

# Fighting Latency: Event-Driven Development with Sybase® WorkSpace

MIKE WEADLEY  
DIRECTOR, ENGINEERING  
SYBASE, INC  
[MIKE.WEADLEY@SYBASE.COM](mailto:MIKE.WEADLEY@SYBASE.COM)

## INTRODUCTION TO WORKSPACE

Most Information Technology (IT) workers know what any good general building contractor knows: There is no one tool, no single correct approach for projects. The size and complexity of an application, the skills of the IT workers, the way IT is organized within the overall business, the degree to which the application is strategic or tactical and dynamic or stable, the historical precedents for successful applications within a business—all of these factors affect the selection and application of development paradigms to solving business problems.

Sybase recognizes that business needs—not tools—should dictate the development paradigms used in an application. In response, Sybase offers a product called WorkSpace that offers support for multiple development paradigms—Model-Driven Development of Applications (MDDA), Service Oriented Development of Applications (SODA) and Visual Rapid Application Development (VRAD)—all within the same environment. Developers can choose the right combination of tools and functionality based on the characteristics of the target application. They can model entire applications from the top down, model only portions of an application, or skip modeling entirely. WorkSpace provides high-level graphical productivity features throughout, but it also offers the flexibility of writing low-level code where it's needed.

Although WorkSpace is a service-oriented development environment, it does not force developers to be experts in SODA. They can easily build services that access to diverse information sources using a variety of methodologies. For example, a developer can implement a service with low-level Java code, or model the service from the top-down *before* jumping into the code. Also, WorkSpace wizards support rapid generation of services that encapsulate a variety of existing resources, such as database stored procedures, Enterprise JavaBeans (EJBs), and Java classes. Other wizards quickly create new services for querying databases, and for reading from and writing to JMS queues, file systems, and e-mail providers. Using WorkSpace wizards, developers of all levels of experience can easily implement, test, package, deploy, and publish a broad set of services.

Moving up the application stack, WorkSpace also offers many ways to consume services within a variety of application types, including Java Server Faces-based Web applications, composite Java applications, and true service orchestrations. Moving even further up the stack, these service consumers can themselves act as providers, leveraging SOA principles to easily expose sophisticated new application functionality across application and organizational boundaries.

## SECURING AN INFORMATION EDGE

Pick up any IT trade publication today, scan the articles, and a common theme will quickly rise to the surface: To beat their competition, companies know they must make better use of information. To secure that information edge, the obstacle they face is “latency”—the time that elapses between a stimulus and its response. For example, mobilizing a sales force is about reducing or eliminating the time between a customer request and its fulfillment. Managing IT risk effectively is about reducing the window between new transactions or events and the point at which their effect on the business is understood and resolved. Applications described as “real-time” and “agile” reduce latency; “just-in-time” applications optimize resources within an acceptable latency; and best of all, “predictive” or “proactive” applications seek to eliminate latency entirely, by generating a response *before* the stimulus.

Sybase has long been focused on reducing and eliminating latency, as shown in the company's market-leading offerings in the replication and mobile areas. This focus informs the Unwired Enterprise vision of open, cross-platform solutions that securely deliver information anytime, anywhere.

So naturally, developing applications that combat latency is also a focus of the new WorkSpace product, which provides Information Liquidity Models to model the replication of information from a source database to one or more remote databases, including databases hosted on mobile devices. For more than a decade, Sybase replication products have been used to address problems of latency in the most demanding financial environments.

## **TACKLING LATENCY WITH EVENT-DRIVEN APPLICATIONS**

WorkSpace supports event-driven applications that are designed to fight latency problems: To reduce latency, these applications automate reactions to business events—even anticipate business events. In event-driven application architectures, it is possible to link many different business application “silos” by orchestrating the flow of events through these silos. Because SOA secures reusability through shared interoperability patterns rather than through concrete implementation patterns, SOA design principles can effectively be used to develop a wide variety of application types, including event-driven applications. Event-driven applications can leverage SOA principles for integration with each individual silo, but they differ from typical service-oriented applications in that orchestrations do not depend upon shared assumptions about interfaces and event sequences in order to function. As a result, event-driven applications can relate events across more silos more easily than service-oriented applications. Also, event-driven orchestrations are frequently asynchronous and can run in parallel for optimal performance. Therefore, event-driven applications have more information at their disposal—more insight about the fully digested consequences of a business event—and can distribute this new insight more efficiently than applications that are only service-oriented.

## **EVENT-DRIVEN DEVELOPMENT WITH WORKSPACE**

Optimizing inventory levels—striking the right balance between satisfying customer needs and controlling cost and waste—is a problem that spans industries. Satisfy customer needs and you grow customer loyalty, but if you do so at too high a cost, the business won’t sustain itself. Optimization requires tracking customer activity, anticipating future demand, factoring in local and regional variations, and coordinating with partners farther up the value chain. Balances must be struck between software-generated insight and the insight of key decision-makers, and between (over-) reacting to every event and identifying the most critical events.

Building event-driven applications starts with capture of a business event, which is defined by Gartner as a meaningful change in state relevant to a business that is generated by software. Good examples of meaningful state changes are placing a new order and recording a sale. These business events could be recorded within any type of software, including databases and ERP systems. To enable event-driven applications, the state change must be captured in some form and published to the event-driven application. Usually, this is achieved by producing a message that conforms to some known schema and by serializing that message using some form of messaging (for example, MOM or Web services). Sybase products can capture business events from any database and deliver them to any JMS-compliant messaging infrastructure; WorkSpace provides full support for modeling and wizard-driven implementation of these solutions. For example, the WorkSpace Real Time Messaging wizard can be used to easily insert SQL code into a database stored procedure or a trigger to publish a message about a new sale to a queue or topic. Using Sybase modeling tools and wizards, the content of the message can also be easily defined as an XML Schema Definition (XSD). The net result: By immediately distributing meaningful business events, a company can avoid other approaches with higher latency, such as applications that poll intermittently for changes and nightly batch extracts.

Once a business event is published, event-driven applications must consume the event. Instances of event-driven applications are automatically launched upon event arrival over any of a variety of transports, such as JMS, SOAP, and e-mail. WorkSpace tooling supports definition of these transports, as well as “late binding” of event consumers to the inbound transport. Design and development of the event-driven consumer only requires an abstract definition of the event (an XSD), and association of the consumer with a given transport occurs later, during deployment. Other than identifying a transport, no user intervention is required to associate events with consumers: WorkSpace handles configuration of the entire lower-level infrastructure for the user.

Event-driven consumers use “simple” bridging services as well as “higher-order” services. One feature that distinguishes event-driven applications from simple service-oriented applications is the independence from tight interface dependencies. Consequently, transformation services (services that transform message content from one form to another) are critical to enable the desired loose binding between simple services. Because event-driven applications operate in the space between application silos, occasionally new logic services are required to supplement logic that exists within the silos. The key distinguishing feature of event-driven applications: The flow-of-control is directed by the event-driven application instead of by the application silos, so the capacity to coordinate flow is also required.

## **USING SERVICE WIZARDS AND EDITORS FOR ORCHESTRATION**

WorkSpace makes working with this array of services—simple transformation, custom logic, composite applications, and orchestrations—masterfully straightforward. Visual development metaphors are used for each kind of service, to gracefully handle both the shared similarities between service types and their unique distinctions.

Service creation wizards and service editors share a common look-and-feel, including representation of a service’s abstract interface. Yet each service editor has unique implementation views tailored to its particular domain:

- Implementing orchestration features a visual canvas based on the Business Process Modeling Notation (BPMN).
- Implementing a custom logic service is based on the popular Eclipse Java IDE.

A Service Explorer view automatically organizes information about all services used in an application, including services available locally, services from other team-member projects, and services defined in network registries. WorkSpace supports drag-and-drop operations out of the Service Explorer onto any service or application capable of service consumption, such as JSF-based Web pages service orchestrations, and Java composite applications. The Service Explorer makes clear distinctions between services that are private, used within an application, and those that are public, available for consumption outside the application.

Using WorkSpace wizards and editors, defining a new service orchestration to consume the business event takes only seconds. Simply create the service, add an operation, and, using drag-and-drop, associate the XSD describing the event with the operation. Then, shift to the implementation tab to begin diagramming the orchestration. In the previous example, consuming a sale event, the developer might start by dragging a set of “simple” service operations from the Service Explorer on the canvas to check on one silo to determine inventory for items sold, another to determine if any items are on sale, another to assess recent sales history for the item, another to update the order planning projections, and another to send an SMS message to the persons responsible for ordering inventory when exceptional conditions arise. Variables holding data used in these operations are automatically defined in the orchestration and the services are automatically associated with the orchestration, to ensure that the appropriate metadata and artifacts are deployed seamlessly. Activities for flow-control (splits and joins), variable assignment, nested orchestrations, or timers, are dragged from the tool palette onto the orchestration canvas, where the developer can begin wiring consumption of the initial event to the subsequent series of activities. When invoking an operation requires data content in a form that doesn’t already exist in the orchestration, the developer can easily create new transformation services to precede invocation in the overall flow. Because WorkSpace supports definition, parsing, serializing, and transforming both XML and non-XML, the developer can easily and visually define an orchestration to calculate the optimal order quantity and supplier, and then update an application for managing orders.

## **BUILDING BUSINESS RULES**

A critical requirement for event-driven applications is the ability to build business rules (policies and practices) and to use these to best handle one or more events. These rules exist in many forms in a business, frequently within application silos, less frequently in some non-actionable electronic form, such as a Word document. But because event-driven applications span application silos and must produce an automated response to the event stimulus, it is necessary to encode the criteria and response in actionable form. It is frequently assumed that such rules would be indicated or encoded while building the flow control described in the previous paragraph, and many vendors make this assumption. However, it is more preferable to encode business policies separately from orchestrations, because business policies are more broadly reusable and change more frequently. Simply stated, business rules define decisions and orchestrations put them into effect. The nature of a decision (for example, what constitutes an urgent inventory/order situation) typically changes more frequently than does the response (notify personnel and/or place an order for the item). For this reason, Sybase offers another “higher-order” service for defining business rules. In Workspace, this service is rendered much like the others: easily created with wizards, easily implemented with visual tools, and easily consumed within an event-driven orchestration. Rule services definitions are managed separately from orchestration metadata, and are executed at runtime on a separate, high-performance rules engine.

## **USING MONITORING SERVICES**

Because event-driven applications have insight into events and data that span systems as they occur, they offer a critical point for monitoring the current state of particular aspects of a business that can't be confined with a specific application silo. Examples include monitoring the performance of an orchestration, tracking the status of Key Performance Indicators (KPIs) over a sliding time window, and tracking changes in a status or state of particular silo-spanning business objects. Workspace supports this aspect of event-driven applications with another “higher-order” service: a monitoring service.

Monitor services are defined graphically and then consumed within orchestrations just as other services are. At runtime, the current state of monitored objects and various aggregated calculations on those objects can be examined in a spreadsheet view or with Web-based dashboards. To make monitored objects actionable, exceeding thresholds can easily trigger proactive alerts that send messages to humans or to other automated orchestrations.

## **CONCLUSION**

Event-driven applications can eliminate business latency and help key decision-makers in an enterprise be more responsive to a changing climate, giving them the information edge they need to stay competitive. Sybase products capture business events from any database and deliver those events to any JMS-compliant messaging infrastructure. In addition, Workspace provides full support for modeling and wizard-driven implementation of these solutions.

