute business process behavior the enterprise is able to employ business rules that “dictate” how to execute right time behavior to deal with changing business situations. Here we propose that having individual business events intercepted and analyzed by the CEP engine allow the discovery of event patterns that can then be tied to new business rules that alter the order in which business process steps are triggered and/or new event handling business services are invoked – thus enabling situational awareness.

In conclusion, we hope to have demonstrated how the components of CEP working with EDA allow an enterprise to move to more of the “management by exception” philosophy. Employing CEP based EDA improves the maturity of business interactions to deal with changing business conditions. Here each human exception handling cycle is used

as an opportunity to infuse business rules into the CEP engine to automate the exception handling flows. Automation includes redirection of new business events (such as the handling of aberrations in event thresholds) to alternate service operations or to new business services that are capable of handling these events. In addition, these alternate business services are brought on-line in phases. This intelligence when built into the CEP engine allows an enterprise to wire in new services and processes “on the fly” to handle the needs of a “changing business environment” or to address new regulatory policies. As can be seen, this ability of dynamic rewiring of services to handle “new” business events would make changing business environments less disruptive and thus enable the enterprise to be “situationally aware”.

## About the Author

**Surekha Durvasula** is an enterprise architect who has had 11 plus years in designing and architecting a variety of enterprise applications in the financial services sector and the retail industry vertical. Most of her work has been in the area of distributed N-tiered architecture including J2EE component architecture. Her more recent focus has been on Service Oriented Architecture and Business Process Management. Her efforts as an Enterprise Architect involve not only architecting new applications and business services but also in leveraging the principles of SOA to extend the life of existing Enterprise Information Systems. Contact Surekha at: [surekha.durvasula@gmail.com](mailto:surekha.durvasula@gmail.com)

**Carey Chou** is an enterprise architect with 11plus years of experience leading enterprise wide architecture solutions in retail and consumer goods industry. He has been a lead architect on a number of enterprise wide business transformation initiatives while leveraging technical expertise in SOA, BPM, digital analytics, and enterprise information management.

**Keywords:** Service Oriented Architecture, Business Process Management, SOA, Event Consumer, Event Publisher, Event Driven Architecture, Service Provider, Metadata, Situational Awareness, Design Patterns