Fiorano
Enabling change at the speed of thought

www.fiorano.com

AMERICA'S
Fiorano Software, Inc.
718 University Avenue Suite 212, Los Gatos,
CA 95032 USA
Tel: +1 408 354 3210
Fax: +1 408 354 0846
Toll-Free: +1 800 663 3621
Email: info@fiorano.com

EMEA
Fiorano Software Ltd.
3000 Hillswood Drive Hillswood Business Park Chertsey Surrey
KT16 0RS UK
Tel: +44 (0) 1932 895005
Fax: +44 (0) 1932 325413
Email: info_uk@fiorano.com

APAC
Fiorano Software Pte. Ltd.
Level 42, Suntec Tower Three 8
Temasek Boulevard 038988
Singapore
Tel: +65 68292234
Fax: +65 68292235
Email: info_asiapac@fiorano.com

Eight core infrastructure requirements for Event-Driven SOA

Driving the reuse of IT assets & Real time information exchange between applications

--- Atul Saini

EIGHT CORE INFRASTRUCTURE REQUIREMENTS FOR EVENT-DRIVEN SOA

Driving the reuse of IT assets & Real time information exchange between applications

Executive Summary

The emergence Service Oriented Architecture has enabled enterprises to unlock application and data silos spread across enterprise, driving the reuse of IT assets and real time information exchange between applications. But the question remains: what are the key requirements for infrastructure to meet the growing business demands of 21st Century enterprises?

Ideally, opening locked systems and allowing real time data exchange is just the starting point to launch an Agile IT infrastructure to enable companies to dynamically change and adapt to new business needs. Together with the data exchange capabilities of Services in an SOA, a key requirement is to enable these services as loosely-coupled agents working independently so they can be updated and swapped as and when required. To achieve this level of agility the underlying SOA fabric or platform - the Enterprise Service Bus (ESB) - should have features that allow Event Driven (agent-like, data flow) behavior in the deployed services.

There are currently many products on the market claiming to implement a distributed Event-driven Service Oriented Architecture (SOA), purporting the benefits of loosely coupled SOA and decoupled Event Driven Architecture (EDA). The eight sections below highlight key requirements of an infrastructure Platform (typically an ESB) to enable an effective, extensible, distributed Event-driven SOA

1. Distributed State Maintenance for Efficiency

Distributing state to prevent centralized performance bottlenecks.

Commercial grade event-driven systems are often complex, requiring events to be routed across multiple applications across a network, often in parallel. If state is maintained in a single location, the central state store needs to be updated often, causing unacceptable performance bottlenecks. By distributing the state of the application across the Service Components running at the end-points of the network, without the requirement for messages to traverse any central hub, the underlying platform can enable dramatic increases in efficiency while obviating the need for a central state store.

2. Enabling direct event-flows between distributed Services

The Ability to create direct event-flows in a distributed environment by automatically creating underlying middleware structures between distributed services.

All integrations require events to be routed in some order between different applications running on distinct machines in a heterogeneous network. Such integrations are surprisingly hard to set up within existing "process driven" Integration suites, since such suites provide no in-built tools to automatically configure the flow of events between physically executing application processes without knowledge of underlying middleware structures; rather, all event-flows have to be manually configured, typically using middleware such as JMS, MQSeries, etc., increasing the time and complexity of the implementation. For instance, in a typical scenario, "topic-names" need to be configured for all event flows, and name-clashes need to be manually detected and developers have to understand low-level messaging concepts such as "durable subscriptions, persistent and non-persistent messages", etc. By creating such underlying middleware structures automatically, without requiring users to understand any middleware concepts, the infrastructure can dramatically simplify the implementation of an SOA.

3. Deploying event-flows across multiple component-models

The ability to define Event-flows independent of the component-models and platforms i.e. an ability to easily define 'ad-hoc' workflows and distributed applications across multiple component-models and platforms deployed within an organization.

An event-driven SOA will typically be implemented at the enterprise level across a heterogeneous technological application portfolio. These heterogeneous applications use a variety of component models. Most current integration suites are limited in their support for multiple component-models and platforms; they are usually biased towards development using particular component models such as EJB, J2EE and

COM/.NET for Microsoft-based solutions. As such, these platforms are unable to easily compose distributed integrations across multiple component models already deployed within an enterprise. Modern integrations require support for multiple component models for quick integration, with added support required for legacy applications and protocols.

4. Ability to handle network-failures in a distributed environment

An ability to handle network-failures within components across a distributed environment
In a distributed Event-driven SOA where services/components run independently, it is important for network failures not to disrupt running business processes as far as possible. Hence, comprehensive support is required from the underlying ESB Platform to handle network failures. Since most integration suites have a centralized hub-and-spoke design, any network failure results in end-point applications disconnecting from the broker, requiring fault-tolerance to be pre-programmed into the Services. A good Event-driven SOA requires a distributed infrastructure model to ensure there is no single point of failure, unlike classical, centralized hub-and-spoke model.

5. Ability to easily configure Services participating in a distributed business process

An ability to easily configure/create and modify services for quick integration of existing applications or deployment of new business processes
To achieve agility in their IT portfolio, enterprises need integration infrastructure that can quickly adapt to changing demands. This cannot be done if every integration process has to go through the traditional application development cycle. Today enterprises need **configuration** instead of **coding** to achieve such agility. Most integration platforms do not provide adequate support to easily configure services and applications (running at the network end-points) that actually participate in the integration workflow. In the typical case, the configuration of each adapter and distributed application needs to be manually updated within a centralized repository, from where it is loaded to compose the integration workflow, leading to increased programming time and maintenance complexity.

6. Ability to debug the flow of Events across multiple distributed applications

An ability to debug the flow of events/data across multiple distributed applications without stopping applications and business processes in production
Most real-world integrations involve the exchange of events and data between applications distributed across a network. During the course of an implementation, it becomes necessary to debug events and data flows between distributed applications. Current integration platforms, mostly based on centralized process engines, provide little or no support for debugging event-flows across applications running at the end-points of the network. Such support is critical for the deployment of event-driven SOA solutions.

7. Ability to provide equal citizen status to multiple programming languages (C, C++, VB, Perl, and others)

Most SOA platforms are biased towards development in one or at most a few languages (typically Java, C and C++). Modern business processes in a heterogeneous environment typically require integrations spanning multiple platforms, across applications written a variety of programming languages including Java, C, C++, C#, Visual Basic, Perl, and a variety of other scripting languages.

Even the more 'modern' SOA platforms are typically biased towards a single language adapter development base, with wrappers for additional language support. For instance, several platforms support only native Java adapters; C adapters are invoked using JNI which is known to have several problems and limitations. As such, lack of support for multiple programming languages can be a barrier to implementation.

8. Ability to easily map changes to abstract business process flows to implementation-level event-flows between distributed services

Quickly map changes in Business Process Flows to actual implementation-level data-flows between applications
Traditional SOA platforms are based on the notion of 'control flows' between abstract processes typically rendered using graphical process-design tools. However, the logical flow-diagram of the business

process within the process-designer tool is completely disconnected from the physical event-flow of the overall business process as it executes across the network, since a single 'activity' within the logical business process may require the flow of events between multiple application instances, across different physical machines. As a result, what one sees logically on the process-management screen is not what happens physically at execution time. The ability to easily map process-level changes to implementation level flows using a coarse-grained Services model is therefore important for a flexible event-driven SOA platform.

So what's the solution?

The problems discussed above are inherent in the design of most current SOA platforms and cannot be easily remedied without a significant rework of the basic architecture of such systems. Most platforms on the market today map well to synchronous request/reply interactions but do not map well to the implementation of other critical integration patterns including database consistency and multi-step processes, which require the flow of events between independently executing processes.

A solution to the problems of current platforms requires,

- An infrastructure platform that supports the intelligent routing of information, including both events and requests, across distributed service components over a network, and
- A framework (including APIs, tools and a coarse-grained component model) for creating business applications as a collection of modular service components

The first of these requirements is addressed by a distributed Enterprise Service Bus (ESB) architecture that allows service-components to run at the end-points of a network and communicate in a peer-to-peer fashion with support for store-and-forward messaging, without traversing a central broker.

The second requirement is addressed by Business Component Architecture (BCA). BCA unifies the notions of request/reply and event-flow (EDA) and turns the focus of development away from the notion of distributed computing (i.e. distributed request/reply and MOM) towards the creation of software modules called Business Components that follow the semantics of business functions. BCA tools allow the creation of request/reply and event-driven interactions between distributed service components, allowing processes to be created, deployed and modified dynamically in response to business requirements.

The Fiorano SOA platform is based on a distributed, event-driven ESB architecture, which implements the concepts of BCA. Applications deployed on the Fiorano platform overcome the seven problems outlined in this paper.

About Fiorano Software

Fiorano Software (www.fiorano.com) is a leading provider of enterprise class business process integration and messaging infrastructure technology. Fiorano's network-centric solutions set a new paradigm in ROI, performance, interoperability and scalability. Global leaders including Fortune 500 companies such as Boeing, British Telecom, Credit Agricole Titres, Lockheed Martin, NASA, POSCO, Qwest Communications, Schlumberger and Vodafone among others have used Fiorano technology to deploy their enterprise nervous systems.