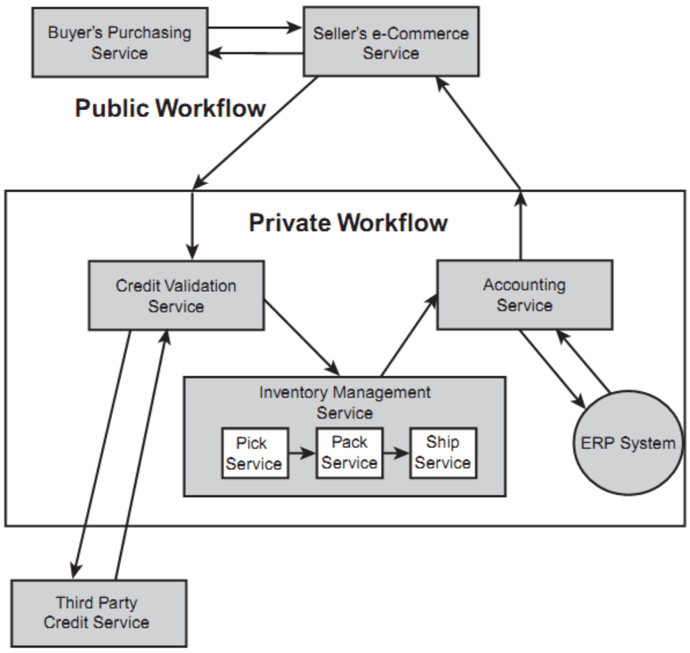
LOGICAL AND DEPLOYMENT ARCHITECTURAL VIEW

The Logical Architectural View: Composition of Web Services

The Logical (or Design) Architectural View starts with the end user's functional requirements and provides a top-down abstraction of the overall design of the system. In the case of B2B functionality (say, in the case of processing a purchase order), the user interface may be handled separately from the Web Services; therefore, the "end users" in this case are the businesses themselves. In other cases, Web Services may provide functionality to the user interface more directly.

In the B2B case, the functional requirements of a Web Services–based system will typically involve complex conversations among Web Services that participate in multi-step business processes. In addition, the individual Web Services involved are likely to be composed of component Web Services. As a result, an architect working from the Logical View will likely be concerned with workflows of Web Services.

For example, let's take the case of a buyer's Web Service contacting a seller's Web Service to make a purchase. Figure shows a possible (simplified) workflow for this interaction.


Buyer's Purchasing Service
Seller's e-Commerce Service
Public Workflow
Private Workflow
Credit Validation Service
Accounting Service
Inventory Management Service
Pick Service
Pack Service
Ship Service
ERP System
Third Party Credit Service

This workflow consists of two separate workflows: a public workflow as well as one private to the seller. From the buyer's point of view, the seller is exposing a single public Web Service that is composed of separate Web Services in succession.

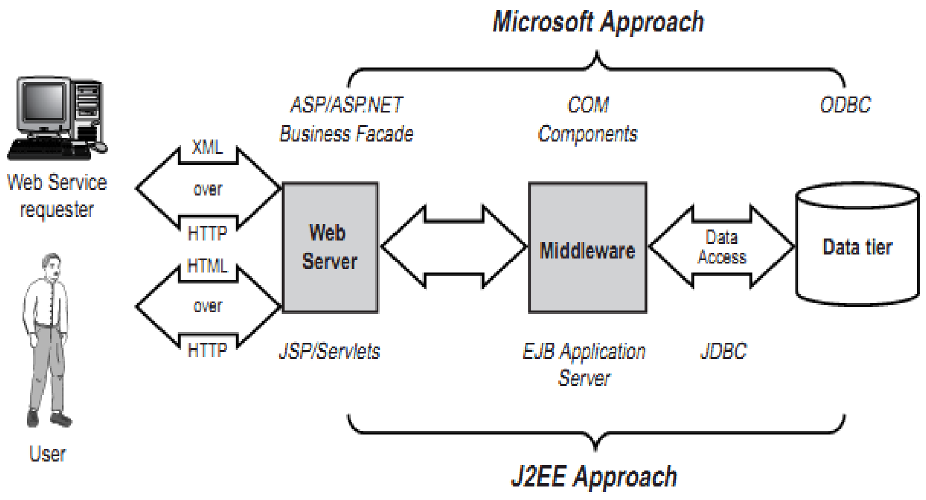
The interfaces to the two public services are both written in WSDL. The buyer has obtained the seller's service description beforehand—either by looking it up in a registry or through a prearranged relationship between the buyer and the seller. The buyer uses the service description to build the SOAP messages it exchanges with the seller.

Once the seller receives a request from the buyer, a sequence of business processes within the private workflow takes place. First, a credit-validation service sends a request to a third-party, credit-checking Web Service, which it may have established a preexisting relationship with. This third-party service is an example of an enabling service. Depending on the response from the third-party service, the seller continues with the e-commerce workflow or possibly sends a "credit rejected" response back to the buyer. (The architect must consider both the "rejected" special case as well as how to handle the situation where the third-party credit service is unavailable.) In a more general case, it will likely not be necessary to query this service if the seller has an established relationship with the buyer.

Once the buyer's credit is approved, the internal credit-validation service sends a request to the inventory-management service. This service is recursively constructed from individual component services (three of which are shown for illustration purposes, but in reality such services would be more complex). The architect must determine the interface for the inventory-management service as well as detail the workflow that takes place within the service.

The Deployment Architectural View: From Application Servers to Peer-to-Peer:

The Deployment (or Physical) Architectural View maps the software to its underlying platforms, including the hardware, the network, and the supporting software platforms. Today, Web Services are hosted on application server platforms such as IBM's WebSphere, BEA's WebLogic, and Microsoft's Windows 2000. There are many benefits to building Web Services on top of platforms like these: They handle database access, load balancing, scalability, and interface support as well as provide a familiar environment for dealing with hardware and network issues.



This model follows a traditional n-tier architecture, except that the Web server is also responsible for sending and receiving the XML messages that form the Web Services interface. The technology that supports Web Services is therefore already well under- stood; the fundamental difference between Web Services and Web pages is that pages are intended for humans to read, whereas Web Services expose an interface intended for machines.

Running Web Services off of Web servers is not the only way to support the services, however. It is also possible to build Web Services on a peer-to-peer (P2P) developer model. P2P, popularized by the Napster music service, is a distributed

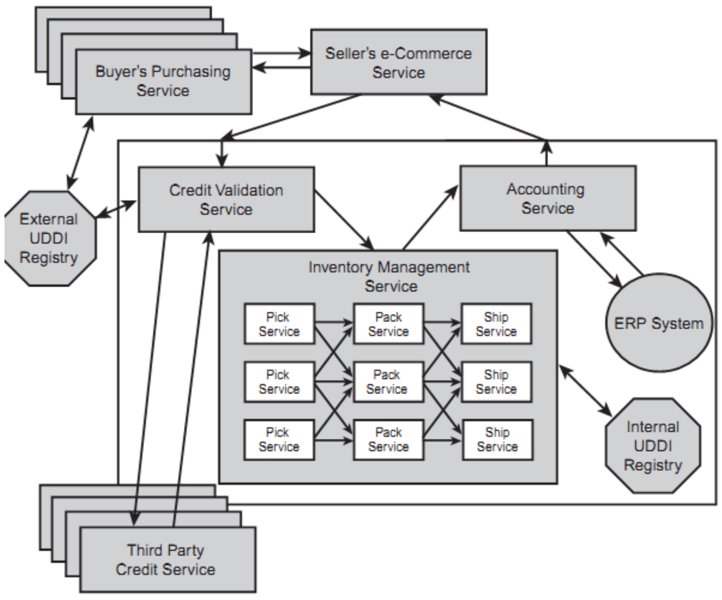
architecture that does not rely on central servers but rather distributes responsibility to systems (called peers) in the network. Unfortunately, P2P technologies are every bit as new and bleeding edge as Web Services, so only time will tell which P2P models will become established. The self-organizing promise of Web Services does lend itself to P2P, but a lot of work remains before we will see how this fascinating area will develop.

The Process Architectural View: Life in the Runtime

The Process Architectural View addresses all runtime issues, including processes, concurrency, and scalability. As the applications of Web Services move up the hierarchy of Web Service integration options to JIT integration the Process Architectural View will take on increasing importance. In fact, the Process Architectural View will be where the bulk of the SOA architect's work will take place.

For example, let's take another look at the simple e-commerce workflow. If you just look at the figure, you might think that there's nothing much new here; this diagram could represent an e-commerce system based on a simple n-tier architecture.

The reason that the diagram doesn't immediately demonstrate the power of the Web Services model is that in the diagram, the buyer has already identified the seller, the seller has already identified its third-party credit service, and the seller's private work- flow is already put in place. If all these statements are in fact true, then, yes, Web Services has little to offer over traditional n-tier architectures. On the other hand, let's take a JIT approach, as shown in Figure.


Buyer's Purchasing Service
Seller's e-Commerce Service
Credit Validation Service
Accounting Service
External UDDI Registry
Inventory Management Service
Pick Service
Pack Service
Ship Service
ERP System
Internal UDDI Registry
Third Party Credit Service

Source: http://nprcet.org/e%20content/Misc/e-Learning/IT/VIII%20Sem/IT1451%20-%20XML%20and%20Web%20Services.pdf