Mindtree

Welcome to possible

WHITE PAPER

Enabling predictive analysis in service oriented BPM solutions.

Summary

Complex Event Processing (CEP) is a real time event analysis, correlation and processing mechanism that fits in seamlessly with service oriented Business Process Management (BPM) solutions. Conceived in the early 1990s by Dr. David Luckham of Stanford University, CEP uses technology to predict high-level events likely to result from specific sets of low-level factors. CEP identifies and analyzes cause and effect relationships among real time events and allows personnel to take proactive, effective actions in response to specific scenarios. CEP is used in security policy development, risk management, Customer Relationship Management (CRM), application servers and middleware.

One important aspect of CEP is Business Activity Monitoring (BAM). BAM is the use of technology to proactively define and analyze the most critical opportunities and risks in an enterprise. CEP is especially effective in situations where numerous factors interact in variable ways, such as investment and lending environments for financial institutions. CEP can also be used in threat management for communications networks.

It is evident that CEP is a technology for low-latency filtering, correlating, aggregating and computing real-world event data. The processing of messages as they arrive is called "real time processing" and the use of a sophisticated and optimized storage mechanism is called "historical processing". A good event processing platform is judged by how well it integrates real-time and historical processing.

Contents

Relevance of CEP solutions in a service oriented BPM

CEP solutions are relevant in the following service oriented BPM implementations:

Organizations where processes change frequently.
Here, event-driven dynamic process models, which are typically rule or plan driven, can be used. Changes can be done at rule levels so the core process model remains untouched and rework is reduced. Eg., TIBCO's CEP offering Business Events uses rules and state models to define event processing, as well as managed decisions.

Automating / monitoring processes across the value chain.
Though monitoring capabilities may be built-in at the BPM level, the customer may want to monitor processes against external metrics (like SOA services, external events, etc). That's a role for CEP.

Eg., TIBCO Business Events can model a value chain as a state model, and correlate events in various ways to support business activity monitoring – including monitoring BPM.

Supporting cross-functional processes.
In an enterprise BPM implementation, CEP tools can define and utilize the appropriate meta-model as concepts, and enjoy full flexibility in how it interacts with events, rules and decisions.

Monitoring the status of work in progress.
CEP allows long-running stateful processes – self monitoring processes come as a basic feature. Eg., many CEP products can continuously question the attributes of entities they are processing and deliver results as events change.

Extending the life of legacy apps.
BPM can layer on existing applications and services. The same is true for CEP, which can invoke and control existing applications as well.

Improving inaccurate, inconsistent or laborious manual work.
CEP is mostly about automated processes or monitoring events from manual processes.

Eg., one can implement a state model for workflow queue handling in TI BCO Business Events, but most people prefer to use the out-of-the-box variant from a BPM tool.

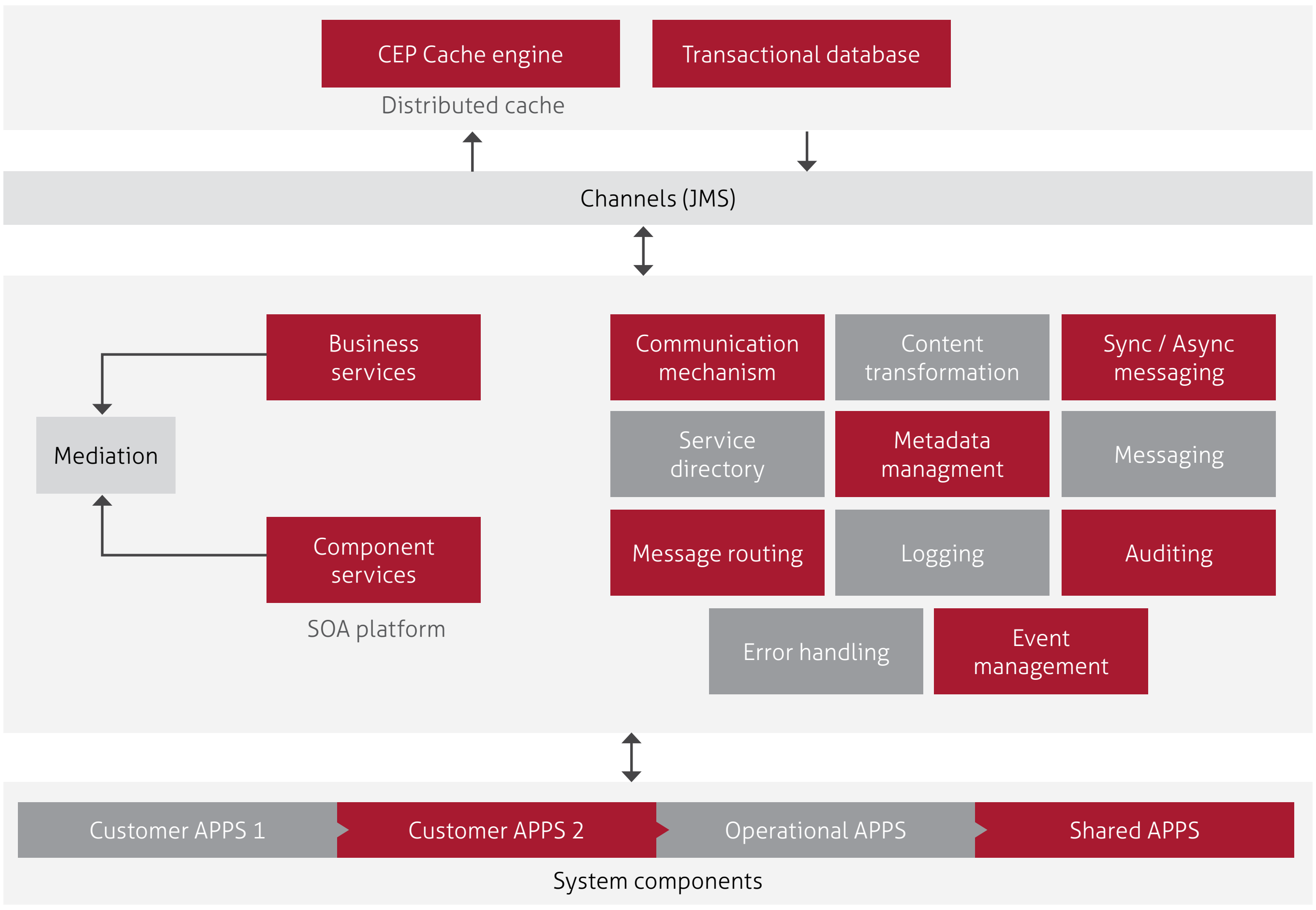
CEP and SOA – how are they related?

Connecting services in a typical SOA environment occurs in a linear and predictable sequence. Event driven architecture on the other hand, allows multiple, unpredictable, asynchronous events to happen in parallel and trigger a single action. An event processing system senses and collects these events and correlates patterns which are disseminated to all interested parties (human or automated) optionally via services. The interested stakeholders evaluate the events and may respond by invoking a service, triggering a business process, or publishing or syndicating further information. Event Driven Architecture (EDA) helps correlate complex relationships of events based on past trends and future predictions. While EDA is often considered a subset of SOA, it is being widely acknowledged as complementary to SOA.

Complex Event Processing (CEP) is an emerging technology that will help companies develop and manage Business Activity Monitoring (BAM), enterprise application integration, network and systems security and business processes. With SOA and CEP, organizations can develop model-driven, agile "sense and respond" systems that can detect events of business value and trigger services to manage the min near real-time. Understanding which applications are optimized for which architecture and how they interface with each other allows for improved architecture. Such an architectural approach enables end-to-end management of business processes and supported functions.

The operational stack supports a distributed architecture. In this the activities of client applications and partner services, both within the organization and outside, are coordinated by orchestration processes. Clients and partners communicate with these processes through the ESB. Internal connections typically use MOM queues to access the bus; external connections use SOAP over HTTP. The orchestration processes, besides coordinating partner activities, also interface with backend systems (databases, mainframes, and so on) and use BPM to delegate manual work to human actors.

The following figure illustrates this architecture:



CEP consists of a set of rules that listen for events from the bus. A publish-subscribe messaging infrastructure allows CEP to get these events without disrupting operational processes. CEP rules, as we discussed above, infer complex events from these operational events. They send complex events back on the bus on orchestration queues, where they are picked up and handled by CEP-aware orchestration processes (see figure).

CEP must be sufficiently scalable to handle the high volume of events on the bus. There is more to this than its sheer processing speed. CEP, like most rule-based systems, is stateful. It keeps its state (i.e., its current set of asserted facts) in a data store called the working memory.

BAM works differently from CEP. Orchestration and BPM processes have state and BAM uses this, combined with business data from backend systems, to present a consolidated view of the state of the processes.

Example, rather than tracking just order processes, BAM might also track the orders themselves, as they appear in the backend order system. In the previous figure, we labelled this piece BAM / BI (for Business Intelligence), because most BAM implementations by themselves lack the ability to incorporate application data. Combining process state with application data often requires the combination of BAM and some sort of BI or analytics tool.

Whereas BAM merely presents a view of the system, CEP, by creating events that it has inferred from its observations, serves as an active participant. SOA platforms with CEP are self-healing; CEP watches what's happening operationally and warns when it detects a significant occurrence.

The dotted line in the previous figure from CEP to BAM suggests that complex events created by CEP could be incorporated into the BAM dashboard. Thus, the BAM dashboard for the trading application could show statistics on completed orders or buy-aheads per broker. The mechanism by which CEP notifies BAM of complex events is implementation-specific.

What CEP and SOA have in common are events. Both technologies use events, but for different purposes. SOA processes use events to drive control flow. An SOA process is started by an event and during the course of its execution waits for further events to propel it forward. Events in SOA, in effect, force process transitions. Most SOA processes not only receive but also send events. When a process sends an event, another process receives it. Elaborate choreographies arise when the processes of multipleorganizations engage in conversations of events.

CEP in action

Example 1:

There are different design choices in an SOA, even when services are already identified. The following example illustrates this:

To cash the process of supply chain management, one starts with placing an order. Take two services: order service and inventory service. The task is to place the order and make a corresponding reservation for the stock level. In this context let us explore few design options (A, B, C):

A. 1. The "process / application" sends a message (sync or async) to "placeOrder" on the order service.
   2. The "process / application" sends another message (sync or async) to "blockStock" on the inventory service.

B. 1. The "process / application" sends a message (sync or async) to "placeOrder" on the order service.
   2. The order service sends a message (sync or async) to "blockStock" on the inventory service.

C. 1. The "process / application" sends a message (sync or async) to "placeOrder" on the order service.
   2. The order service publishes an "orderReceived" event.
   3. The inventory service subscribes to the "orderReceived" event.

The same context above can be viewed in the light of a business service. Three business services related to this are: sales, inventory and shipping.

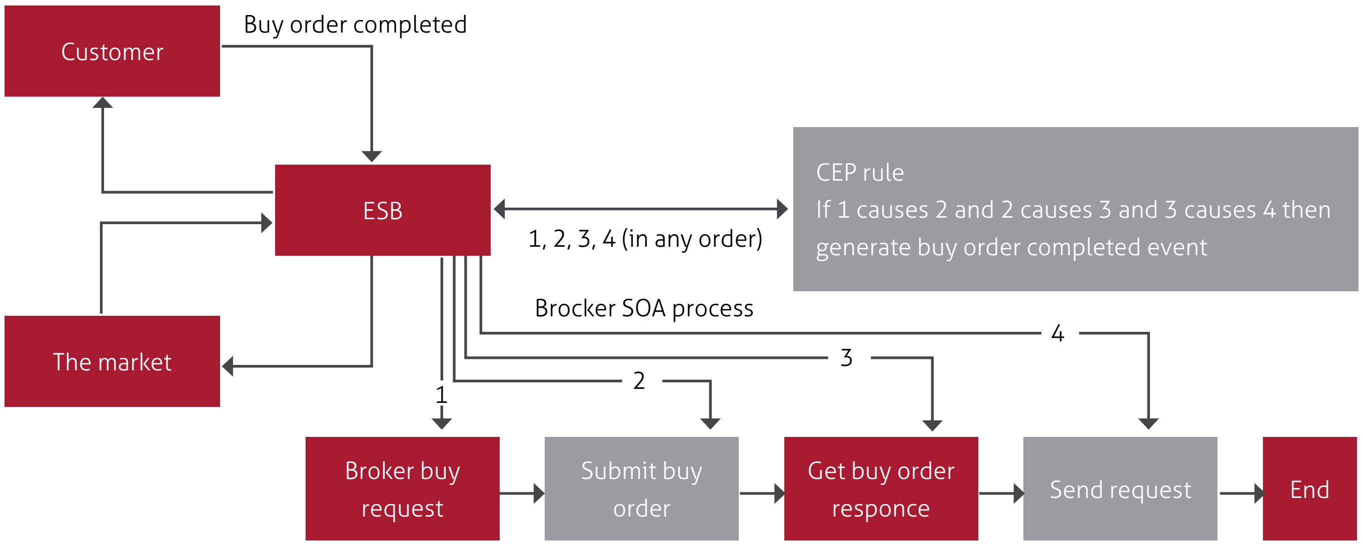
- Many applications and people operate in sales, including the person and the application that was used to submit the order. When an order is submitted, it goes through all the internal validations and sales raises an Order Tentatively Accepted event
- Inventory, which is subscribed to this event, checks if it has everything in stock for the order. For every item in stock, it allocates that stock to the order and publishes the Inventory Allocated To Order event for it. For items / quantities not in stock, it starts a long running process which watches for inventory changes
- When an Inventory Changed event occurs, it matches that against orders requiring allocation – if it finds one that requires stock, it publishes the Inventory Allocated To Order event
- Sales is subscribed to the Inventory Allocated To Order event. On receiving any event pertaining to the order tentatively accepted, it publishes an Order Accepted event
- When inventory receives the Order Accepted event, it generates the pick list to bring all the stock from the warehouses to the loading docks, finally publishing the Pick List Generated event containing target docks
- When shipping receives the Pick List Generated event, it starts the yard management necessary to bring the trucks needed to the docks

Example 2:

In stock trading, a customer's purchase of shares is the combination of four events:

1. A request to the broker to buy the shares
2. The broker's placement of the order
3. The result of the order, including the price at which the shares were bought
4. The broker's response back to the customer

When CEP detects these four events, it publishes a complex event – let's call it Buy Order Completed – back to the bus, where interested listeners may pick it up. The figure in following page illustrates the sequence of steps:



CEP can also detect breakdowns in the buy order. For example, if the broker fails to respond to the customer in time, CEP easily spots this and publishes a Buy Order Broker Response Late event. CEP also spot anomalies that span multiple orders. For example, it can detect suspicious broker activities like a buy-ahead, in which the broker, when directed by a customer to buy shares, buys his own lot of shares first and sells them after placing the customer's order. It would be difficult to build a detection mechanism for buy-aheads into SOA processes; CEP, as a watchdog off to the side, is much better equipped.

This trade monitoring example is one of many CEP used cases. CEP used cases arise naturally from SOA used cases. If events move through the bus to drive services and processes, there is bound to be some important business reason to infer complex events from them. Indeed, many CEP used cases require SOA. Fraud detection, for example, which checks for anomalies in transactions on an account, must reside on an SOA platform with processes in place to perform transactions on accounts. The fraud detection rules must understand the structure of the account events and the protocol governing how they are exchanged.

As it is evident from these examples, many business domains services are invoking / subscribing to events, which are a combination of SOA and CEP. CEP listens on the SOA bus for the same events that pass in and out of SOA processes, but – unlike SOA – is concerned mainly with the detection of patterns that emerge across events. It does this by applying event-pattern matching rules to the events. CEP thus relies on both SOA's event bus and its rules engine.

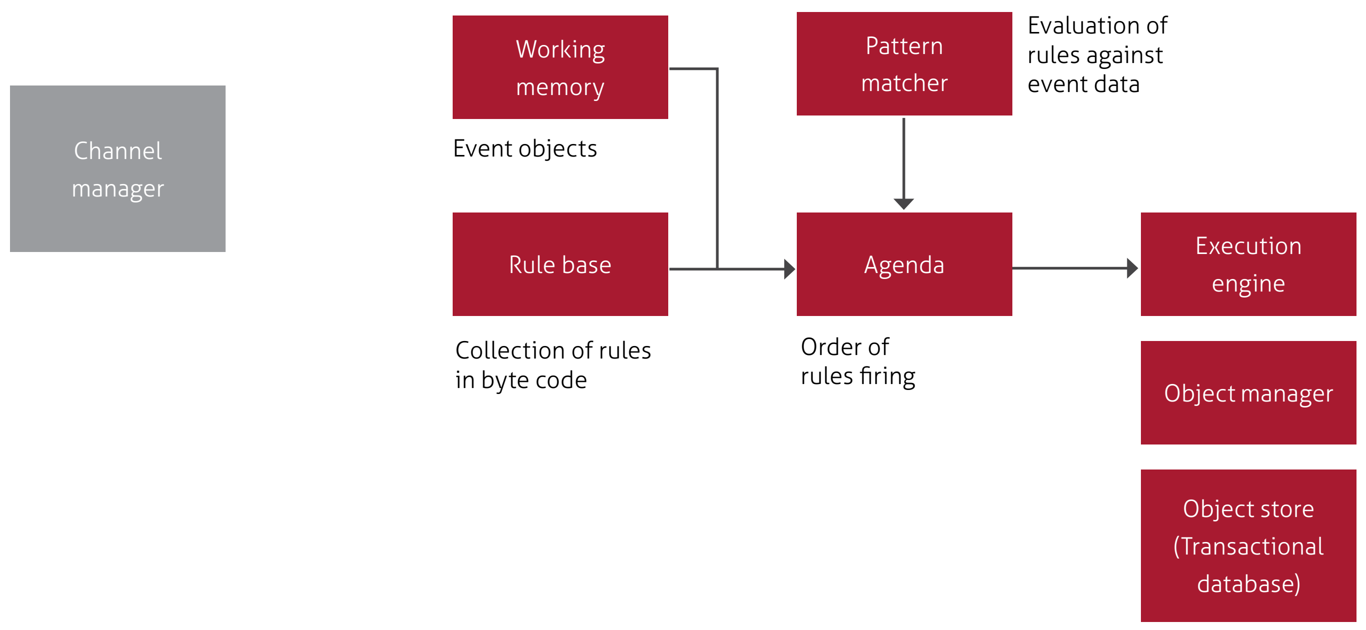
CEP platforms –market status

IInitially the most established commercial implementations came from small vendors, such as Coral8 and Streambase. But within the past few years the major SOA vendors have started to incorporate CEP into their stacks. IBM acquired Coral8 and came up with a CEP offering. TIBCO aggressively sells its CEP tool, Business Events in its SOA and BPM deals. Oracle, thanks to its BEA acquisition, offers a product called "Oracle" CEP.

The combined SOA / CEP offering is encouraging, because CEP's value does not lie solely in itself, but from its contribution to an overall solution. SOA vendors, in a sense, have been in the "overall solution" business for years. They promote the idea that business architecture should be built on services. They also provide a platform on which to build those services. But what's always been missing is a tool to watch the services, to make sense of what's happening operationally.

The figure that follows shows a typical CEP stack offered by major SOA vendors.

Typical CEP stack offered by major SOA vendors.



The channel manager manages all channels on which real time events are received from business systems. Rule base is a collection of dynamic rules that the CEP engine effectively uses for decision making. A decision tree based on industry standard algorithms like RETE is utilized to build and process hierarchical rules. SOA based processes running on the orchestration engine can send uncorrelated and unanalyzed events received via its partner systems to the CEP engine which can process and send correlated, analysed and contextual responses back.

CEP and Business Activity Monitoring (BAM) are not a part of the core operational SOA platform, but rather provide an important business monitoring capability. BAM, in most implementations, is a summary the view of business processes in the system. Like CEP, it observes the progress of operational processes, though it handles this data somewhat differently. BAM's purpose is to create a rolled up, read-only data view. CEP, on the other hand, tries to infer complex events that can trigger follow-on actions. Both the CEP and the BAM funnel have a large volume of events from the operational stack, reducing all that chatter to a smaller set.

The following table maps this stack to the four major SOA vendors: IBM, Oracle, TIBCO and Microsoft.

Vendor	BPM	Orchestration	ESB	CEP
TIBCO	iProcess / Active Matrix BPM	Business Works	Active Matrix Service Grid / Bus	Business Events
Oracle	Oracle BPM	BPEL Process Manager	Oracle Service Bus	Oracle CEP
IBM	Websphere Process Server, FileNet	Websphere Process Server, Websphere Interchange Server	Websphere Enterprise Service Bus, Websphere Message Broker	Websphere BusinessEvents
Microsoft	Sharepoint BPM	Biztalk Server	Biztalk ESB	Microsoft CEP

About Mindtree