E-Commerce Supply Chain Management
Domain Research and Standard Architectures

Kunal Chopra, Jeff Elrod, Bill Glenn, Barry Jones

Introduction

E-Commerce Supply Chain Management involves the co-ordination of ordering supplies and raw materials at a rate just sufficient to produce products at the rate of consumer demand. The goal is to move from push forecasting to consumer pull based delivery of materials. This will eliminate the costly effects of having to store and warehouse raw materials but the burden of keeping everything running on schedule and ordering the proper inputs at the proper time must be solved efficiently for the scheme to work.

This is where an efficient and well-suited architecture fits in. We must fully consider the needs of all of the actors who are involved, as well as the systems they may currently be using before architecting our Supply Chain Management (SCM) system. In the following sections we present our analysis of the domain, actors, features and requirements, uses, and the constraints and drivers of the architecture. Finally we present some sample architectures that have been applied to similar problems.

Analysis of the Domain

Our SCM system will be deployed into an already functional supply chain or a supply chain that is just being established. In either case, our system may be deployed into one segment of the supply chain or all segments of the supply chain. Here a segment is an individual entity on the supply chain, a manufacturer. This means there must be an interface for communicating with another instance of our system or some other ordering system above and below the current segment on the supply chain. We should note at this point that the use of the term supply chain is a bit misleading. The chain is more like a tree that is rooted at the consumer, with each intermediate node being a supply chain manufacturer and the leaves being raw inputs that are mined, harvested, or otherwise obtained. Thus there is a ripple effect throughout the chain, when demand increases at the end-consumer level all parts of the chain will need to produce more to meet the increased consumer demand. In addition to wanting the ripple effect to take as little time as possible, each segment of the supply chain also desires a quick throughput time. The throughput time of a segment on the supply chain is the time it takes to convert money invested on input inventory into new money through sales.

Our system must influence throughput time by rebalancing input inventories and creating a flow of input stocks to keep up with the demand. The condition of balance is met when different items at different input levels and consumption rates are expected to meet the current consumer demand. The SCM must find, through analysis, the strategic buffers (the longest path) in the segment's manufacturing process and pace every other input accordingly such that consumer demand is met. It is worth note that our system will not replace the legacy software that is already being used by the supply chain segments but will work with it to achieve balance.

Actors

We found that there are at least four types of actors: Customers, Suppliers, Authorities, and Agents. For any one member of a supply chain (a manufacturer or retailer) there will likely be numerous instances of each type of actor. Customers are either the end-users of a product or other SC segments. Likewise suppliers are either the leaves of the supply chain tree that provide raw materials or they are other SC segments that provide intermediate goods to other SC segments. The Authorities are the actors that impose political and other constraints that the system must take into account before proceeding with a transaction. Finely the Agents are the actors that make a transaction possible. The Agents include credit card brokers, banks, and transport providers.

Features and Requirements

In this section we will present what each actor demands of our system, or what our system should provide each actor. Any given segment on the supply chain may have multiple suppliers and multiple customers so our architecture should easily support this. These customers and suppliers may also change quickly so we must be able to adapt at an equal rate.

The SCM system will also need to maintain information on the current inventory that the SC segment sells and that which it requires. The SCM must be able to adapt to availability changes on the supply side as well as on the consumer side. These changes in demand may occur rapidly or cyclically and the SCM must adapt. The SCM will also be responsible for brokering a contract on the supply and consumer sides of the SC segment. This requires the SCM to provide some sort of shopping interface for the consumers and some ordering interface that will work with the supplier's sales interface. There will also be some agent that handles the transaction or phases of the transaction. This includes agents that handle payment aspects and agents that handle delivery of the product.

The customer needs a current knowledge of the status of the supply of any good or product. The customer also demands the best product at the best price. Further desires are reduced restrictions, simple use procedures, and minimum operation costs.

The seller (the supply chain segment or manufacturer) needs better market share, highest profit, and reduced restrictions. Time to market can effect market share and is also important. Simple procedures and low operation are also important. The seller also needs guaranteed payment from the customer.

The authority actors will need to be able to ensure that regulations are easily enforceable and the agents will need efficient procedures with a minimum of restrictions and delays. Furthermore most agents will have proprietary systems and we will need to architect a common interface to use with them.

Uses

The uses of the system involve purchasing, orders, and contracts. Purchasing is the whole process, from decision making to ordering to contract agreement to settlement. An order is part of the purchase use case and involves a list of desired products and a contract. A contract is the agreement to trade.

Another use of the system involves inventory management. This encompasses the whole process of keeping orders of the needed supplies scheduled to meet consumer demand.

Constraints

Various constraints on the system exist and this list will grow as we further refine our understanding of the SCM system. At this point we anticipate political constraints of trade barriers, embargoes, and government regulation. The economic constraint of the ability to pay for an input that is needed by an SC segment or a product ordered by a consumer will be important. Also the environmental constraint of dealing with hazardous goods and health could affect our architecture.

Drivers and Quality Attributes

Our architecture must be designed for the following quality attributes:

- Scalability – The supply chain changes over time and some supply chains are fairly dynamic so the architecture must work well deployed to one supply chain segment or to the whole chain
- Usability – This is discussed in the constraints section, and is required by the consumers and the suppliers
- Security – The system controls inventory for a supply chain, as well as purchasing information such as bank account numbers and credit accounts. If compromised it could be devastating to the consumer or to the supplier.
- Accuracy – Mistakes made by the system could be extremely costly, so the system should be architected in a way that ensures testability and accuracy.
- Modifiability – This is a rapidly changing domain and the system should be changeable as new techniques and strategies come into play.
- Maintainability – The SCM system, once installed, is expected to be used for a long period of time. Therefore we need an architecture that will support upgrades and customizations as time progresses.
- Performance – The system must perform at a rate sufficient to handle all customer orders as they arrive while also maintaining inventory information and keeping the input inventory at a sufficient level.
- Traceability – For the sake of security and liability all transactions should be logged and recorded.
- Modularity – By creating a highly modular architecture we will positively affect Modifiability, Maintainability, and Scalability.

Standard Architectures

Chains of commitment architecture

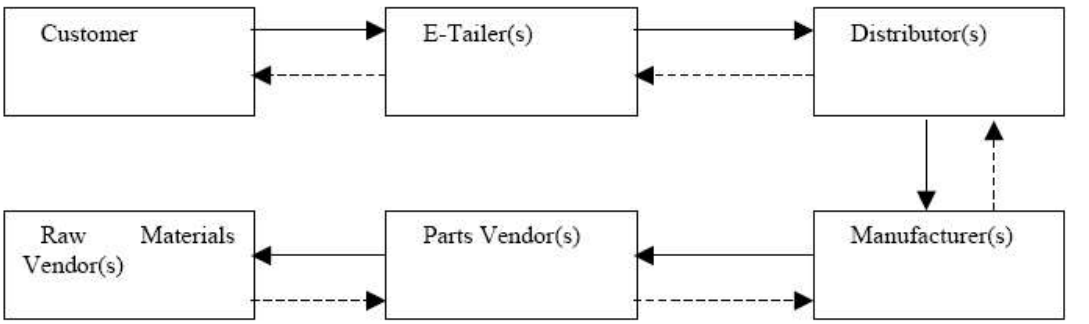
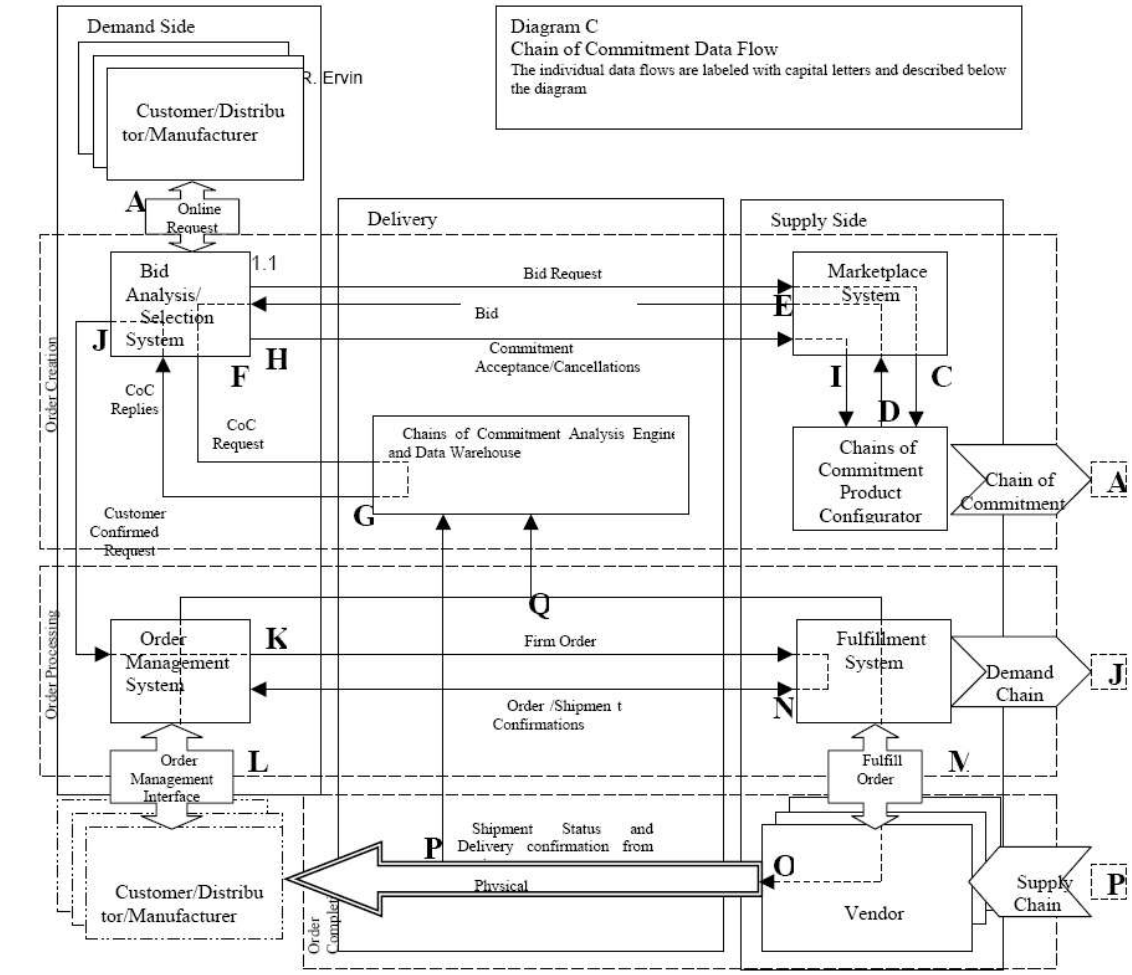


Diagram A: An example chain of commitment
Solid arrows are "commitment requests". Dashed arrows are vendor commitments.

We find this architecture interesting in that it provides a view of supply chains that is similar to that which we have observed from the requirements documents. The diagram above is a simplified view of a supply chain, though the text of the paper (chains) acknowledges that the solid arrows in the diagram may well represent a one-to-many relationship. The dashed arrows mate with the black arrows one-to-one.

Unfortunately the architecture diagram presented is confusing; it is more of a data flow diagram than anything else. Still, the domain is divided into "components" that do seem logical. The following elements are present in this architecture: The order system, the fulfillment system, the marketplace system, the chains of commitment product configurator, the bid/request analysis system, and the chains of commitment analysis engine. It is worth note that the chains of commitment product configurator allows for customization of the products ordered, and the chains of commitment analysis engine is responsible for handling product orders and input scheduling.

Should we choose this pattern, or rather a similar set of patterns, we would implement our Balanced Flow™ type engine in the analysis engine. One interesting aspect of this particular scheme is the obvious separation of modules into three layers stacked horizontally. This makes sense to separate the demand side from the supply side with the decision making layer having access to each. There are also three logical layers stacked vertically including order creation, order processing, and order completion.

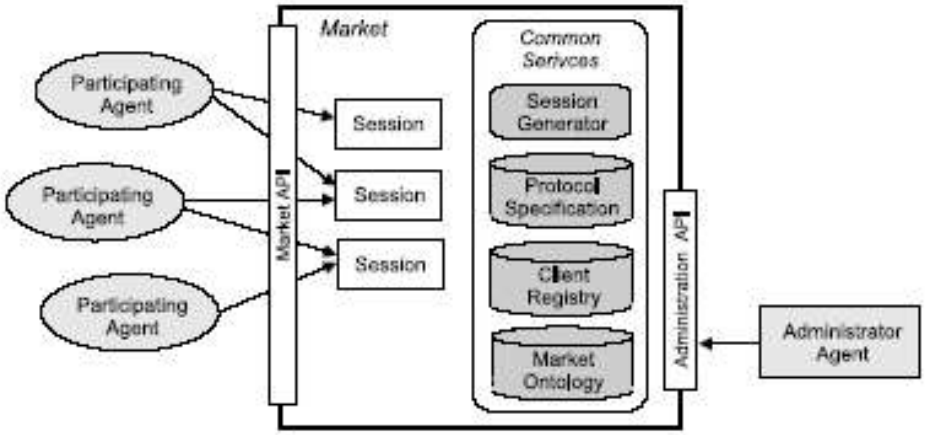


A. The demand side partner (customer) generates a request for bid.. This could be in response to an internal or an external request (i.e. a customer placing an order).
B. A bid request is submitted to one or more marketplace systems. This initiates a large chain of requests, but only the immediate request is illustrated here.
C. Marketplace system requests COC Product Configurator to analyze the request, determine the chain of commitment requirements, then submit one or more commitment requests, creating a new link in the chain. That is, this is spawning a new instance of data flow (A).
D. After receiving all the vendor requests.
E. The marketplace systems respond with a number of bids and their accompanying commitment keys (these are temporary commitments until a firm order is placed).
F. The bid analysis engine submits a request for a CoC quality analysis from the CoC Analysis Engine for each bid under consideration.
G. The CoC Engine computes a score for each vendor bid based on historical information and customer determined criterion.
H. The Bid Analysis Engine selects from the available bids based on the score and sends commitment accepts and rejections to the marketplace applications.
I. The commitment acceptance or rejection is forwarded to the product configurators for complex bids, which then send a chain of acceptances or cancellations for all the requests being held by them. This protocol remains to be developed as the subject of a separate research or development effort.
J. The bid analysis engine sends the customer request to the Order Management System, where it is managed for the remainder of its lifecycle.
K. The OMS sends the approved order to the winning vendor's fulfillment system.,triggering a series of (J) data flows throughout the chain of commitment.
L. On the customer side, the order is tracked in the OMS by the customer or the customer's purchasing agents.
M. On the vendor side, the order is managed in the Fulfillment System until it is shipped.
N. All status updates (acceptance, rejection, ship confirmation, etc) are shared between the OMS and FS for the life of the order.
O. When the order has been completed, the product is shipped via the designated carrier by the vendor.
P. When possible, information from the package carrier is either sent to or pulled by the CoC Analysis Engine as part of the quality of commitment historical information.
Q. The OMS and FS share data with the CoC Data Warehouse. This data is crosschecked and stored for future use.

Again, this is a data flow diagram but it does convey the relationships between the modules, and it does illustrate the layers used. This architecture and the diagrams come from (Ervin).

Agent Based

The diagrams presented in the two papers we found on agent based architecture left a little to be desired but the ideas seem potentially interesting. An agent is a unit of software that is autonomous, it runs as a process. Agents tend to be composed of multiple components that work together to provide a common service or interface. Then a collection of agents is interconnected onto an agent grid (which may span multiple processors).



This diagram (Collins) provides a hint at what the authors were presenting in the paper, agents are used to connect to a central analysis engine. The agents contain an architecture similar to that shown below (Liu).

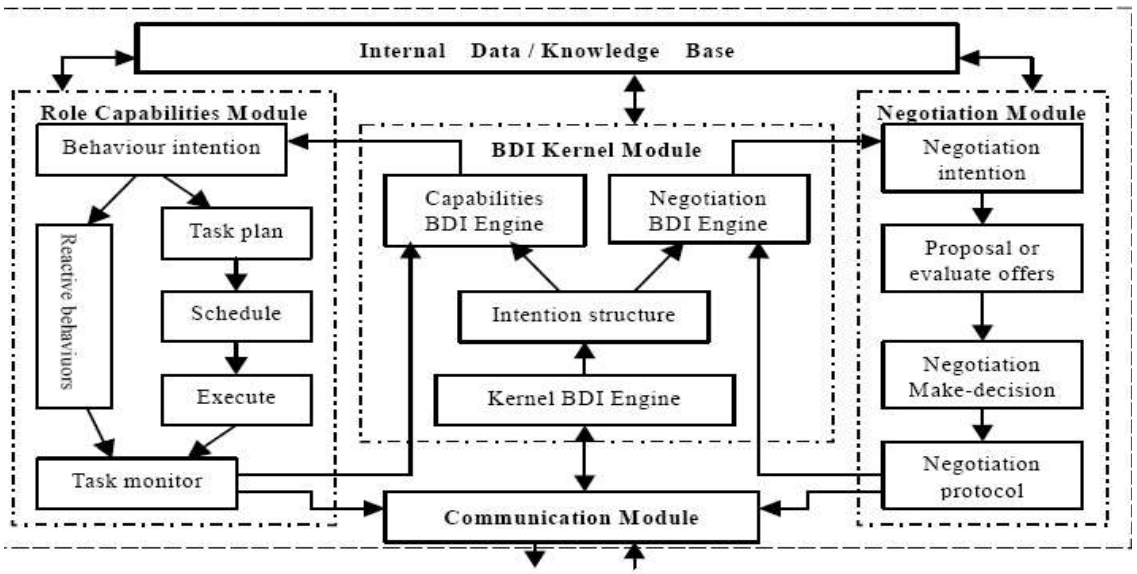
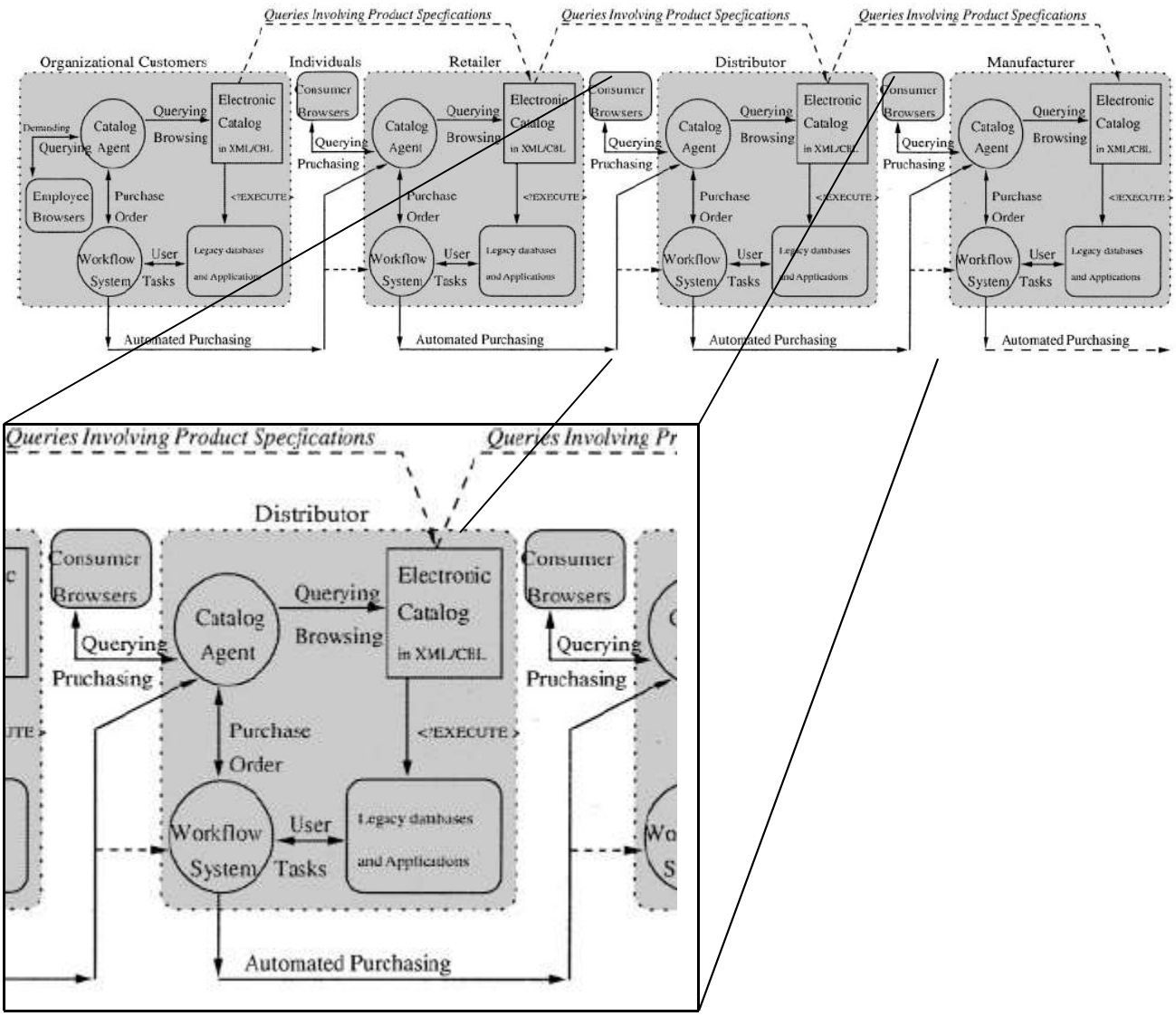


Figure 2: Agent Architecture for SCIC

One further supply chain architecture



This pattern (Cingil.) shows the use of agents a bit better than the previous diagram. The grey dashed-line boxes represent the separate entities and the boxes inside are agents. Their application didn't involve a supply chain but with a little work the four large elements to the diagram could be merged into layers and an instance could run at each SC section.

Combined Approach

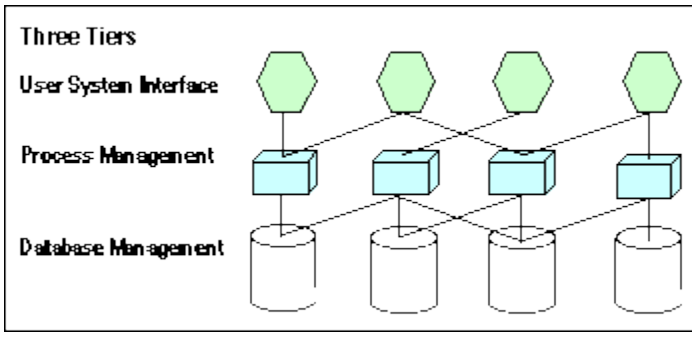
What we've seen so far is that there aren't too many references to complete architectural patterns used in this domain. This is due in part to the system strategy being a relatively new concept. This is also due to the proprietary nature of the current solutions to this problem. We will probably end up taking ideas from the above patterns, especially the chains of commitment pattern, and combining them with the two strategies shown next.

For security reasons a three-tier architecture seems appropriate, and it also provides usability in that it is specifically designed for high load and high throughput situations. We could then use J2EE to provide a secure user interface for both end-user web orders and web service connections from other SCM systems. The chains of commitment unit would then run in the business logic tier with the data warehousing performed by the database management tier.

The Three-Tier Client/Server Architecture

Another architecture in consideration is the three-tier architecture. See figure below. The three-tier architecture was created to increase the number of users as well as improve performance, increase flexibility, modifiability and reusability. The traditional two-tier client and server architecture could not handle a large number of users. The three-tier architecture consists of a client/user layer, a middle layer that provides process management where business logic and rules are executed and can accommodate a large number of user, and a database management layer for data storage and retrieval.

The client/user layer, also called a presentation layer in an e-commerce system is usually implemented in a language for web-based applications. A combination of HTML, Javascript and CSS is used for client side processing. The middle layer or application layer is responsible for carrying out the business rules of the system. It is also responsible for accessing data from the database management layer. The database management layer or persistence layer consists of database tables for storing data, views for presenting data for security or to hide the database structure and stored procedures or functions for maintaining database integrity.

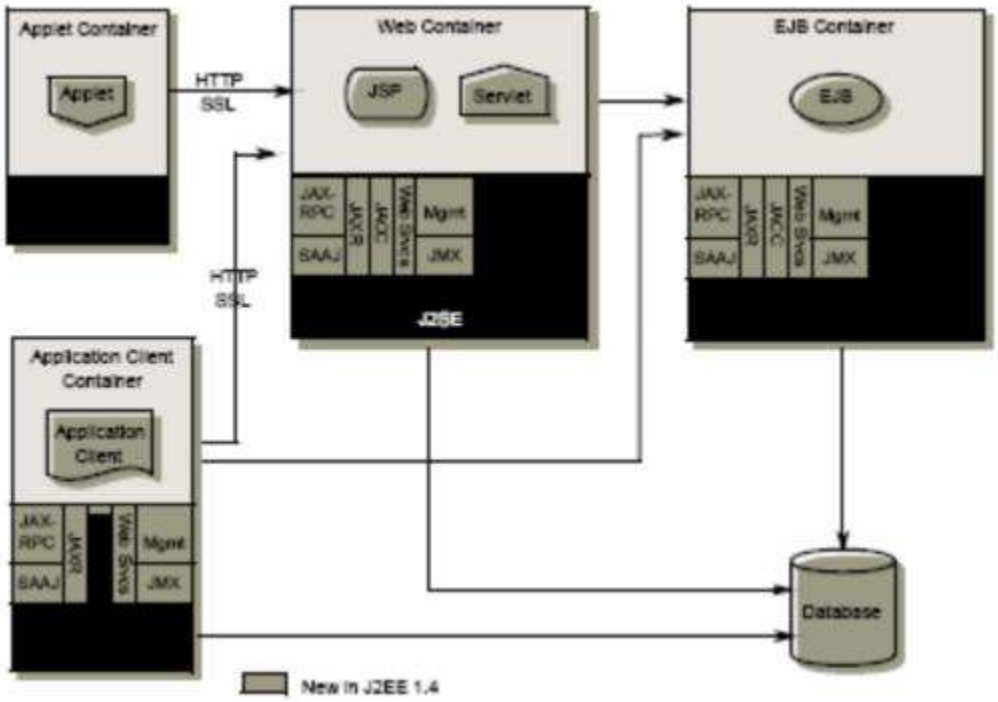


J2EE Architecture

Another architecture under consideration is the J2EE architecture. The architecture allows for an application to be secure, scalable, portable, transactional, distributed and multi-tiered. Basic client side views are presented using JSP pages to be viewed through a browser, while Servlets handle control structures, data access and data mapping.

Client applications may access secure web services provided by an EJB container to pool data and remote services from other vendors. For example, a supplier may use a desktop application to handle inventory management while accessing a web service to provide updated inventory records to our centralized database. Likewise, when a purchase is made through the web system, an EJB could be accessed to call a web service to notify the supplier of the sale and schedule a pickup and delivery from a shipping company.

Overall, the ability to use web services transparently and securely within J2EE architecture allows the seamless pooling of resources throughout a transaction process.



Sources

Cingil, Ibrahim and Dogac, Asuman "An Architecture for Supply Chain Integration and Automation on the Internet." Software Research and Development Center, Middle Eastern Technical University.

Collins, John et. al. "A Framework for Mixed Initiative Agent-Based Contracting." Department of Computer Science and Engineering, University of Minnesota

Ervin, Rich. "Chains of Commitment Software Architecture." 2MX Consulting, NanoGuys.Com

Liu, Sanya and Wang, Hongwei. "Agent Architecture for Agent-based Supply Chain Integration and Coordination." Institute of Systems Engineering, Huazong University of Science and Technology, Wuhan, China