Designing a flexible services-based architecture for Internet Applications

Vijay Mehra
KPMG Consulting, LLC
99 High Street,
Boston, MA 02110
1 617 988 5681
vmehra@kpmg.com

Richard K. Walker
KPMG Consulting, LLC
757 Third Ave
New York, NY 10036
1 212 954 2474
rkwalker@kpmg.com

Jeffrey S. Brashear
KPMG Consulting, LLC
99 High Street,
Boston, MA 02110
1 617 988 1386
jbrashear@kpmg.com

Mohan Tavorath
KPMG Consulting, LLC
99 High Street,
Boston, MA 02110
1 617 988 5953
mtavorath@kpmg.com

ABSTRACT

This article describes a flexible architecture for developing services-based business applications that use component-based architectural services to demarcate the application into various tiers that enable the decoupling of each layer, both logically and physically. This provides the speed and flexibility needed to embrace new product ideas, channels, and markets.

1. ARCHITECTURE OVERVIEW

The following system architecture description describes the design of a distributed, object-oriented, multi-tiered, Internet-based application. It describes both conceptual and implementation-specific views of particular approaches, technologies, object models, and communication protocols. Architectural decisions regarding specific tools and platforms that may be used to construct such an application were made taking into account constraints such as existing skill sets, standards set by the organization, and future state architectural direction. This involved build-versus-buy decisions based on the effort required to integrate and maintain the application. It is based on practical experiences that have extracted lessons learned from various projects. The use of an eXtensible Markup Language (XML)-enabled architecture created the opportunity to deploy quickly and interoperate with a wide variety of business systems

2. ARCHITECTURAL APPROACH

The architecture consists of four major layers/tiers of abstraction enabled by the use of eXtensible Markup Language (Client, Context Processing, Business Logic, and Legacy systems). This abstraction layer is built on middleware that connects to back end systems by wrapping an adaptor interface layer around legacy systems, thereby providing a standard interface. XML is used for structured data messaging, sending rich, self-describing platform-independent streams of data between legacy systems and new applications in XML format.

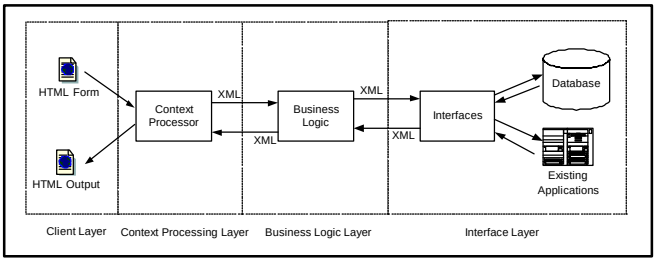
OOPSLA 2000 Companion Minneapolis, Minnesota

The architecture strives to:

- be scalable, flexible, adaptable and extensible to support the changing business environment
- support a consistent strategic architectural vision
- reduce cost of technology ownership
- leverage existing assets (platforms, environments, tools, custom development, and packages)
- reuse, buy, and build needed functionality (in that order)
- support short-term tactical implementations and the long-term strategic vision

3. COMPONENT LAYERS

The architecture has been separated into four layers, each one completely decoupled from the layers with which it interacts. Communication between each layer uses XML data streams to ensure that inter-layer messaging is platform-independent. Communication within each layer may use proprietary protocols or XML and these decisions are made contingent upon the speed and efficiency requirements of the application.



3.1 Client Layer

3.1.1 Conceptual View

For every new request made by the Client Layer, data from the back end database server is processed by the application server and arrives at the web server in the form of a standard extensible Markup Language (XML) document. The web server then dynamically generates Hyper Text Markup Language (HTML) from this XML using eXtensible Stylesheet Language (XSL) files. This HTML file is then displayed in a browser frame on the screen. This achieves the important objective of decoupling the format and display characteristics of the user interface from the content of the data being displayed. Changing the look of this incoming data is simply a matter of changing the XSL stylesheets. The same XML data content can also be rendered on any other client User Interface.

3.1.2 Implementation View

The client-tier displays HTML pages that conform to the HTML 3.2 specification and supports 4.x or higher of either the Navigator or Internet Explorer browsers.

3.2 Context Processing Layer

3.2.1 Conceptual View

The context processing layer takes the input HTTP request object and converts input parameters into an XML stream that is then passed on to the business logic layer. The business logic layer performs required processing and returns an XML output data stream. The style sheet-rendering engine merges the XML data stream with the appropriate XSL template file to render HTML output to the browser.

3.2.2 Implementation View

The user interface controllers are Java Servlets. These are inherited from a base ancestor Servlet and use an HTTP-Request-to-XML transformer Java component to convert the name-value pairs that are intrinsic to the HTTP Request object into an XML stream that represents those input parameters. This provides the flexibility to construct more sophisticated XML streams that more accurately represent a business domain or metadata model. This also effectively separates the client-layer access mechanism, e.g. a browser, from the request coming in to the web server. This input data is processed by the business logic layer, which returns an XML output stream that represents the output data.

3.3 Business Logic Layer

3.3.1 Conceptual View

The business logic layer accepts an XML data stream with input parameters and passes it to a business controller object (BCO). The business controller object uses a business object factory (BOF) to instantiate appropriate business objects. Each business object instantiates shadow objects that talk to the interface layer.

3.3.2 Implementation View

The User Interface Controller (a servlet) passes the input data as an XML data stream into the Business Controller Object (a Java object). The business controller object represents business logic associated with a use case (or a set of use cases), as defined by the business requirements and use case analysis. The business controller object uses a Business Object Factory (BOF) to instantiate Java Business Objects that collaborate to process the required functionality. The business objects have shadow objects associated with them that provide an abstraction layer that encapsulates access to back-end or external systems that are accessed. The results are returned as an XML stream to the context-processing layer.

3.4 Interface Layer

3.4.1 Conceptual View

The interface layer accepts an XML data stream that represents input parameters as criteria to request data from or t persist data to back-end systems. Interface objects communicate with those back-end systems, leveraging existing native or open communication protocols to retrieve data and then pass them back as an XML data stream to the business logic layer.

3.4.2 Implementation View

Interface objects are implemented as Java objects that utilize appropriate mechanisms to access back-end systems. These use native and XML interfaces to facilitate speed of access and persistence to those back-end systems. Some of these were determined to be reusable across other applications that perform the same function, and were converted to component-based services (CORBA-based) with clearly defined interfaces.

4. ARCHITECTURE SERVICES LAYER

The Application Architecture can be subdivided into services. A service is a collection of classes or components that provide some common functionality to be used by the other components across the application.

4.1 Runtime/Deployment Infrastructure

These services represent the runtime environment for the application and include the following components:

Operating System: The application can be developed on various environments such as Solaris or Windows NT/2000. This decision is made based upon the software products used for application development.

Network Connectivity: The protocol used for the underlying network connectivity.

Application Server: The runtime environment for the business application is provided using standard Application Servers.

Database Server: Any standard relational database server may be used for the application as per the system requirements.

4.2 Application Architecture Services

These services represent the components/classes that provide the common functionality required to support the components that constitute the Application Services Layer. The following application architecture services were developed:

Database Connections and Transaction Management: Database connectivity between the application and the database is maintained using appropriate drivers.

Error Handling: The error handling service traps and logs errors related to application business logic, database access and system infrastructure.

Security/Encryption: The security service prevents unauthorized users from gaining access to the system or gaining access to personal information about system users. This service is embedded in to the component layer at various levels to prevent unauthorized access to the system.

Application Utility Services: These services represent the components that provide supporting services to the Application services layer.

5. SUMMARY

The architectural approach outlined in this paper delivers technology solutions through the aggregation of granular services and business-based components, achieving a high degree of reuse. The use of eXtensible Markup Language (XML) created the opportunity to deploy quickly and interoperate with a wide variety of business systems. providing the speed and flexibility needed to embrace new product ideas, channels and markets.