SOA, EDA, BPM and CEP are all Complementary
by
David Luckham

PART II

In part 1 (link) we argued that SOA and EDA are complementary architectural design concepts, and that the fusion of the two, event-driven SOA (or ED-SOA) should be the design philosophy of both SOA and EDA today.

Now we come to the other two technologies, Business Process Management (**BPM**) and Complex Event Processing (**CEP**). We argue that they are complementary with SOA and EDA, and that CEP is an essential component of all three.

First, lets discuss the BPM world. Business processes have been event driven since the days of workflow. Nowadays, using modern BPM systems, we can build processes so that not only is the execution of the process event driven but so also are its performance monitor and control. Events that are used to trigger and synchronize the process steps are also fed to a process monitor. The monitor uses them to check progress, track performance, and issue alerts to human operators when the process starts to misbehave. Furthermore, the monitor can interact with the process using events to keep it on track. This completes a feedback control loop, a step towards making the process fully autonomous.

The easiest way to build a new process is to use an appropriate ED-SOA, i.e., one that offers services needed by the new process. Figure 3 shows an event driven process for handling mortgage loan applications. It uses an ED-SOA that provides services such as credit checking and house appraisal. The loan process executes by triggering the first three steps simultaneously. When the process is initiated to handle a new loan application, it sends request events (blue arrows) to three services, income review, credit check and house appraisal. These services are themselves processes. These sub-processes send their results (red arrows) to a risk management service. And its results go to a funding service. The mortgage bank may own the loan process, but all the sub-processes could be services provided by other organizations.

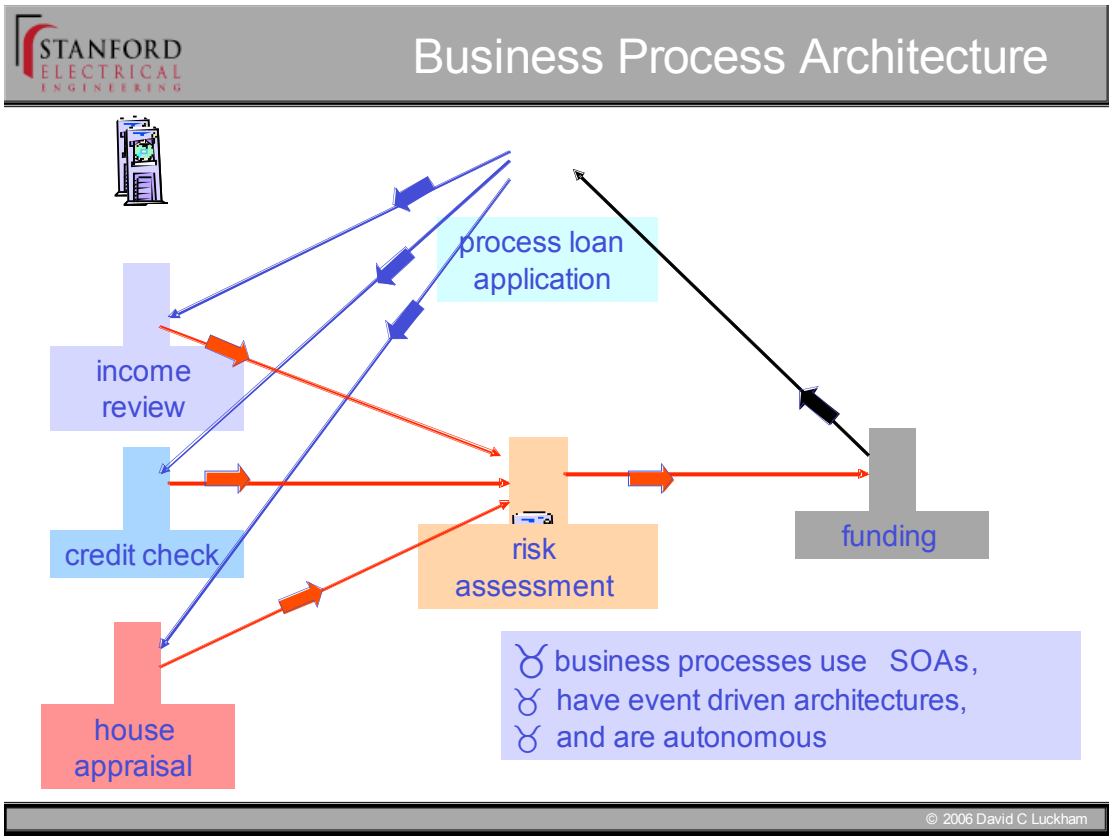

Figure 1: An event driven business process

So the loan process in Figure 3 has been built using five services. Put another way, it out sources the steps in its process. Typically these services would be specified in the interfaces of various service providers that are members (or components if you like) of an ED-SOA specializing in loan processing. The loan process simply orchestrates the activities of the sub-processes by telling them what data to use and where to send their results. As illustrated here, ED-SOA makes the job of building event driven business processes easy.

Conversely, processes are a natural way to extend the power of ED-SOAs. The loan process itself can be viewed as an addition to the ED-SOA it is built upon. The mortgage bank could offer it as one of its services. In fact, we're seeing the layering of EDAs in this example. That is, event-driven BPM is a way to build ED-SOAs.

But now we have a downside! ED-SOA and BPM have made building enterprise applications so easy that the scale of event activity in the world has climbed dramatically. The modern enterprise operates in a cloud of business events emanating from sources (and outsourcing) all over the world. We must now strive to understand how our own business processes are behaving, and also how

other people's processes are affecting ours. After all, it's a competitive world out there! Moreover, keeping control of our processes is compounded by a drive towards autonomous processing to handle real time demands.

This is where complex event processing (**CEP**) enters the mix. CEP defines principles for building tools to design, monitor and manage event processing systems.

One of the first principles of CEP is the use of *event patterns* – in fact complex patterns – to monitor event activity. Here are two simple examples. The first is the use of event patterns to monitor stock markets for conformance to regulations. This kind of monitoring for illegal activities such as "front running" is being deployed in various markets today. See figure 4.

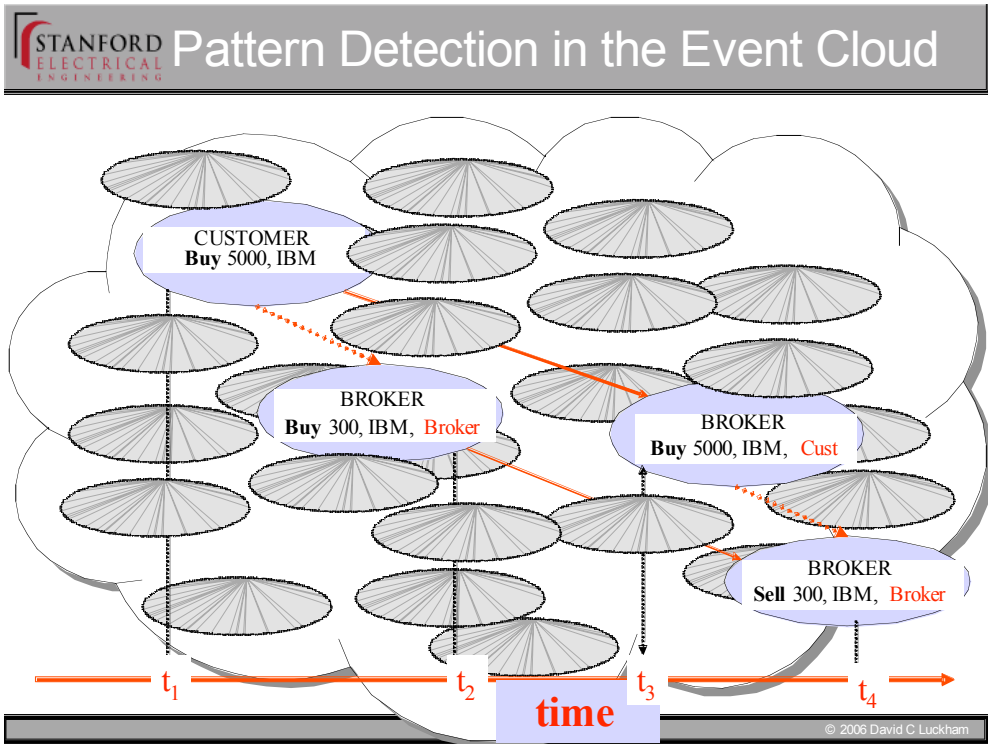

Figure 2: Detecting trading ahead in the cloud of stock market events

The pattern is a template for four events related by causality (red arrows) and time. The first event in the pattern is a buyer issuing a **buy** instruction to a brokerage for an amount of a stock. All the parameters in the pattern (buyer, stock, dollar amount etc) are open. The pattern matching engine replaces them by the values in **buy** events to match those events as they arrive at the engine.

What's important is that any instance of this first event must happen earlier than any of the other events in the pattern, and it must cause the broker to buy the stock for the customer[1]. Time wise, the next event in the pattern is the broker buying some of the stock for its own account, followed by executing the buyer's instruction, and then followed by selling from its own account on the up tick created by the buy order. Any instance of this pattern involves four events happening with the required timing and causal relationships. By the way, the dotted arrows indicate "don't know" (or uncertainty) relations between events in the pattern. So, this pattern specifies that the customer issues a buy order to a brokerage house *before* the brokerage buys the stock for its own account, but that whether the first event *caused* the second event may be unknown. Complex event patterns may involve, time, causality, and, among other relations between events, uncertainty.[2]

This kind of conformance monitoring is being installed across multiple market feeds, truly an event cloud. It involves real time detection of event patterns like this example in a cloud of 100,000 events per second. The scalability issues can be imposing. Bear in mind that the detection engine may be monitoring many partially instances of the pattern simultaneously to see if any of them result in a complete match. And events that match a pattern don't necessarily arrive at the engine in the order (time or causality) in which they are related.

Another application of CEP is the use of event patterns in monitoring and autonomous feedback control of processes. Figure 5 shows the pattern of business events resulting from completing steps in one instance of our loan process. Causality (red arrows) and completion times are shown. Figure 5 also shows a service level agreement (SLA) that requires a decision on the funding of every loan application to be made within three days. The problem here is to keep the process from violating the SLA.

SLAs can place constraints on the information the process must deliver, its timing and other performance factors. In the real time world, the problem of keeping a process within such constraints can boil down to milliseconds. This is where the human operator has to be taken out of the loop and autonomous control introduced.

---

[1] It won't necessarily be the first event in a match to arrive at the pattern monitoring engine. Events do not necessarily arrive in their causal or timing orders. The engine has to allow for this.

[2] . High level event pattern language design is another topic in CEP, but we can't deal with that here.

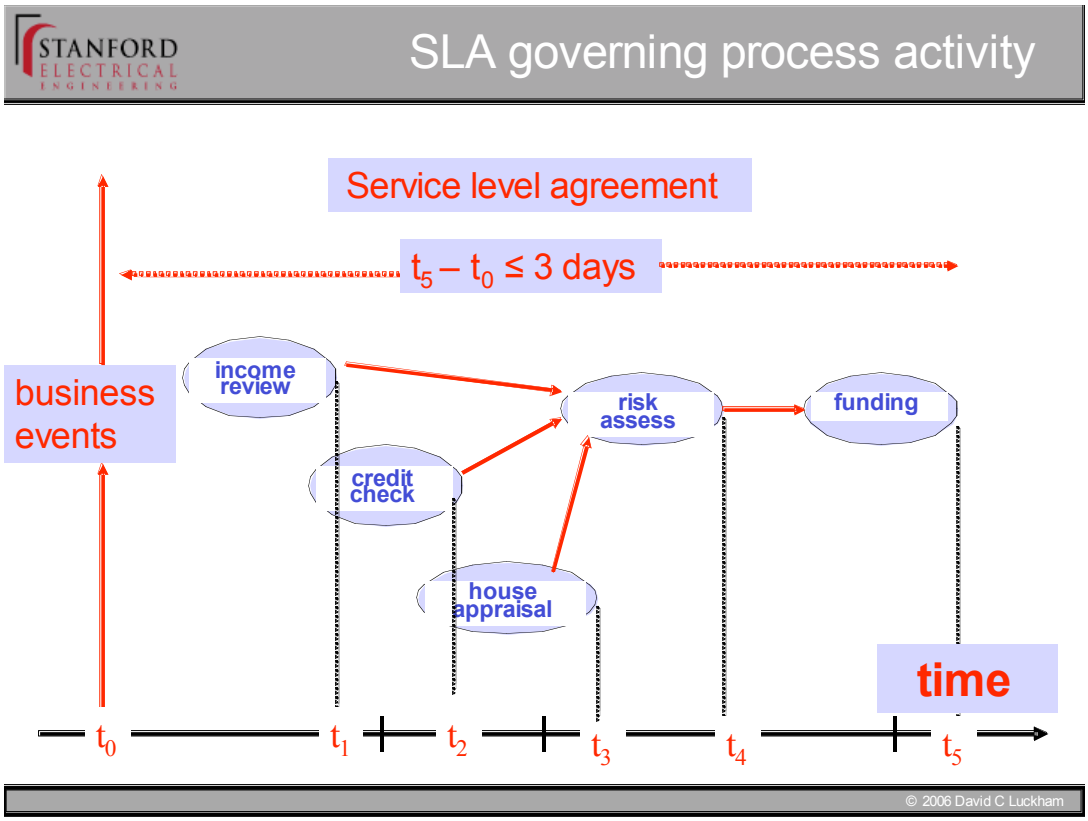

Figure 3: Pattern of events corresponding to steps in a loan process.

Remember that hundreds of these loan applications may be in the system at any one time. So we need an automatic way to keep them within the agreement. Figure 6 shows a monitoring process that "watches" instances of the loan process whenever the loan amount is a million dollars or more, and takes an action if an instance seems to be in danger of violating the SLA. The monitor has an event pattern trigger that matches any instance of the first three loan process steps. This pattern matches events signifying completion of the income review, credit check and house appraisal steps for the same loan. It requires that the steps are executed in parallel ($\|$ relation, meaning the events are independent of one another). If the time for these three events to happen exceeds a preset bound then the overall process instance for that loan is judged to be in danger of violating the SLA. The monitor is triggered and an action is taken – the priority for executing the risk assessment step is raised. The value of the bound will vary with system load. Hopefully this feedback will keep the SLA from being violated.

The monitor is often a separate process deployed in a business activity monitoring application whose job is to watch over the business processes. However, if the BPM system incorporates complex event pattern definition and matching, the monitor could be built in as part of the loan process itself. CEP is being included in BPM systems, and this allows us to include the monitoring

process as part of the loan process itself. If we can do this, the loan process is a step closer to being autonomous.

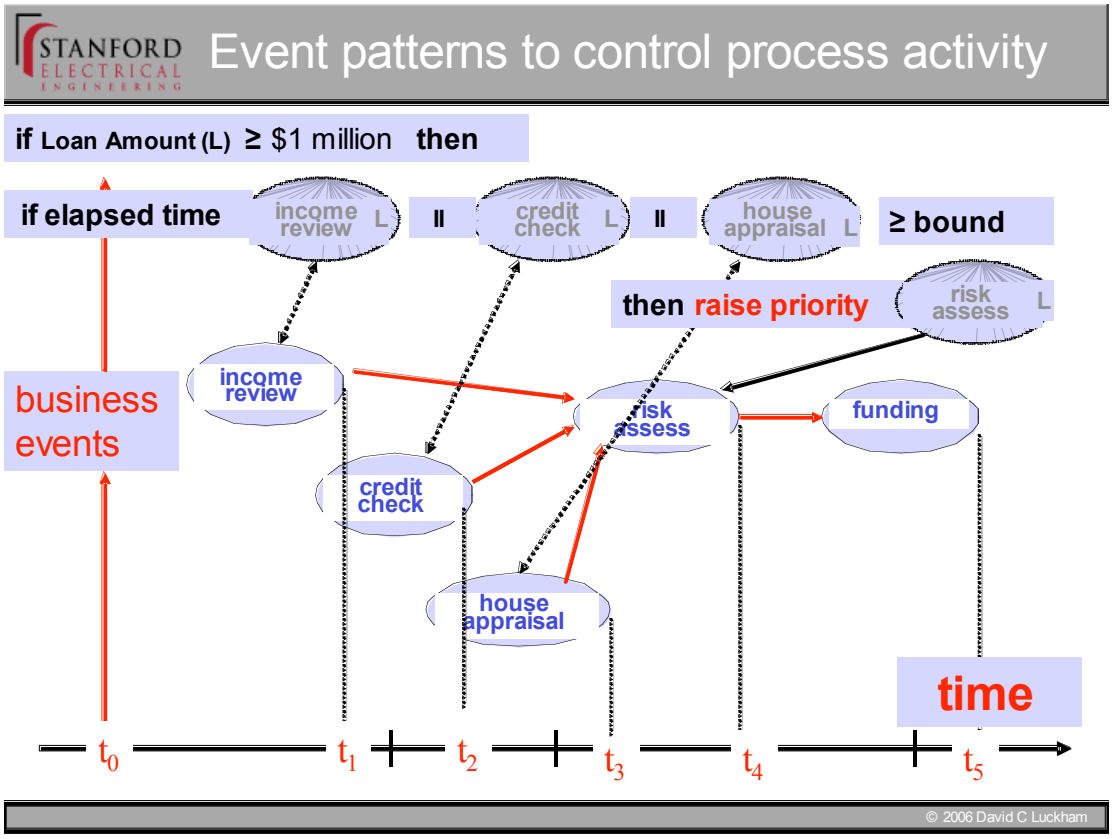

Figure 4: Event pattern monitoring to control process execution

In summary,

(1) the fusion of SOA and EDA into ED-SOA (event-driven, service oriented architecture) is the way of the future,
(2) building processes is greatly facilitated by ED-SOAs,
(3) conversely ED-SOAs can be constructed as layered architectures using BPM systems.
(4) CEP principles must become an integral component of both ED-SOAs and business processes because of the ever increasing quest for control of our business processes, real time autonomous operation, and the need to gather business intelligence from the events flowing through our IT systems.

Acknowledgements

I am indebted to Roy Schulte and Tim Bass for conversations on these topics.