zapthink
white paper

PUTTING THE CONTROL OF BUSINESS PROCESSES INTO THE BUSINESS USER'S HANDS

PUTTING THE CONTROL OF BUSINESS PROCESSES INTO THE BUSINESS USER'S HANDS

October 2004

Analyst: Ronald Schmelzer

Abstract

In business, the only constant is change. Businesses, like people, are continuously evolving and as such face rapid and continual change. As markets and customer needs evolve, enterprises must respond with new ways to attract and retain customers and partners, increase operational efficiency, and achieve greater visibility into their business processes.

In most businesses, however, business people control the processes, while IT people control the systems. IT staff see business processes through the lens of the low-level parts of the flow, rather than at the business level. As a result, they aren't capable of implementing the processes so that they will meet continuously changing business requirements, thus impeding business agility. Business users are increasingly demanding that they have control over their own business processes – and so, are requiring systems that put control of the flow and logic into their hands, not those of IT.

Fiorano Software offers a business-driven approach to building application functionality. They tie together the notions of Service-oriented process, Service-oriented integration and event-driven, message-based interaction into a single environment that enables users to combine their assets and information from multiple points of view. Fiorano's event-based, business process-driven, Service-oriented integration approach solves today's business problems by direclty mapping the model of a business process to the underlying implementation, removing the disconnect between business and IT that companies face today.

I. Giving Business Users Control over their Processes

Ever since the development of Information Technology (IT), companies have felt that control of their business processes and logic has been in the hands of their IT organization. Many times, they have felt powerless as new business requirements emerge, evolve, and mature, while their IT systems are incapable of dealing with these changes and instead are programmed to deal with yesterday's inefficient and ineffective business logic. Clearly, in order to deal with the pressures of today's dynamic business climate, companies must put the control of business logic into the hands of the owners of the business processes: the business users.

In business, the only constant is change. Businesses, like people, are continuously evolving and as such face rapid and continual change. As markets and customer needs evolve, enterprises must respond with new ways to attract and retain customers and partners, increase operational efficiency, and achieve greater visibility into their business processes. At the same time, they must release new products or introduce new services, deal with competitive threats, and account for major corporate structure changes such as mergers and acquisitions.

In order to meet these continuous market pressures and challenges, businesses must be agile. *Business agility* is the ability of a company to respond quickly and efficiently to change, and to leverage change for competitive advantage. However, the ability for businesses to make and respond to change is hampered in many instances by their IT systems. Information technology is often the area most relevant to discussions of business agility, because achieving agility begins with removing the bottlenecks that impede it, and IT is usually where the bottlenecks are. In fact, companies are so used to the fact that IT creates a bottleneck within their organization that technology and its limitations often drive business decisions.

Part of the reason that IT poses an agility barrier is that companies are composed of a wide range of technologies implemented with disparate technologies and distributed, disconnected systems. In order for companies to meet their continuously changing requirements, they must integrate and connect these various systems and data sources – much of it locked in isolated, closed, and proprietary systems. This need for *integration* is spurred by the desire to link these systems together in order to provide a cohesive view and access to information that can power decision-making, customer interaction, and the delivery of products and services.

Counterbalancing the urgent desire to integrate systems is the fact that most of enterprises harbor a hodgepodge of heterogeneous systems and architectures that resulted from years of purchasing and implementing legacy systems, hardware platforms, operating systems, database storage technologies, network protocols, component object models, middleware platforms, programming languages, and file formats. In addition, executives in charge of IT implementation and purchasing decisions are wondering how to make their heterogeneous IT environments more flexible and responsive to changing business requirements, reduce the complexity of their infrastructure, get more value out of existing systems, avoid getting locked into purchasing technology from a single vendor, and reduce unnecessary implementation time and cost, all without breaking the bank.

Clearly, in order to successfully address the needs of business agility, companies must deal with integration in a heterogeneous IT environment. However, successfully solving integration challenges by itself does not provide the answer to all of business's agility concerns.

Putting Application Development into the Hands of Business Users

Integration by itself is not the problem – the real problem is that business people can't manipulate IT resources directly such that when they make changes to the process, the IT resources are affected, and vice-versa. When changes occur in the IT environment, the business processes should be adjusted to account for those changes. As a result, we need a direct connection between business control of processes and IT control of resources so that business logic remains in the hands of business users.

However, accomplishing this goal can be technically challenging. Traditionally there's been a disconnect between business users and the applications they use, because there's always insufficient interpretation and long implementation time by IT application developers, whose role it has traditionally been to interpret business requirements into IT implementations. Also, business logic is a concept that at some point has to be implemented in the underlying systems. Business logic resides in databases, application servers, desktop applications, presentation layer clients, Web pages, middleware, portals, network devices, and everywhere in between. And there's the contradiction: how can business logic be *business* logic if it's locked away in the technology, rather than in the hands of the business?

Yet, coding business logic into application and systems is the only way to realize business requirements in IT, and businesses have been doing so for decades. Over time, companies have sought to encapsulate business logic in increasingly abstract constructs to improve the maintainability and flexibility of the code. And while there have been modest flexibility improvements since the days when all application functionality resided on the same system, the unfortunate truth is that these advances have been little more than a business logic shell game, moving the hard-coded logic from one system to another. Instead of concentrating functionality in a single location so that it can be easily managed and manipulated, businesses create instant legacy code all over their infrastructure.

II. New Architectural Approaches for Business Agility that Shift the Control of Power

Clearly, today's IT executives require fresh approaches to dealing with heterogeneous environments and the increasing pace of change, in the face of tight budgets and a tough economy. They need more than just a new set of integration technologies, but also a fundamental change in the way companies architect systems and processes to handle integration requirements.

In order to bring IT into the 21st century and give the business users the control of IT resources that they desire, IT organizations must reconceive their systems in the light of emerging architectural and technological approaches that are changing the landscape of corporate systems. Namely, there are three key trends that companies must embrace to achieve the vision of business logic controlled by business users:

- *Systems must interconnect in a loosely coupled fashion*
- *Business users should define business processes at runtime, using metadata to control and compose underlying systems*
- *Interactions between systems should be event-driven*

Loose Coupling and Coarse Granularity: Keys to Composability of Business Logic

There are a variety of reasons why issues of integration, business logic, and application reuse continue to plague enterprises, even though many people have proposed a number of technological approaches over the past few decades. On one level, the cause is the lack of standard ways of programming different systems to communicate. For any two different systems, the traditional approach to integration is to write programming code for each system that "teaches" it how to talk to the other system. Such an approach is expensive and time consuming, and doesn't scale well or respond to change in a flexible way. This approach to integration is also *tightly coupled*, which means that one programming team must control the integration code on both systems to get them to communicate with each other. In addition, such integration is point-to-point, which means that the complexity of maintaining the connections dramatically increases as the number of connected systems goes up.

Yet, tight coupling is only one reason why integration is still troubling companies today. Another key reason is that the level of abstraction at which business analysts and other people think is different from the level of abstraction at which technical people think. Business people think of connections between business systems at a higher level, while technical implementors think of low-level components at a"fine level of granularity that is most concerned with individual operations, connections to data sources, or exchanging messages. This disconnect between the higher-level, or coarse functionality that is business people understand, and the fine-grained functionality that technical people work with causes problems with integration, since implementors have to break down or decompose the biggerbusiness-oriented blocks to the lower-level requirements of their IT systems.

Companies thus require a *loosely coupled* approach to integration—one that does not require control of systems nor an intimate connection between the requester of information and the provider. Furthermore, the implementation of widely-established standards such as XML-based Web Services should guide these loosely coupled interactions . However, standards-based loose coupling alone is not sufficient to get enterprises out of their integration rut. Rather, organizations need the right architectural underpinnings to be able to translate high-level, coarse-grained business logic building blocks into lower-level IT implementations.

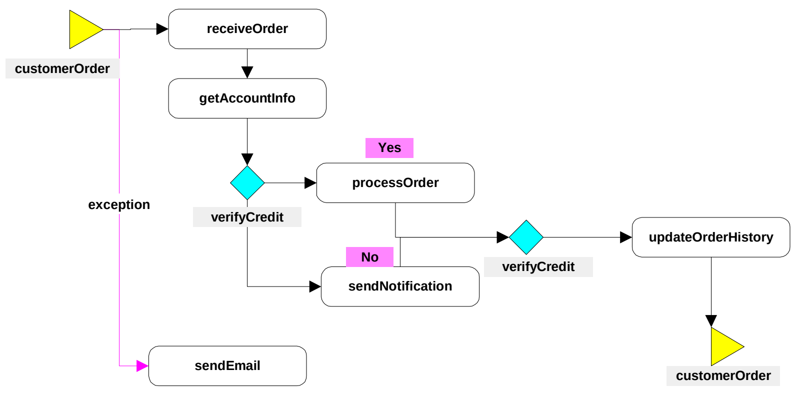
Business Logic Defined by Coarse-Grained Business Process, Controlled by Metadata

In order to connect the seemingly disparate worlds of business and IT, a common understanding of the world is needed by both parties. In this case, both business and IT need to have a common understanding of the business process and its requirements in order to translate those requirements into implementation, without any loss of agility, quality, or ambiguity. As a result, companies need a notion of a single view of a business process that provides a means for business users to define the process in a way that changes are automatically reflected in the IT implementation, and vice-versa. As business logic changes, the flow of information in the underlying IT systems should be changed automatically and dynamically.

Business process, defined as the logical ordering of activities a business performs to meet a particular business goal, is a fundamental part of business, in much the same way that architecture is a fundamental part of IT infrastructure. Process not only helps to instantiate business requirements, but it helps to facilitate integration. Well-defined process combined with an effective means to manage and execute those process definitions accomplishes the basic tasks of integration. Furthermore, in order to implement process in a Service-oriented manner, process definition and execution must exist separately from the Services that comprise those processes. In essence, IT capabilities, or Services, must be developed devoid of process in order that they can participate in an SOA that meets the goals of business agility.

The first step to implementing business process in an enterprise is to create a representation of a business process that can then be executed by human and/or machine-based systems. Business managers and analysts tend to think visually. When they describe a process, they often reach for a pad of paper or marker to sketch out their ideas. To higher-level managers and line-of-business users, a process is something that can be written down and visualized in a way that can be communicated to others in the organization, as shown in the figure below.

Figure 1: Business Process Visualization



However, historically there has been little way to connect these meaningful diagrams into true composite applications. Since business-oriented users will rarely be successfully converted into programmers, these users are expecting the very same visual representations to be the primary way in which they can build composite applications that will implement their process flow desires.

In general, we can glean a number of important points as they apply specifically to business processes:

- *A process is a collection of activities* –The concept of *activity* represents the lowest level of decomposition of the business process. An activity represents a unit of work performed by a user, system or partner. An activity may have some input and output and some associated actions. A process is not a singular activity or action, but a collection of activities and actions that when tied together in a logical flow result in an outcome. A business process is merely considered a succession of activities following a specific control flow.

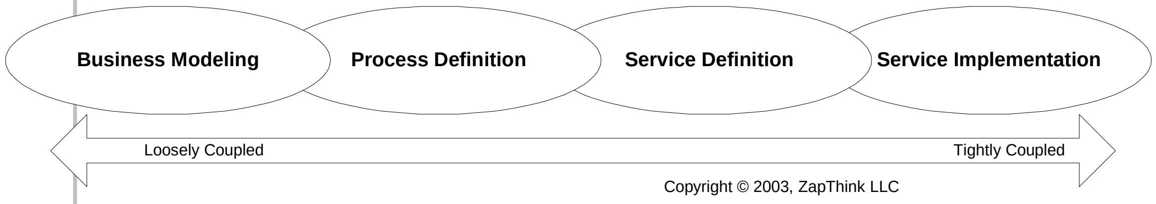
- Business activities can be carried out by wide range of *participants* -- individuals, organizational roles, or supporting IT business systems.
- *Processes may contain other processes* – Since a process is a collection of activities, any process may be represented as a type of activity that can be consumed by other processes. In the case where a process is consumed by other processes, it is called a *subprocess*.
- *Processes may occur within, outside, or between enterprises* – Business processes define the way in which work is done both within and between organizations.
- *Processes have outcomes* – Processes inherently impact the operation of a business at various levels in the organization. Processes have defined conditions triggering their initiation and defined or expected outputs.
- *Processes are an inherent part of the enterprise* – An enterprise cannot function without processes, whether formally defined or occurring as a result of normal business operation.
- *Processes may consist of automated activities and/or manual ones* – A business process is comprised of activities that may either be performed by a person or a proxy for that person, namely an automated system.

A particularly potent way of viewing the IT ecosystem is by separating the various business needs into the following four layers of infrastructure abstraction:

- Business Requirements generation at the Business Modeling layer
- Process definition and execution at the Process Definition layer
- Individual task and activity definition at the Service Definition layer
- Service execution and application logic invocation at the Service Implementation layer

These layers can be seen in Figure 2 below.

Figure 2: Layers of abstraction that support Business Agility



The movement to various levels of abstraction touches upon an industry holy grail: application functionality reuse. A primary focus of IT architectures is the ability to reuse functionality. Applications are more maintainable because the business process is not hard-coded but instead described by separate process definition layer that can be changed at runtime without requiring change to the underlying Services. This abstraction enables flexibility and responsiveness to change at the most frequently changing architecture layer, that which implements business processes and procedures.

Moving process from the Service functionality into a separate process layer considerably facilitates business agility for the following reasons:

- A change in the process definition does not require a modification of underlying application functionality. Processes can be rapidly changed and redeployed as business requirements demand with minimal, if any, programming. Companies won't have to constantly rip into the technical plumbing each time they want to change or implement new business processes.
- Businesses can easily monitor, audit, and escalate critical business processes since they have greater visibility into the overall process, rather than having the processes hidden by discrete application functionality.
- Processes can be composed of sub-processes, thus delegating business rules authority to different parts of the organization or to external organizations.
- Processes can be implemented using any Service that fulfills the basic requirements of that process flow, allowing for variable implementations depending on the requirements of a user.

However, if we want to successfully extract the process-oriented logic from the applications, we must first agree on how to abstractly represent process flows, and then how to automate and execute those flows. In addition, there are significant layers of complexity involved when we aren't talking about individual, atomic pieces of functionality, but rather the interactions of entire systems of resources. Furthermore, enterprises need technologies, products, and solutions that have been developed with process in mind, rather than just application functionality.

As application logic and functionality becomes more coarse grained, they become more "business-like" and less "API-like." The coarsest of all services might be the "do business with my company" service. At this level, all of the details of how the company operates (its processes and services) are hidden from the service consumer. Fundamentally, if the runtime control of business logic by the business user can be achieved through the means of coarse-grained business logic, defined through business processes as abstracted away from specific systems into metadata, it becomes possible to achieve the goal of "zero impedance" between business and IT.

Last Part of the Puzzle: Connecting Processes and Services with Events

While the goal of "zero impedance" between business and IT sounds good in theory, in practice achieving loose coupling and coarse granularity is challenging. One approach to making loose coupling work in practice is to utilize the notion of loosely coupled communications – or specifically, event-driven, asynchronous modes of communication.

Communications between distributed systems fall into two basic categories: synchronous and asynchronous. Synchronous communications consist of round-trip messages in which the sender waits for a reply. Submitting a Web page form and waiting for a confirmation page is a familiar example of a synchronous operation. In contrast, with an asynchronous message, the sender can submit a request, and then go about its work. If a reply does come, then the original sender can pick it up when it wants. Email works asynchronously, for example.

Most importantly, the notions of asynchrony and coarse granularity are actually very closely related. Since coarse-granularity of business logic drives overall business agility and productivity, the flow of the business logic must match that of the business process. Since business process rarely occur in a completely controlled, synchronous fashion, the underlying systems must be able to handle the "real-world" that is event-driven and asynchronous.

Asynchrony is also necessary for the proper operation of an SOA because it is essential to the ability to perform long-running transactions. Typical transactions -- say, transferring money from checking to savings -- must take place very quickly and reliably, because the various systems involved must wait to make sure the transaction was successful before they go about their business. Such ordinary transactions are therefore constrained by the fact that they must be synchronous. However, a transaction might have multiple steps, for example, company A submitting a purchase order to company B, who fulfills the order and then submits the invoice to company A, who pays it. Such a transaction might take weeks, and thus must be handled asynchronously.

One of the best ways to implement a loosely coupled, asynchronous infrastructure is to utilize *event-driven* approaches. Events are simply asynchronous messages that are sent between independent software components that are completely unaware of each other -- in other words, decoupled, autonomous objects. Events can be two-way interactions between systems or one-way notifications. An event source typically sends messages through some middleware integration mechanism like a bus, and then the middleware publishes the messages to the objects that have subscribed to the events. The event itself encapsulates an activity, and is a complete description of a specific action. Since events facilitate loosely coupled modes of interaction, SOAs can fundamentally be event-driven, then, to realize the primary benefits of business agility.

However, traditional event-driven systems, especially those popularized by publish/subscribe approaches, suffer from a fatal flaw: they can never be used to realize the goal of business users controlling their own logic using coarse-grained Services, since they require too much intimate knowledge of how the underlying systems work in order to be of value. Traditional event-driven systems require users to know about topics and channels as well as the specifics of how systems interconnect and their protocols. In order to solve the requirements outlined above, the loosely-coupled, coarse-grained system of the future must *isolate* the business user from having to know about these details and automatically define, configure, and manage the event-streams on behalf of the business user.

As a result, what many companies need is an autonomous runtime infrastructure that provides a mechanism for definition of coarse-grained business logic, the loose coupling of the Services that implement the business logic, a means to build new processes from existing defined Services, and an automatic means for interconnecting those Services to enable the business processes in a way that doesn't sacrifice agility.

III. Tying it all Together with Architecture: Service Orientation

Fortunately, there is an architectural approach and a new class of products that implement them to provide reality to the goals of business users controlling their own business logic in an agile fashion. This approach is known as *Service Orientation*.

Service Orientation, and its application to architecture, *Service-Oriented Architecture* (SOA), is an approach to designing distributed computing infrastructures that considers software resources as services available on a network. Producers of these services must be able to publish information about them in a service registry or repository, where service consumers can then look up the services they need and retrieve the information about those services they need to bind to them.

Rather than thinking about how to get information into or out of different systems, we can think about how to expose system functionality to whatever system cares to access it. In this way, we release ourselves from thinking of information in a point-to-point fashion and instead think of information as freely available on a network of Services. As a result, we can reuse existing functionality in different ways – for building new, composite applications, or extending existing functionality. In order for this new class of integration solution to escape the problems of previous attempts, the solution must contain the following characteristics:

- *Loosely coupled* – In a loosely coupled system, application consumers don't need to have knowledge beforehand about a given piece of system functionality, other than where to find it. Application functionality and the programs that invoke them can be changed independently of each other, instead of requiring a redesign of the involved components.
- *Coarse-grained* – Rather than interacting with a large set of detailed, fine-grained APIs, users can interact with systems through *coarse-grained,* business-level interfaces that roll up the functions of many different API calls into a small number of business-oriented messages. Often, the first step is simply to create fine-grained, low-level services and then bring them together as higher-level Services to produce a set of coarse-grained, business-level capabilities that didn't exist before. It is important to mention that underlying IT infrastructure needs to be able to support and model coarse-grained Services in a way that is natural for the business user. Users should be able to compose fine-grained IT-focused Services into more coarse-grained business Services without having to be technical experts.
- *Standards-based* – Instead of utilizing proprietary or closed APIs that require users to learn the intricacies of a particular vendor's platform, integration tools that successfully address the challenges of business agility will be standards-based, lowering the cost of integration and allowing the widest possible range of developers.

Coarse-Grained Services

The key to realizing the benefits mentioned in this paper is the simplicity by which coarse-grained business Services can be created. SOAs that implement a model for coarse-grained service manipulation and process meet these goals by helping companies get more value out of existing resources, componentizing legacy applications, enabling system migration, application reuse, and consolidation of application functionality. Fundamentally, SOAs that leverage standards-based, event-driven, loosely-coupled, coarse-grained Services have the flexibility and responsiveness to enable business priorities to finally drive technology decisions.

The practice of SOA is an evolution of distributed computing and the practice of enterprise architecture, which is itself an aggregation of all the individual IT systems within an organization. The potential rewards for enterprises that understand this evolution and make the move to such architectures are enormous. SOAs promise to address the complexity, inflexibility, and brittleness issues of existing approaches to integration, while embracing heterogeneity in the IT shop. Finally, distributed computing technology promises to be flexible and nimble enough to respond to business needs and provide the business agility that companies have craved for so long, but which has always been out of reach.

Implementing Event-Driven Service-Oriented Integration: The Enterprise Service Bus

While the concept of event-driven SOA seems good in theory, how can companies implement this vision given their current infrastructure? Most companies today are implementing distributed computing systems on top of application servers to meet their non-SOA computing requirements. At the same time, companies have significant investments in other means of connecting their disparate systems, ranging from message bus technology to integration broker middleware. How can companies tie together these disparate approaches into a cohesive solution for event-driven SOA?

One growing approach to meet these emerging requirements is the *Enterprise Service Bus (ESB)*. While the definition of an ESB is still in flux, one increasingly well-accepted definition is the combination of standards-based messaging middleware, distributed Service containers that use standards-based, coarse-grained Services, XML transformation, and rules-based routing of documents. ESBs also often provide some value-added services beyond those found in those previous middleware approaches, such as message validation, transformation, content-based routing, security, Service discovery, load balancing, failover and logging.

As a result, ESBs can provide the backbone over which loosely-coupled, asynchronous interactions can happen. In addition, ESBs represent a viable implementation approach for SOAs that relegate the application server to a role that is one of executing particular synchronous Services, while the ESB forms the backbone for asynchronous inter-Service communication.

While there are several implementations of ESBs, they differ in how they implement and approach the notion of a Service. The Service abstraction model chosen by an ESB implementation goes a long way in determining its ability to model and implement real-world business processes. Hitherto, most traditional software infrastructure platforms, such as messaging servers, application servers and integration servers, have modeled services as relatively fine-grained objects – i.e. as objects that IT developers can understood and easily manipulate, but exist at too low of a level for business people to understand directly. As such, using a fine-grained object model within an ESB results in implementation problems since higher level business processes have to be implemented as a collection of finer-grained Services that business users don't understand, and whose complexity is the root of today's integration problems.

In addition, today's development approaches and methodologies are inappropriate for handling the business agility requirements described above. Structured programming languages limit reusability to blocks of procedural code, and component models such as COM and EJB failed to live up to the requirements of coarse grained business level Services because such component models were defined to handle programming level services.

As a result, an effective ESB must be able to provide a mechanism by which business users can manipulate live business-level components to compose and deploy event-driven business processes in response to real-time business changes, and IT development staff can create coarse-grained components that are directly manipulated by Business users. In these systems, the business user is in reality manipulating the implementation-level objects themselves.

What's important to note is that all of the above key technology trends are converging to help realize the goal of business agility and remove the disconnect between business and IT: the event-driven, message-oriented, loosely coupled approach of ESBs provide an optimal base for running loosely-coupled, coarse grained Services implemented in an SOA. Process-driven, Service-oriented solutions provide the business process guidance necessary to ensure that business users can compose fine-grained Services into real, run-time business processes. Through the combination of these approaches, companies can move toward the vision of software that integrates automatically. And the binding force that brings business users, business process, and IT capabilities together in a standards-based, loosely coupled way is the coarse-grained Services provided by an event-driven ESB platform.

IV. The Fiorano Solution

Fiorano Software, through their Fiorano ESB offering, has pioneered the concept of coarse-grained Services combined with an efficient and effective ESB. Their product is the first to market with the capabilities of providing a coarse-grained model for Services where each service is at a business process level that by business users can use as needed without having to make low-level IT implementation changes.

While there has been much attention paid to early Web Services entrants and vendors of products that enable SOAs, there have been few companies to pull together all the requisite components required to meet the needs of enterprise architects looking to build SOAs that deliver on the promise of business agility. Fiorano Software is one of the earliest vendors to do so. The company has leveraged their early visibility and dominance in JMS-compliant messaging to deliver a solution for process-driven, Service-oriented integration. In addition to a robust messaging platform, the company has recently unveiled an integration solution that combines visual, process-driven development of composite applications with an SOA-based integration core that can meet the needs of enterprise architects searching for an SOA solution.

Fiorano has built a platform for deploying integration solutions based on visually defined business processes and systems components. In Fiorano's SOI vision, the company sees their role as providing a reliable messaging framework that supports an integration platform that in turn connects systems through Service interfaces, rather than via adapters or other connection logic. On top of their implementation of an ESB, the company has built a visual tool for creating complex business processes that are composed of Services accessible to or created by the Fiorano platform.

The Fiorano product line consists of the following components:

- *Fiorano ESB* - An Enterprise Service Bus platform that implements standardized interfaces for communication, connectivity, transformation, portability, and security. What makes Fiorano's implementation unique is its coarse-grained component model combined with a linearly scalable, fault tolerant distributed architecture, which allows processes to be coordinated with transactional integrity across large numbers of applications running on heterogeneous platforms and protocols.

- *Fiorano MQ* - The ESB product is based on Fiorano's messaging infrastructure, based on the Java Messaging Server (JMS) protocol.
- *Fiorano Event Process Orchestrator* - A set of integrated tools for building event-driven business processes using coarse-grained Services created from existing IT assets and legacy functionality or custom-built code.
- *Fiorano Business Integration Suite* – The combination of Fiorano's ESB infrastructure with their process composition tool set, enabling the coordination and interaction of software assets across the extended enterprise.

Fiorano also provides some advanced capabilities for dealing with business-level events, including rich support for event debugging, and their *Peer Server* product that hosts coarse-grained business Services and provides the capability of peer-to-peer connectivity without traveling through a centralized hub.

Fiorano's primary challenge rests in their relative unfamiliarity as a name brand among customers and their size compared to their rivals. However, the company has proven to be a respectable adversary, competing head-to-head with messaging and integration platforms from firms like TIBCO, IBM, and Sonic Software. Combined with their flexible and aggressive pricing and quality products and customers, they are giving their rivals a run for their money.

V. Conclusions

Enterprise architects are increasingly demanding architectural solutions to their integration challenges that combine aspects of loosely-coupled, coarse-grained, asynchronous integration with visual process definition and runtime process management. While many companies have realized that SOA provides the architectural underpinnings for realizing the business benefits of business agility, implementing SOAs in practice requires a robust infrastructure that is both event and process-driven. Without an infrastructure that provides those capabilities, achieving loosely coupled, Service-oriented integration is difficult in practice.

To achieve the goals of business agility, businesses require that their underlying IT systems provide access to functionality through coarse-grained Services. These Services must then be able to be combined and composed by business users as they desire, without requiring them to become IT experts. While loose-coupling and event-driven infrastructure enable solutions for meeting these requirements, it is the architectural approach of Service-Oriented Architecture and the discipline of building coarse-grained Services that are treated as first-class citizens within the infrastructure that will make the goal of business agility a reality.

Fiorano Software provides the functionality that meets enterprises' business agility needs by providing an ESB that implements coarse-grained Services as real, manipulable objects that can be implemented on a wide range of systems. Fiorano's approach is to leverage their robust messaging infrastructure as a platform for building composite applications that interact with applications through Service interfaces. Fiorano's products provide a simple to use and manage infrastructure that can be utilized by data center administrators, developers, as well as line-of-business users.

Copyright, Trademark Notice, and Statement of Opinion

About ZapThink, LLC

ZapThink is an IT market intelligence firm that provides trusted advice and critical insight into XML, Web Services, and Service Orientation. We provide our target audience of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink's role is to help companies understand these IT products and services in the context of SOAs and the vision of Service Orientation. ZapThink provides market intelligence to IT vendors who offer XML and Web Services-based products to help them understand their competitive landscape and how to communicate their value proposition to their customers within the context of Service Orientation, and lay out their product roadmaps for the coming wave of Service Orientation. ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into how to assemble the available products and services into a coherent roadmap to Service Orientation. Finally, ZapThink provides demand intelligence to IT vendors and service providers who must understand the needs of IT users as they follow the roadmap to Service Orientation.

ZapThink's senior analysts are widely regarded as the "go to analysts" for XML, Web Services, and SOAs by vendors, end-users, and the press. They are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry.

ZapThink was founded in October 2000 and is headquartered in Waltham, Massachusetts. Its customers include Global 1000 firms, public sector organizations around the world, and many emerging businesses. ZapThink Analysts have years of experience in IT as well as research and analysis. Its analysts have previously been with such firms as IDC and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, CPExchange, ebXML, EIDX, and CompTIA.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how XML and Web Services impact your business or organization.

ZAPTHINK CONTACT:
ZapThink, LLC
11 Willow Street, Suite 200
Waltham, MA 02453
Phone: +1 (781) 207 0203
Fax: +1 (786) 524 3186
info@zapthink.com