

# An e-business Integration & Collaboration Platform for B2B e-Commerce

**Kumar Bhaskaran, Jen-Yao Chung, Terry Heath, Santhosh Kumaran<sup>1</sup>**

E-Commerce Group, IBM Research, Yorktown Heights, NY

**Raja Das and Prabir Nandi**

B2B Engagement, IBM Software Group, Somers, NY.

## **Abstract**

The B2B e-Commerce area is poised for tremendous growth. While this is a great opportunity for the system designers, the challenges are abundant as well. In this paper, we list the requirements on an application platform for B2B e-Commerce. We describe a framework-based approach to designing such a platform. We use design patterns to describe the frameworks on which the platform is based. These patterns illustrate how the platform supports collaborative business processes that integrate enterprise applications and trading partner systems. We present a programming model that enables the assembly of complex e-business applications on the platform.

**Keywords:** Collaborative Commerce, B2B Integration, e-Markets, Design Patterns, Application Platform

---

<sup>1</sup> Contact author. IBM Thomas J. Watson Research Center, Route 134, Yorktown Heights, New York 10598. [kumaran@us.ibm.com](mailto:kumaran@us.ibm.com). 914 945 1469.

# 1 Introduction

The wave of euphoria, exploration, and failures in e-Commerce, with the consolidation and demise of a number of e-markets, is giving way to the next wave of B2B commerce. There is a broad consensus that e-markets deliver aggregation and collaboration value for buyers as well as sellers. However, the success or failure of the e-markets hinge on how well they are aligned and integrated with the supply chain business processes of their participants.

In fact, the next phase of B2B commerce will go beyond just e-markets to embrace the notion of dynamic e-business services that assimilate e-markets (public exchanges and private e-markets) in the context of collaborative commerce driven by business processes within and between businesses. Businesses realize that the aggregation and collaboration value from e-markets can only be realized if the commerce relationship transcends simple exchange of messages and transactions between enterprises to include collaborative supply chain business processes.

In this paper, we describe the architecture, design, and implementation of an application platform for B2B e-Commerce that addresses these issues. In the ensuing sections, we outline the requirements on such a platform, describe the platform in detail, and present a programming model for composing complex B2B e-commerce applications using the platform.

## **2 Platform Requirements**

Any non-trivial BtoB eCommerce solution poses large number of architecture, design, implementation, and integration challenges. The key requirements of such a solution are:

### 1) Integrated User Experience

A multiplicity of systems with proprietary interfaces might be integrated in assembling a solution, but it is important to provide an integrated user experience, including role-based rendering and shared sessions,

### 2) Common Security Mechanism

Typically, each ISV (Independent Software Vendor) application and middleware component uses its own unique security subsystem. The platform should hide these disparate systems and enable single sign-on and global registration.

### 3) Business Process Integration

The key to a successful realization of a collaborative BtoB solution is the support for business process driven integration of legacy systems, I SV applications, users, and trading partners.

### 4) Common Messaging Infrastructure

Enterprise application integration via a message hub and standard XML (Extensible Markup Language) message sets reduce integration complexity.

### 5) Solution Management & Monitoring

This includes facilities for audit logging, tracing, and exception handling.

Additionally, there are a number of non-functional requirements as well:

#### 1) Performance and Scalability

As transaction volumes increase, it should be possible to maintain the system performance by adding more resources.

#### 2) Reliability and Availability

The platform should support 24 by 7 availability.

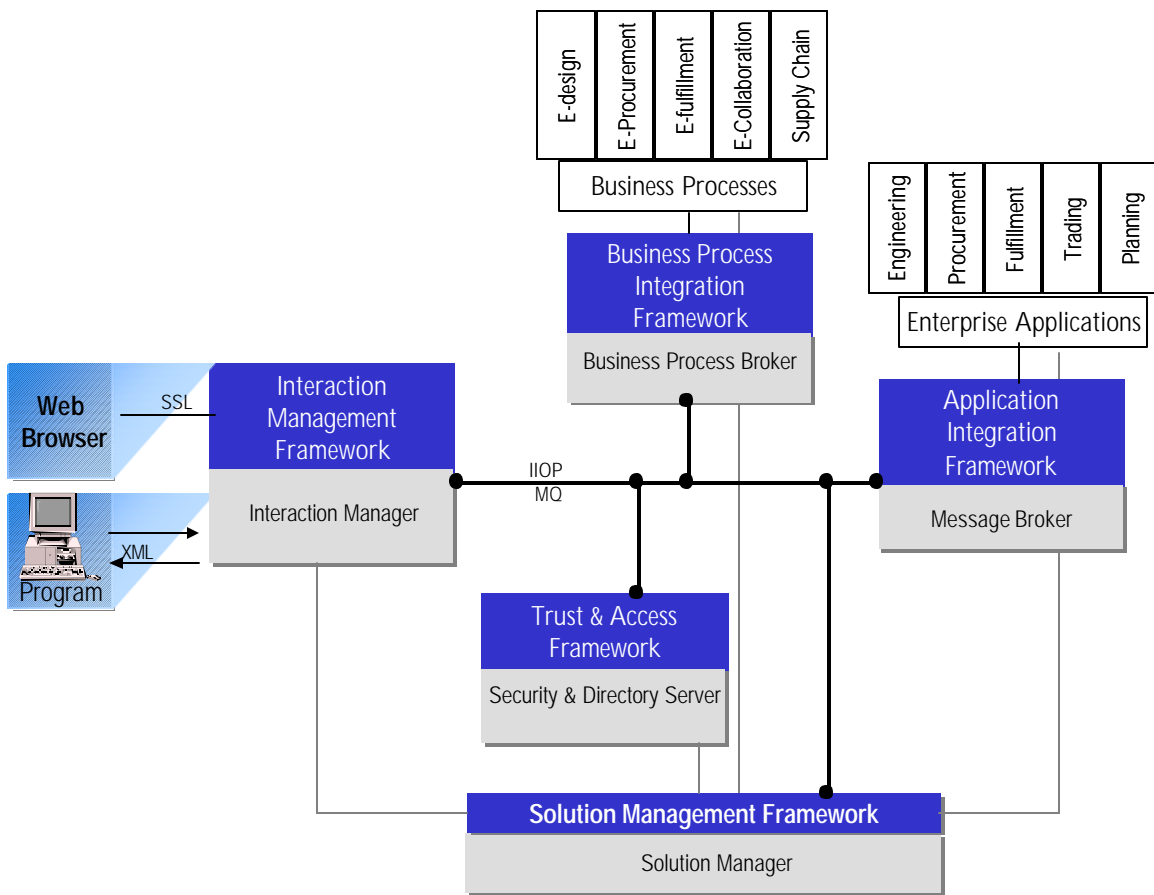
#### 3) Extensibility

The next wave of B2B e-commerce systems need to be very dynamic as enterprise boundaries fade and supply chains and virtual corporations

emerge. The platform should provide support for easily modifying and extending business processes.

### 3 Architecture

The platform is conceived as a collection of cooperating frameworks, as shown in Figure 1.



**Figure 1: Architecture Overview**

Frameworks define the protocols between collaborating components and enable the assembly of the constituent components [5]. The architecture of the platform is described here by documenting the individual frameworks as a set of patterns [6]. The patterns in turn provide a common software architectural vocabulary for understanding the design [3].

In the ensuing sections, each individual framework is described.

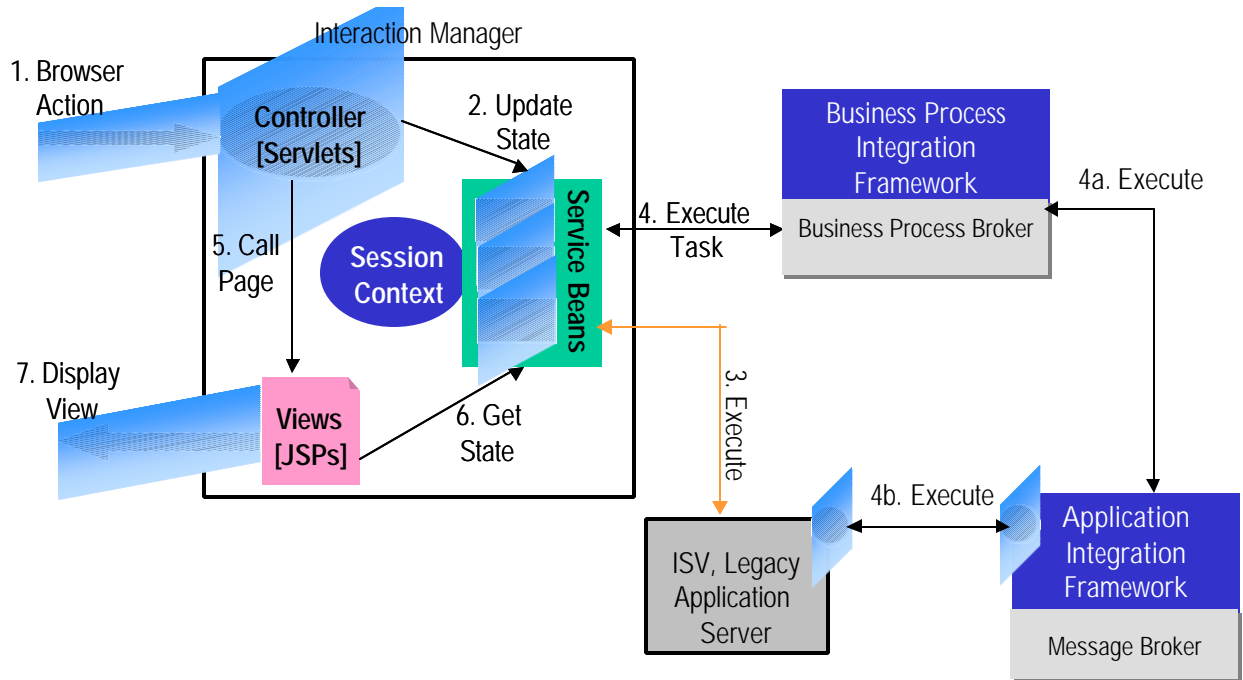
## **4 Interaction Management Framework**

The Interaction Management Framework houses solution parts responsible for driving the client interaction. This is a model-view-controller framework consisting of a set of Java Beans, JSP (Java Server Pages) templates, and Servlets. The solution assemblers create the client interaction components of their applications leveraging the Beans, JSPs, and Servlets in the framework.

The Interaction Manager can support two types of clients: the Browser clients and the Program clients. The program clients can communicate via XML over a variety of transport such as HTTPS and Message Queuing (MQ) using a number of different business protocols (e.g. OBI , RosettaNet).

## 4.1 Web Transaction Pattern

The web transaction pattern (see Figure 2) illustrates the interaction



**Figure 2: Interaction Management Framework for Browser Clients**

management framework for browser clients.

The interaction as noted before is based on the model-view-controller design pattern. The Java Beans, the JSPs, and the Servlets are associated with the session context that is initialized when the user is authenticated upon logon and the role-based desktop is rendered based on the authorization privileges of the user (see Trust & Access Framework for authentication and authorization).

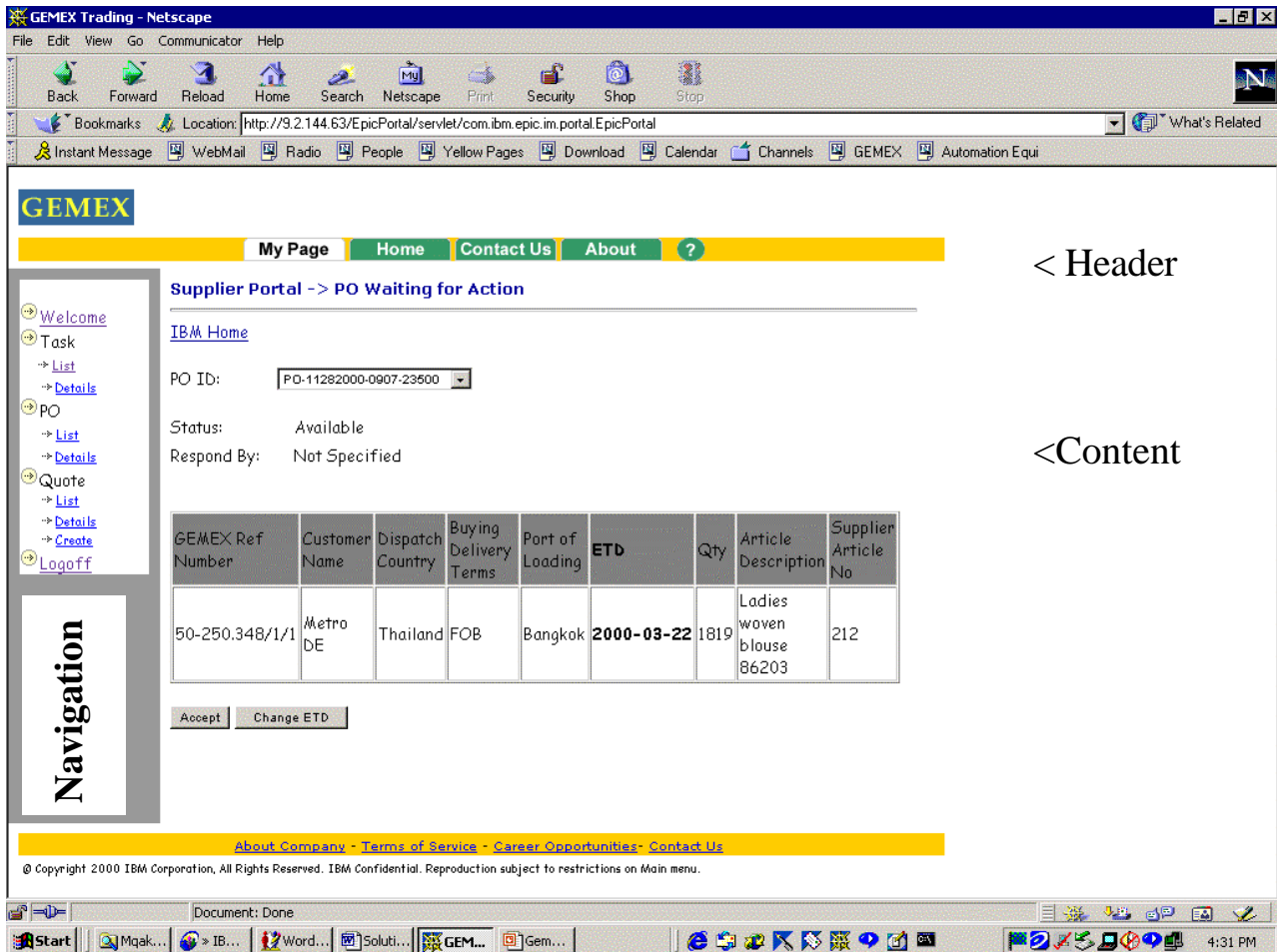
- The user triggers the web transaction via the http request using the browser (# 1).
- The Servlets in the framework serve as the view-controllers and handle all HTTP (Hypertext Transfer Protocol) requests from the client. These servlets are designed using the Template Design Pattern and customized for any specific solution by associating them with a set of plug-in service beans (Java Beans that serve as proxies to various back-end business services). The Servlets flow the web transaction to the back-end business services (#2).
- The back-end business services can be invoked using different communication modes. The service bean in the interaction manager can directly communicate to an application server (#3). Such a transaction is typical when the user is interacting directly with a particular application function (e.g. browsing a catalog in an e-commerce application).
- Alternately, the web transaction can be related to a business process task (e.g. submitting a RFQ) and the transaction triggers a state change in the business process broker (#4). The business process

- broker choreographs a variety of workflows across a number of different enterprise information systems. It publishes a message as part of the state transition to the message broker (#4a). The message broker in turn transforms the message if necessary and routes it to the appropriate application (#4b).
- o The view-controller servlet then calls JSPs to render the new views on the browser in response to the web request (#5). The JSPs get the updated content from the service beans based on the back-end activity as described above (#6) and displays the updated view (#7).

In general, a view is composed of three frames (see Figure 3):

- a. **Navigation:** Holds menu items available depending on the role of the user, for example "List work items" or "List orders".
- b. **Header:** Holds header information for the web page.
- c. **Content:** Used to render the requested business document, for example the details of a Request For Quote (RFQ) or a Purchase Order (PO) and the actions available to the user to perform on the

document, depending on the role of the user and the process context,  
 for example submit, approve, or reject a purchase order.



< Header

<Content

**Figure 3: Example of a View Displayed on a Web Browser**

The sequence of screen flows that drive the client interactions are modeled using a Deterministic Finite Automata (DFA) that is executed by the view-controller servlet. Each view node in the screen flow is associated with a

state of the DFA. The actions available in each view correspond to the transitions permissible in the corresponding node of the DFA.

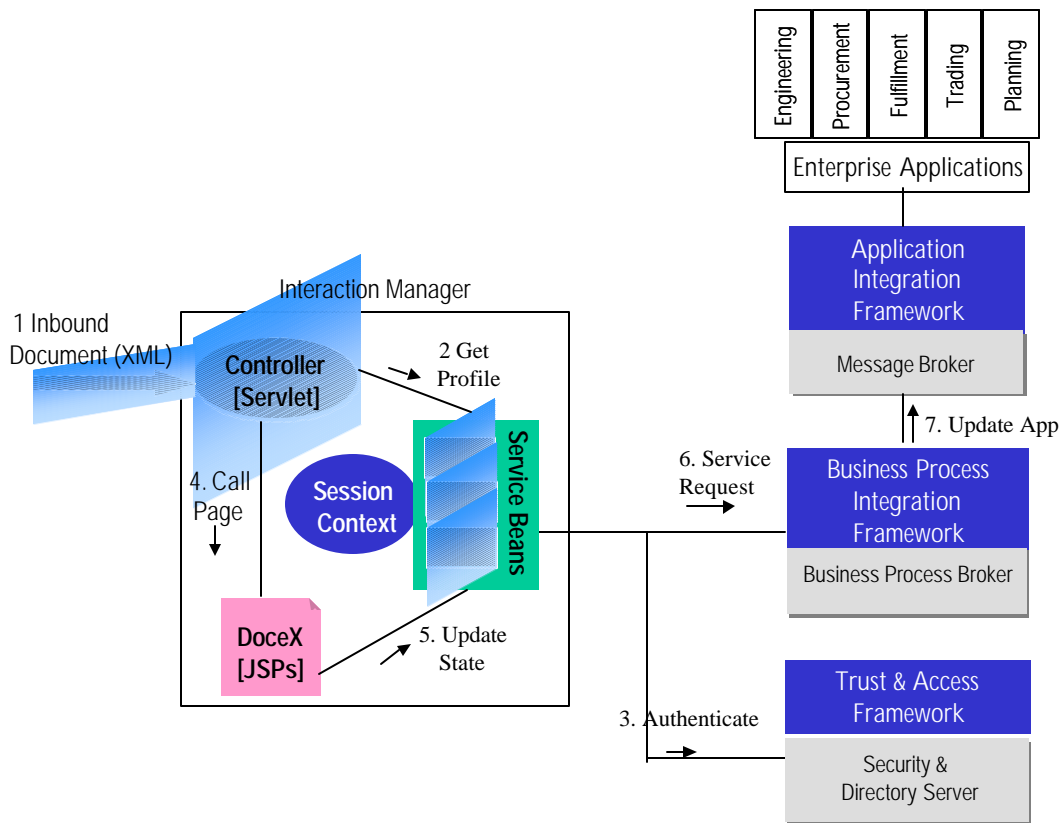
## 4.2 B2B Interchange Pattern

The B2B Interchange pattern captures the program-client interaction (Figure 1), i.e., the interaction of applications and systems across enterprise boundaries. An example is in e-procurement where systems from buyer and seller organizations interact [1].

Such interchange typically involves business documents that are captured as data in messages. These messages are then interchanged using different *transports* (e.g. Message Queuing or MQ, https, TCP/IP) after they are packed for delivery where the *delivery formats* can also be different (e.g. XML flavors, EDI). The interchange of messages are also governed by different *business protocols* (e.g. RosettaNet that is used widely in electronics industries) that define the sequence of interactions such as receiving an acknowledgment following the transmission and receipt of a message as well as other parameters such as timeouts that are part of an

overall *trading partner agreement*. Further, a *business process* governs the entire B2B interaction.

The B2B Interchange Pattern for an incoming XML message with http transport is shown in Figure 4.



**Figure 4: B2B Interchange Inbound Pattern**

- o An incoming http request with the XML message is processed by the Controller in the Interaction Manager much like the web client interaction (#1).

- The header attributes of the message are then used to authenticate the sender using the Trust & Access Framework (#2 and #3).
- Following the successful authentication of the sender and gathering the trading-partner profile, the controller calls the appropriate Document exchange (DoceX) JSP (#4).
- The DoceX JSP unwraps the message, parses and transforms it (using the XML and XSL services provided by the Interaction Manager) and calls the service bean to update state, i.e., initiate the back-end transaction (#5)
- The service bean, more specifically the business broker bean, invokes a service request and communicates the business document data that was received to the business process integration framework (#6).
- The business process broker correlates the incoming business document with a business process instance (e.g. forecast received is correlated with a collaborating forecasting process that is in progress), updates the state of the process as warranted, and then may initiate a transaction against one or more enterprise applications (#7) that is handled by the Application Integration Framework.

The B2B Interchange pattern for an outbound message is shown in Figure 5.

- A business process instance, executed by the business process integration framework, at some point in its evolution may initiate a request to interchange data with a business partner (#1).
- The message broker then fetches the trading-partner profile for the business partner(s) from the trust and access framework (#2). This profile describes how the message needs to be constructed and delivered to the business partner.
- The message broker then splits and transforms the message as required and routes it to the Interaction Manager (#3) – example is a forecast for parts that is split and communicated to various suppliers.
- The Interaction Manager upon receiving the message initializes the controller (#4).
- The Controller calls the relevant DoceX JSP to pack and deliver the outbound message (#5-7).

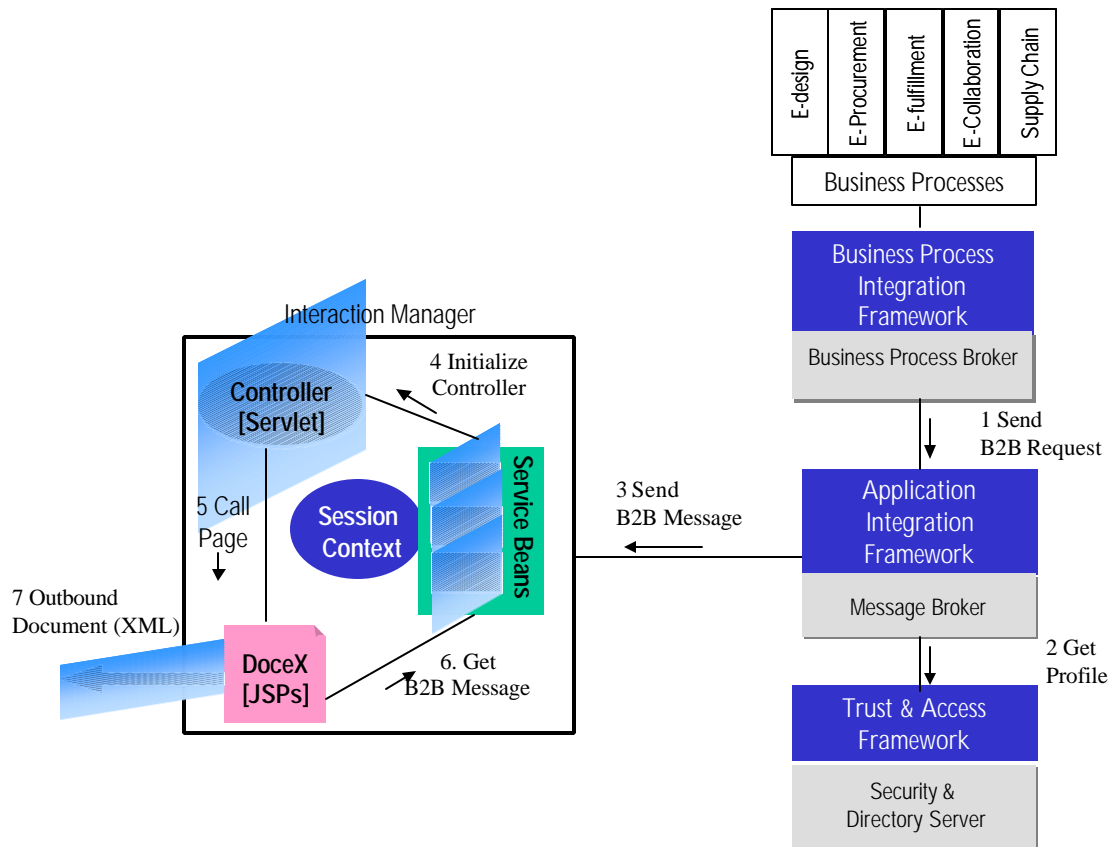


Figure 5: B2B Interchange Outbound Pattern

## 5 Trust & Access Framework

The Trust and Access Framework houses the Organization Model and the ACLs.

These form the basis for client authorization, which is the process of determining whether an authenticated client has the right to perform an operation on a specific resource in a secure domain.

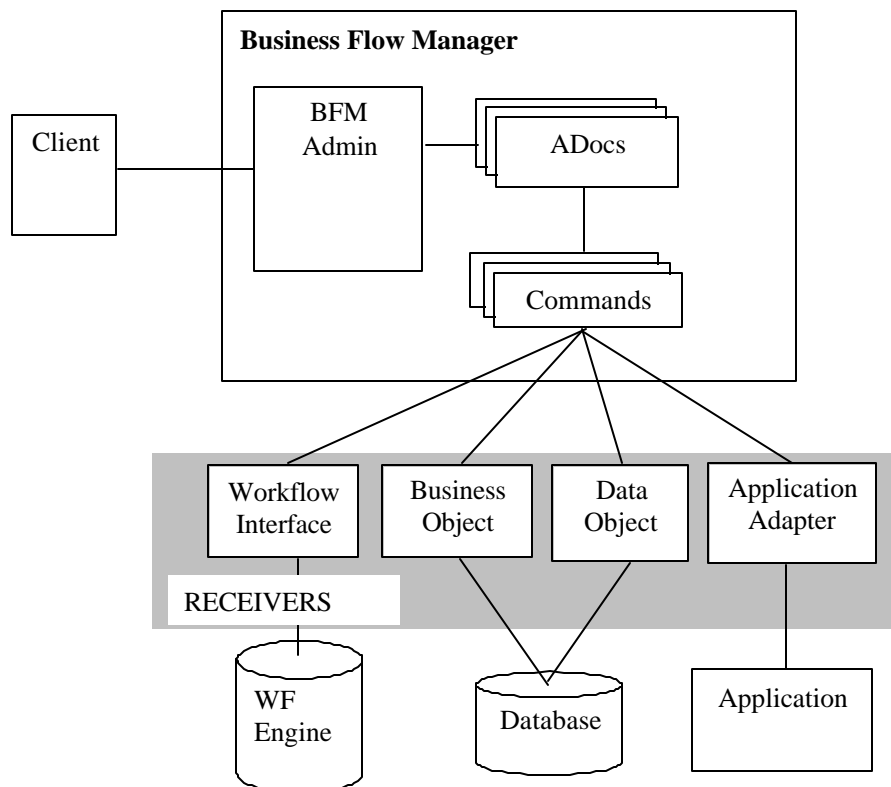
The organization model consists of intranet and extranet users as well as trading partner systems. The authentication is based on user id and password. The Framework provides a single sign-on mechanism, eliminating the need for multiple logon to backend applications that participate in the solution. An LDAP (Lightweight Directory Access Protocol) directory is used as the central user registry. The framework supports user management from a single point of control.

The framework enables the configuration of an authorization policy consistent with the specific requirements of a B2B e-Commerce solution. In particular, the Trust and Access Framework provides mechanisms to allow for fine-grained access control. In an e-market solution, fine-grained access control is critical as the confidential agreements with the trading partners determine the access model for transactions and resources.

The details of the Trust and Access framework are outside the scope of this paper. More information on trust and access control may be found in [11].

## 6 Business Process Integration Framework

The Business Process Integration Framework realized in the business process broker, hosts the solution components that are responsible for the management of the “business state” and the state-dependent orchestration of the business-processes. The Business Process Broker, also known as the Business Flow Manager or BFM, externalizes the flow definitions (control flows and data flows) and the business rules that drive the process choreography. The architecture of the Business Flow Manager is shown in Figure 6.



**Figure 6: Business Flow Manager Framework**

The key component of the Business Process Integration Framework is a collaborative business object referred to as the Adaptive Document (ADOC). An ADOC is different from a traditional business object in that its main objective is to serve as a broker of business services. It is designed using a Finite State Machine and the Command Design Pattern [3].

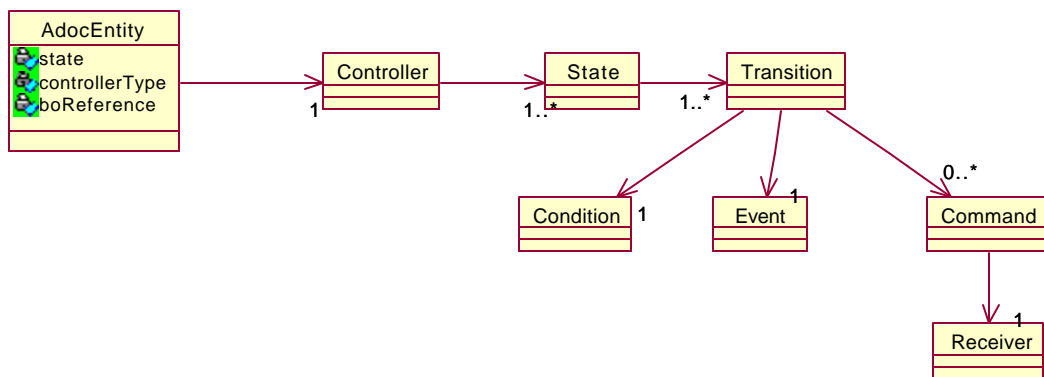
The BFM is populated with domain-specific ADOCs. The collective behavior of these ADOCs delivers the business process orchestration function of the business process broker.

## **6.1 ADoc Design and Implementation**

An ADoc represents a unit of collaborative business. An ADoc manages the state of that unit of business and provides a mechanism for implementing the state-dependent behavior of that unit of business. The ADoc is implemented using two components: an ADoc entity that stores the persistent state of the unit of business and an ADoc controller that executes the behavior. The controller is modeled using a Finite State Machine. A client may trigger a state transition in the Finite State Machine by means of a service request to the

ADoc. As part of the service request, the client passes an event identifier, a set of input parameters, and a context to the ADoc. Associated with each state transition is a set of commands. The controller executes these commands as a Logical Unit of Work. The commands are designed using the Command Design Pattern [3], which implies the actual work is delegated to a **receiver**.

Figure 6 shows ADoc Class Model:



**Figure 7: ADoc Class Model**

Though any component could serve as a receiver, there are four receiver types that are most commonly used in application development:

- 1) **Workflow Services Component**: This component implements joint flow interface, a standard defined by the Workflow Management Coalition and the Object Management Group.
- 2) **Business Objects**

3) Data Objects

4) Application Adapters: The adapters provide a mechanism to support asynchronous message passing based model of programming within the ADoc architecture. An ADoc could communicate with an application via asynchronous messaging using an adapter. The Application Integration Framework of the platform provides support for creating the adapters.

In addition to its primary role as a business services broker, the ADoc serves as a context-dependent aggregator of business content. An ADoc holds references to data objects and business objects that contain business data relevant to the unit of business. It aggregates business data via commands that uses these business objects and data objects as receivers.

## **6.2 Process Management in BFM**

The Business Flow Manager models long running business processes as directed flow graphs. A workflow engine provides the execution environment for these processes. Serving as a business process broker, an ADoc may engage multiple processes simultaneously. This translates to a one-to-many mapping from an ADoc instance to the set of active tasks in the processes that are currently

engaged by that ADoc. The ADoc extends process execution support by providing a mechanism to execute micro workflow for all such tasks.

A task is a node in the process flow graph, while the micro workflow refers to the activities that are performed in the context of a single task. The controllers and commands described earlier are used for micro workflow execution as well. A controller that executes micro workflow is called a Task Controller. The controller of an ADoc consists of a base controller, called Doc Controller, and zero or more Task Controllers. The task controllers are associated with the doc controller dynamically based on the active tasks that are brokered by the ADoc. This is the basis for the adaptive behavior demonstrated by an ADoc.

## **7 Solution Management Framework**

This is an agent framework that can be customized to provide solution-centric management and exception handling. The framework houses specialized domain agents that monitor the solution and respond to exceptions as they arise. The details of this framework are outside the

scope of this paper. Solution Management functions are described in detail in [11].

## **8 Application Integration Framework**

Application Integration Framework hosts “adapters” for XML-based enterprise application integration. These adapters integrate enterprise applications in the context of the business processes executing in the BFM. This framework is hosted on a Message Broker, enabling content-driven message routing, message transformation and translation, message repository management, and message aggregation and splitting. The details of the Application Integration Framework are outside the scope of this paper. More information on this framework can be found in [11].

## **9 Putting It All Together**

The five frameworks described above are the constituents of our powerful B2B e-Commerce platform. In this section, we revisit the requirements on the platform and show how the platform satisfies them. The Interaction Manager framework provides the integrated user experience, the Business Flow Manager framework delivers business process integration and brokering capabilities, the Trust and Access framework ensures common security mechanisms, Application

Integration framework presents a common messaging infrastructure, and the Solution Management framework supports solution management and monitoring.

## **10 Conclusion**

We have validated the platform by using it in numerous customer engagements to build complex B2B e-Commerce solutions. These include public e-markets, private exchanges, web-based procurement portals, and supply chain integration systems. These customer engagements have underscored the power of the platform by delivering on the promises of integrated user experience, common security mechanism, business process integration, and common messaging infrastructure.

## **11 References**

- [ 1] J.P. Baron, M.J. Shaw, and A. D. Bailey Jr., "Web-based e-Catalogs in B2B Systems," *Communications of the ACM*, Vol. 43, No. 5, May 2000, 93-100.
  
- [ 2] M. Fayed and D. Schmidt, "Object-Oriented Application Frameworks," *Communications of the ACM*, Vol. 40, No. 10, October

1997, 32-38.

- [ 3] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns – Elements of Reusable Object Oriented Software," Addison-Wesley Publishing Company, NY, 1995.
  
- [ 4] D. Georgakopoulos, H. Schuster, A. Cichocki, and D. Baker, "Managing Process and Service Fusion in Virtual Enterprises," Information Systems, Vol. 24, No. 6, 1999, 429-456.
  
- [ 5] R. H. High, Jr., "Component Model for Managed Objects in Large-Scale Distributed Systems," Component-Based Software Engineering, CUC96, Thomas Jell Editor, Cambridge University Press, SIGS Books, 1998, 117-136.
  
- [ 6] R. E. Johnson, "Components, Frameworks, Patterns," ACM 0-89791-945-9/97/0005, 1997, 10-17.

- [ 7] F. Leymann, and D. Roller, "Workflow-based Applications," IBM Systems Journal, vol 36, no 1, 1997, 102-123.
- [ 8] R. Prins, A. Blokdijk, and N.E. van Oosterom, "Family Traits in Business Objects and their Applications," IBM Systems Journal, vol 36, no 1, 1997, 12 - 31.
- [ 9] S. Tai and I. Rouvellou, "Strategies for Integrating Messaging and Distributed Object Transactions," Middleware 2000, J. Sventek and G. Coulson (Editors), LNCS 1795, 2000, 308-330.
- [ 10] C. Wilson, "Application Architectures with Enterprise Java Beans," Component Strategies, [www.componentmag.com](http://www.componentmag.com), August 1999, 25-34.
- [ 11] IBM Software Group, "WebSphere BtoB Integrator Solution Workbench Guide," WSBtoBI Library, June 2000.