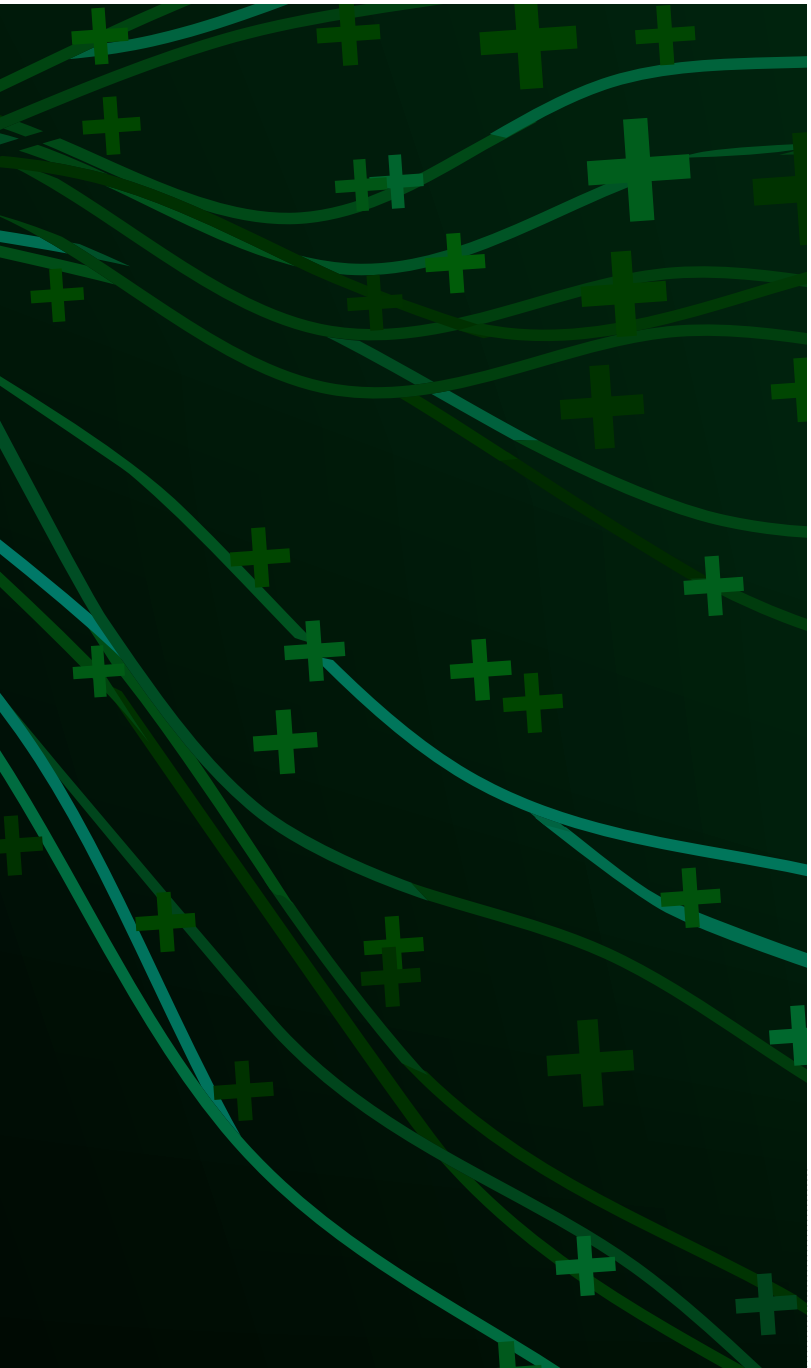
SearchSOA

e-guide

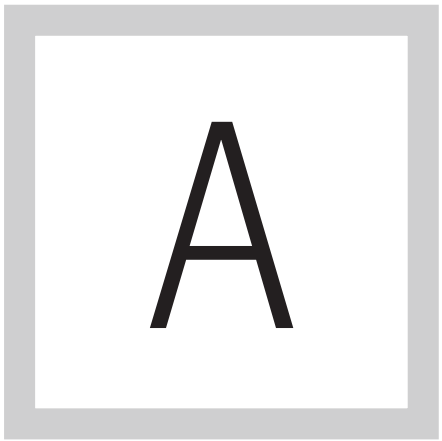


Software Development University

APP MODERNIZATION AND TRANSFORMATION

TechTarget

SearchSOA



S SOFTWARE FOCUSES increasingly on self-service and mobile worker empowerment, enterprise architects may even be breaking the SOA-REST model of the present to create a new binding component for application modernization.

SearchSOA

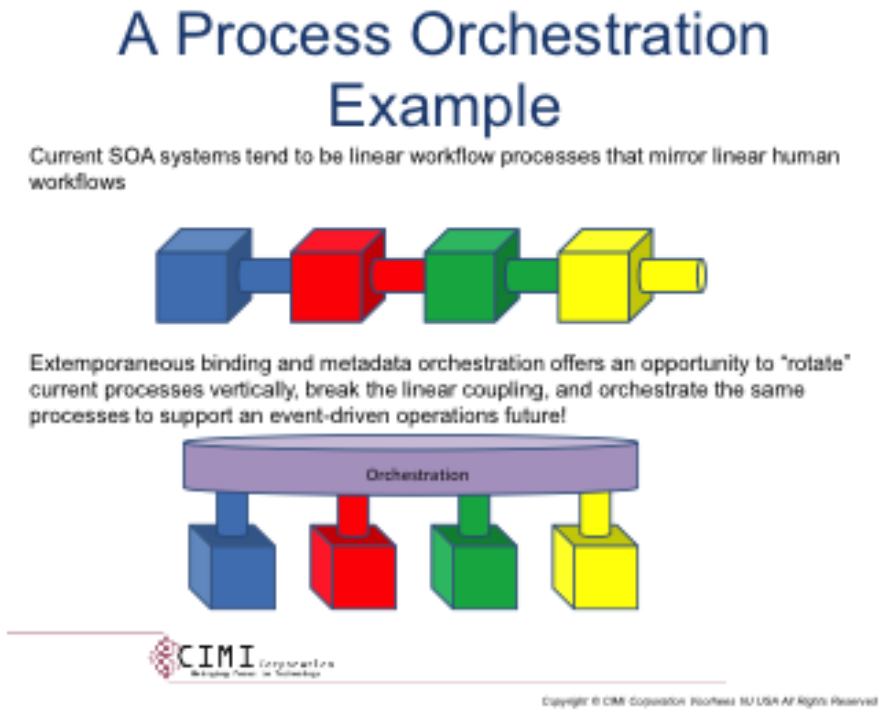
In modernizing applications, binding components to suit event-driven models of execution is now crucial to supporting business processes. Indeed, there's nothing more fundamental than component binding to modern software design and evolution. As software focuses increasingly on self-service and mobile worker empowerment, enterprise architects may even be breaking the SOA-REST model of the present to create something new, an "extemporaneous" component-binding model.

Components were once viewed as rigidly linked and later as loosely or indirectly coupled. In today's evolution, binding components starts with considering components as event-handlers, evolves to effective linking of process to context, and then becomes a means to implementing and sustaining extemporaneous applications.

Probably the biggest change in application requirements architects face is the transition from a "workflow" orientation to an "event-driven" orientation. The former approach to applications supposes that business processes are synchronous evolutions from predictable starting points to completion, with orderly transitions between application elements along the way. This has proved inadequate for many applications driven by online activity, which have to process things as they happen. An example of this is that a four-step

SearchSOA

progression from order to fulfillment might have been seen as a linear process that sequences specific steps, but is now seen as four steps driven by independent events that ships something when every component has completed its work.



Traditional message bus and business process orchestration strategies don't support event-driven activities well for the obvious reason that they are

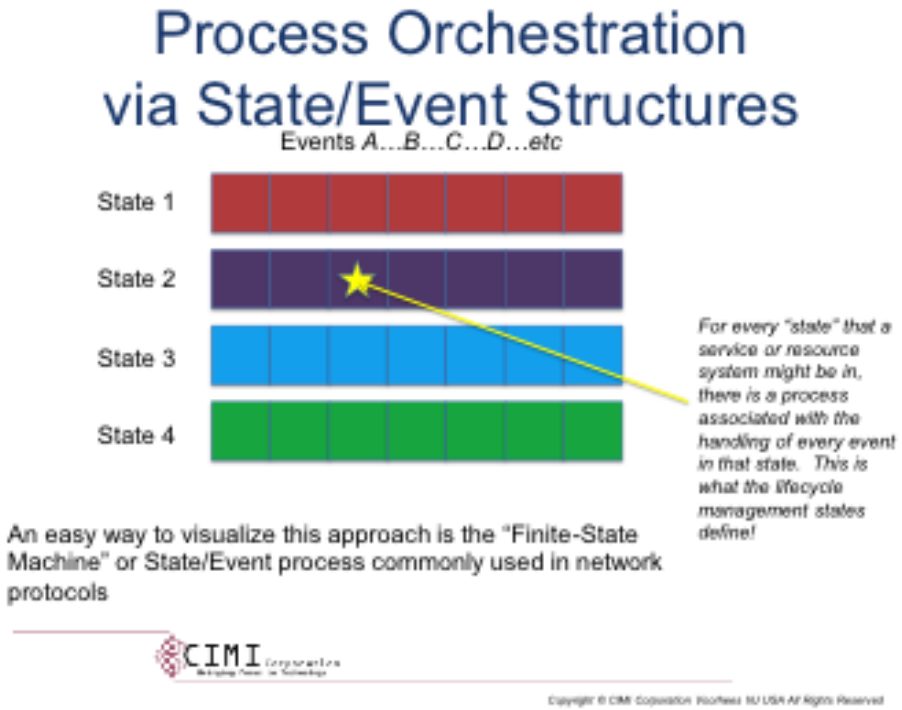
self-sequencing. However, simply embracing RESTful interfaces instead of SOA doesn't necessarily address the issues either. The challenge is that notion of "every component completing its work." If a business process ends when a collection of conditions are met, then the processes themselves must be able to determine when that's true, which means applications themselves must be context-aware.

Context awareness isn't the same thing as statefulness, it's simply that some component of an application understands the total set of conditions it needs to understand and can make decisions on that basis. "Inventory-Available" plus "Order-Received" equals "Pick-and-Ship" is an example of a "context equation." If an order is received before inventory is available, that means that when inventory becomes available the decision to "Pick-and-Ship" is made. The relationship between components is looser in this example; any event can drive a sequence of component executions, so it's important that the component linkages not anticipate a preferred workflow.

Loose coupling or "binding" has been a trend, but the notion has focused on making it possible to easily reuse components across applications or to compose linear workflows. When you introduce contextual event processing into the picture you need to think beyond traditional loose coupling toward

SearchSOA

something that could be called “extemporaneous binding.” Such a binding strategy relies on the notion of application context driving event handling in some way. There are two broad strategies for extemporaneous binding; the traditional state-event or FSM approach and a more novel and modern policy-driven binding model.



In an event model, the application process is represented as a finite-state machine with specific operating states. For example, the model for the simple order entry application might have an "Inventory-Present" state, a "Waiting-for-Inventory" state and an "Order-Pending" state. If an order is received in the first state the result is a "Pick-and-Ship" but if it's received in the "Waiting-for-Inventory" the context changes to "Order-Pending" where a "Received-Product" event now triggers the shipment. State-event processing is often driven by a table of events, making it easy to see how a given event will be handled, but the tables can be tricky to construct and require careful testing to insure that the progression of context through the possible states is correct.

The policy-driven binding expresses the handling of an event in terms of policy conditions. There is no artificial intermediary concept of a "state" or context because any event is handled based on its own complex set of policies, which can relate to the status of other components of the application. An "Order" event when the inventory state is "Available" would then invoke a "Pick-and-Ship" and a "Received-Product" event with orders pending would do the same. Because it's not necessary to define specific contextual states the process of building complex event handling descriptions is more straightforward—just define the policies needed.

The challenges with policy-driven binding are related to policy completeness and execution efficiency. It's easy to define a set of policies for events that will leave some combinations of context and events received undefined. In this situation, the application will either do nothing at all or take some undesirable default path. Inspecting policies to determine whether they are complete becomes more difficult as complexity rises. As complexity rises, the execution of policy statements may also become more time-consuming which can impact performance. There are few tools available to manage either policy completeness or execution efficiency at this point, which means that architects will likely have to employ careful development practices and rely on good design to overcome these issues.

Both event and policy-driven binding of components work best with RESTful interfaces and back-end or database-mediated state control. That means that component design to support one model can be easily adapted to support the other as well. The key point is that all extemporaneous binding concepts rely on transaction state maintained per logical transaction. An "order" or a "material receipt" is a record of context as well as a record of activity, and it's important to treat it as such. Some call this a "data and metadata" merging.

At present, state and event modeling for extemporaneous binding of events

to processes is likely easier to implement and document, but as the shift toward event-driven software accelerates there are certain to be more tools available on the policy-driven side, and it seems likely that this approach will be preferred. Coupling data modeling and process modeling together in a single architecture is a particularly promising direction because policies based on data conditions can then be easily linked to events. This is likely the future of component binding, so architects should monitor the space carefully for developments.

SearchSOA

FREE RESOURCES FOR TECHNOLOGY PROFESSIONALS

TechTarget publishes targeted technology media that address your need for information and resources for researching products, developing strategy and making cost-effective purchase decisions. Our network of technology-specific Web sites gives you access to industry experts, independent content and analysis and the Web's largest library of vendor-provided white papers, webcasts, podcasts, videos, virtual trade shows, research reports and more —drawing on the rich R&D resources of technology providers to address market trends, challenges and solutions. Our live events and virtual seminars give you access to vendor neutral, expert commentary and advice on the issues and challenges you face daily. Our social community IT Knowledge Exchange allows you to share real world information in real time with peers and experts.

WHAT MAKES TECHTARGET UNIQUE?

TechTarget is squarely focused on the enterprise IT space. Our team of editors and network of industry experts provide the richest, most relevant content to IT professionals and management. We leverage the immediacy of the Web, the networking and face-to-face opportunities of events and virtual events, and the ability to interact with peers—all to create compelling and actionable information for enterprise IT professionals across all industries and markets.