The IBM® WebSphere® Integration Reference Architecture enables organizations to take a service-oriented approach to integration and to avoid the pitfalls associated with traditional integration approaches. This paper explains how this complete and comprehensive architecture covers the breadth of integration needs within an enterprise.

From the IBM WebSphere Developer Technical Journal.

Introduction

The emergence of service-oriented frameworks results from the evolution of software development and implementation over the last 20 years. Our industry has evolved from monolithic applications and hard to manage client-server solutions and has now discovered that the incremental development of components, enabled via a service-oriented architecture (SOA), increases the quality of applications, the speed of development of new solutions, and better addresses the requirements of business stakeholders.

The concepts embodied in SOA enable an enterprise to integrate its business processes across its lines of business and their supporting information systems. The ability to coordinate the challenges of enterprise-level integration necessitates an architecture to facilitate the modeling and managing of services spanning information, applications, and people. Through a component-based approach, organizations can build more flexible integration solutions that leverage a common set of core infrastructure services. This solution framework enables increased agility and adaptability of IT solutions through a more simplified loosely-coupled approach. An SOA-based integration foundation provides support for technology co-existence through support for a standards-based design, development, and implementation. This approach provides a scalable infrastructure across existing and future technology assets to provide a solid foundation for the enterprise integration architecture.
Although many integration products claim to support this architectural approach, many products fall short in their ability to provide enterprise integration requirements. This article focuses on the breadth of capabilities required to support the myriad integration styles required for an enterprise-level approach to integration, and discusses the need for a comprehensive service-oriented architecture built on open development and run time standards as the critical component for integration. Through this architectural foundation, enterprises can extend their solutions by combining existing components with new components and by implementing composite applications as integration requirements emerge.

**Enterprise integration strategy**

An enterprise integration strategy requires, at a minimum, the exchange of messages as the foundation of the solution architecture to enable the integration of people, applications/processes, and information. Through this approach, integration solutions can be implemented at departmental levels and easily extended across enterprise boundaries. This approach to integration provides flexibility and a time to market advantage for an enterprise and becomes a critical differentiator for success in today's complex and competitive business environment.

Currently, a shift is occurring in enterprise integration strategies. Although tactical initiatives continue to be deployed to solve departmental requirements, organizations are defining enterprise integration architecture initiatives based on an SOA approach. This approach requires organizations to identify core integration assets/components and advocates an approach to encourage the reuse and refinement of these assets for integration projects. In this manner, integration across departments can be approached as an enterprise-level initiative that promotes reuse through a standards-based approach to definition, discovery and invocation.

An enterprise approach to integration is compatible with a more tactical approach to development of departmental/line of business integration solutions. In fact, the tactical solutions become assets for the overall integration repository. It is important to provide a governance model that ensures that the development of tactical solutions is done within the context and rules of the enterprise's integration architecture. Without this rigor, enterprises will continue to build departmental integration solutions in isolation and fail to achieve discernible long-term benefits from integration. Given the rate of change in business and technology, the lack of governance on integration approaches will to hinder the ability to effectively and proactively support integration objectives across the enterprise.

**Service-oriented integration architectures**

Coupled with the need to provide a disciplined approach to integration, it is essential that the enterprise integration architecture leverage the strengths of a service-oriented architecture. Service-oriented architecture enables the creation of modular composite applications through the packaging of new as well as existing functions as reusable service components using open and available standards. Service-oriented architectures provide a "set of business-aligned services that are combined (composed and choreographed) to fulfill business goals ... services are manifested as a set of interfaces without any dependencies on the implementation mechanism or location." (Endrie, 2003) Loosely coupled component architectures are powerful as they provide
the ability for components to act as service consumers and/or providers by exposing interfaces in a standard format across a distributed topology. Composite applications provide improved flexibility of IT systems through function isolation, an increased ability to reuse components in multiple contexts, a simpler model for integration, and flexibility in construction of integration processes.

The application of a service-oriented approach for integration provides many benefits:

- Leverages open standards to surface integration assets as services, fostering reuse of existing assets and helping avoid vendor lock-in.
- Provides a standard way to represent and interact with integration components (such as maps, processes, discrete transactions/services or interfaces), providing flexibility to reuse assets across multiple business process solutions.
- Shifts the integration focus to application assembly rather than implementation details, enabling efficient implementation of new and modified business solutions.

A service-oriented architecture provides for the connectivity of applications and organizational resources by representing these assets as services which can be composed into higher order service components. From a technical standpoint, a service-oriented framework for integration enables resources to exchange information (messages, documents, business objects) through a standard interchange framework. By expressing new and existing applications as services, services can be composed to address integration requirements.

The SOA approach enables an architectural style consisting of service providers, requestors and service descriptions enumerated through the familiar publish/find/bind service framework. SOA approaches enable and encourage design principles and patterns encompassing encapsulation, service composition, loose coupling, and reuse.

The service-oriented framework needs to be pervasive across all of the aspects of the integration solution. Some current middleware approaches support the ability to define or invoke integration components via SOAP wrappers, but integration architects need to recognize that an SOA approach is far more than just SOAP-based integration. The solution needs to support an ongoing evolution of interface definitions (both user-defined and industry-based) as they emerge from standards bodies such as OASIS and OAG. The underlying integration assets within the architecture must conform to an SOA-based approach:

- Service description and definition via a standard interface; for example, WSDL.
- Service invocation/interaction over a standards-based transport/mediation layer.
- Service choreography for orchestrating service interactions; for example, BPEL4WS.
- Service discovery through integrated registry/directory services; for example, UDDI.

Composite applications enable creation of coarse-grained services with flow logic to determine the execution sequence of the services. These composite applications become the IT analogy of business process in the business domain. The SOA design leverages middleware and operating environment functions to separate flow logic from the underlying business logic and implemented in the individual services. Services may be enabled from existing components (for example, mainframe CICS application) or may be completely new functions.
A key challenge in a service-oriented integration approach is in the definition and implementation of services at the appropriate level of abstraction. The ability to apply sound architectural design in developing an enterprise approach to integration is tied directly to the concept of separation of concerns as well as the emerging concepts of model-driven design and architecture.

**Separation of concerns simplifies integration architecture implementation**

Development and implementation of information technology comprises multiple dimensions that need to be reconciled within an enterprise-level integration strategy. IT’s role in defining and managing the boundary between business applications and the underlying operating environment becomes the key factor in the implementation of business solutions. The operating environment provides infrastructure services for business applications and represents common services to support the integration solutions.

When defining an enterprise's integration architecture, it is critical to consider the breadth of integration requirements. These integration requirements may include traditional workflow processing based on human task interaction, choreography of activities between different systems, distributed data management involving structured and non-structured information, and user interaction capabilities. An enterprise requires the ability to differentiate and design service artifacts appropriate for specific styles of integration required to solve a particular integration problem. As a result, the concept of separation of concerns is the foundation to the definition of the integration services.

Separation of concerns suggests decomposing business needs into services; and composing combinations of existing and new services that represent business processes. Through separation of concerns, integration requirements are decomposed into more granular services. This decomposition is accomplished through the identification of the specific functions that need to be defined to support business requirements, as shown in Figure 1.

**Figure 1. Separation of concerns**

![Diagram](image)

1. Decompose the Business Need into its necessary functions and services (Separation of Concerns)
2. Reuse/Create Business Application Specific functions and services (Create reuse services)
3. Utilize common services provided by the Operating Environment (Leverage Infrastructure)

RESULT: Composite Application representing a business process

The **separation of concerns** approach to defining services results in a design that isolates and characterizes functions and services that are implementation independent. With this approach, building integration solutions becomes an iterative process and components are refined over successive integration projects. Enterprises provide design governance and integration mentorship to development teams to promote best practices and the reuse of services and components.
In an SOA-based solution framework, service components are isolated and defined. In addition to supporting a top-down model-driven approach, the solution approach enables a bottom-up approach for the development of new services. The resultant composite application consists of service invocations with flow logic to manage the execution order of the individual services. The composite application can be made available to other services as a service, itself the development of additional composite applications.

The prevailing approach for building component-based architectures is via model-driven architecture (MDA), and the MDA approach provides a foundation for the development of an SOA. In An MDA Manifesto (MDA Journal May 2004), Grady Booch comments that an MDA is "a style of enterprise application development and integration based on using automated tools to build system independent models and transform them into efficient implementations". The MDA approach provides for more efficiency in the IT development process through tool-based generation of artifacts, as well as enabling business stakeholders to participate in development of business processes. MDA is rapidly converging with both SOMA (service-oriented modeling and architecture) and SODA (service-oriented development of applications) methodologies for service-oriented application development.

In Guide to Enterprise IT Architecture (C Perks/T Beveridge), Perks states that "IT persists in reinventing technical functions because the existing functions were not built by the current faction. At some times this approach may be perfectly justifiable; however in the many circumstances it is merely a whim of the project team. Application development is all about business functions, not building technical infrastructure". This comment underscores the need for an enterprise-governed approach to developing integration in tandem with the application of a separation-of-concerns perspective to building enterprise-level integration strategies.

The WebSphere Integration Reference Architecture

Business integration benefits from an enforcement of separation of concerns to integrate data and applications as appropriate for business needs. It is not enough to have a process solution or an application server foundation - it is the aggregation of multiple integration styles using common components and services that provide long term sustainability, preserves investment in existing IT assets, and avoids exposure to a "cobbled" integration approach. Effective business integration also requires an operating environment that supports the implementation of service-oriented solutions, can be implemented in a modular, build-as-you-go fashion, and supports all functions required for enterprise-level implementations.

The WebSphere Integration Reference Architecture provides a comprehensive set of services to enable business integration. The services provide the breadth of functionality needed to solve integration requirements. More importantly, the component services can be implemented in stages to enable incremental evolution on a project-by-project basis while working towards an enterprise integration solution architecture. Although specific projects may not require all of these services, enterprise-level integration will require the ability to add these functional capabilities to the integration architecture. The resultant architecture provides for separation of concerns by enabling business logic, control logic, routing and transformation logic to be loosely-coupled and,
as a result, more flexible to change. At the organizational level, this approach facilitates simpler integration solution development and enhances maintainability and operation of the solution.

The WebSphere Integration Reference Architecture (Figure 2) shows the key integration functions that are required for comprehensive, enterprise-level solutions. These service groupings provide the ability to apply separation of concerns to enterprise integration requirements, and lead to a convergence with the principles of service-oriented architecture, as they apply to integration.

**Figure 2. WebSphere Integration Reference Architecture**

This high-level architecture depicts the integration functions/services required to enable a comprehensive approach to integration. Since these services are described by their interfaces and not by their implementation, a given solution may be made up of mainframe applications, local, or remote services, choreographed processes described by BPEL (the standard for business process description) or components built with J2EE. The implementations of these integration components provide support for non-functional requirements including reliability, security, availability, and management at both the operating environment level, and the component/service level as well.

**Connectivity services**

At the core of the WebSphere Integration Reference Architecture is the connectivity services. This component provides the infrastructure to support and instantiate the Enterprise Service Bus (ESB) architectural pattern. The ESB delivers inter-connectivity services across the distributed component topology. Transport services, event services, and mediation services are provided through the ESB. Transport services provide the fundamental connection layer; event services enable the system to respond to specific events arising as part of a business process; mediation services enable the loose-coupling between interacting services in the system. The ESB essentially becomes the extended enterprise’s arterial system providing messaging, notification, and invocation services across the enterprise’s various operating environments. The ESB is a key factor in enabling the service orientation of the WebSphere Integration Reference Architecture to be leveraged for implementing service-oriented solutions today, as well as in the future.

The ESB provides for multiple connection technology alternatives to support multiple messaging topologies and patterns, along with implementations such as SOAP/HTTP, RMI/IIOP, and others. In most cases, loosely-coupled connections are required to facilitate the management of
interactions between services. However, some transactions, by their nature and critical business functionality, may require tightly coupled connections. Both patterns are supported within the WebSphere ESB architecture. The requirement for support of heterogeneous technologies is reflected in support for multiple protocols within a message flow instance, middleware interoperability, specification of different qualities of service (such as persistence, reliability, transactional management, availability), support for various message distribution, and routing (to include publish-subscribe, queue-based, and broadcast-based). Additionally, the connectivity services and ESB architecture provide support for specialized message/information delivery, such as Point-of-Sales, SCADA, and other pervasive device solutions. Customers using current solutions with WebSphere MQ can implement the ESB pattern today and will be able to support new protocols as they emerge.

Within the ESB, there are three major service dimensions that are provided as part of the connectivity framework:

- **Event services** provide event-driven services to enable the components (as well as organizational resources) to respond to stimuli; for example, business events.
- **Transport services** provide communication services across wired and wireless networks for synchronous and asynchronous message delivery with varying levels of delivery assurance to provide for both location and protocol independence.
- **Mediation services** enable dynamic in-flight message transformation, dynamic routing, and service binding resolution services.

An SOA approach to integration is not possible without the fundamental capabilities of connectivity, routing, and transformation. These functions become enabled via the ESB within the WebSphere Integration Reference Architecture and form the foundation of the solution architecture. The ESB, in many cases, is implemented using traditional and new emerging middleware solutions to provide access to integration components.

**Business logic services**

Business logic services provide the capabilities required for the execution of business logic in the form of service endpoints. The implementation of service endpoints is a critical part of comprehensive integration architecture. Services may be provided through any combination of existing applications; through newly implemented components and through external connections to third party systems:

- **Partner services** enable the integration of service endpoints provided by trading partners over different transports and using document formats. This layer of services provides support for traditional B2B partner integration solutions with outside partners and suppliers. It is the component of the architecture in which the concerns of cross-enterprise interactions are isolated. These services provide the document, protocol, and partner management services required for efficient implementation of business-to-business processes and inter-actions:
  - **Community services** enable the management of the trading community for the trading hub, as well as enabling partner self-management.
• **Document services** enable support for business protocols such as RosettaNet, EDI, and AS1/AS2, as well as associated state management to support conversational processes.

• **Protocol services** provide transport level services, including authentication, document routing, and edge service functions for automated document interchange.

• **Business application services** provide the J2EE run time environment for integration components developed as custom application components coded in Java and running in the application server environment. This layer of services provides the framework and run time operating environment for the management of custom application components developed with J2EE, XML, and Web services programming models. These components provide new functions required to implement fully modern business processes that meet the requirements of today’s business environment. The implementation of these components as services within the WebSphere Integration Reference Architecture enables re-use directly across new solutions. The business application services include functions important to the traditional programmer for building maintainable, flexible, and reusable business logic components, as well as the run time integration to enable higher levels of autonomic administration and management. The service functions reflected in this layer include:

  • **Component services** provide the run time container management services that automatically handles issues such as object persistence, relationship navigation, and object query and transaction management within a J2EE framework.

  • **Core services** provides the run time services, such as memory management, object instantiation and pooling, performance management and load balancing, event notification, availability, directory, and security as part of the J2EE, XML, messaging and Web services programming model.

  • **Interface services** provide the services for bi-directional integration with databases, messaging systems, management frameworks, and enterprise applications.

• **Application and information access** provide the capabilities to interact with third party applications (for example, ERP, CRM), mainframe interfaces (for example, CICS, iSeries), custom applications (via technology bridges such as messaging, application programming interfaces, data handlers), as well as heterogeneous data sources (like RDBMS, XML, non-RDBMS data sources; for example, IMS, text files, and content management systems). This functional layer provides for the access interface to existing applications and data with support for transactional services and connection services for databases, messaging systems, and other data sources. Existing host-based applications and enterprise data are accessible from the ESB through a set of access services. These access services provide the bridging capabilities between legacy applications, pre-packaged applications, enterprise data stores (including relational, hierarchical, and nontraditional, unstructured sources, such as XML, text and content management systems), and the ESB. This layer provides for the integration of mainframe systems through multiple run time solution patterns such as Web facing, communication-level integration, messaging and Web service enabling. As applications and data implementations evolve to become more flexible participants in business processes, enhanced capabilities of their underlying operating environments will continue to increase in utilization. The following services are isolated and enabled in this layer:

  • **Event detect services** provide event notification services based on the event framework that is enabled through the specific application/data source interface. For example,
creation of a new customer in a CRM system would generate an event that the ESB would distribute to the subscriber to this event type.

- **On-ramp services** enable application integration patterns, including one-way inbound, request-reply, and solicit reply message patterns to support application and data wrapper functions, including query execution planning and data retrieval to support data integration. For example, if one step in a business process needs to update the order, a message with the order data would be sent through the ESB to the mainframe CICS application which would then return an acknowledgement.

**Control services**

Control services provide the capabilities to effectively integrate people, processes, and information within the WebSphere Integration Reference Architecture. These services control the flow of interactions and data among people, process, and information services appropriate to the development and run time implementation of business processes:

- **Interaction services** provide the capabilities required to deliver functions and data to end users, meeting the end-user's specific usage preferences. Interaction services also provide the capabilities required to integrate various pervasive devices (sensors and actuators, for example) with the other components of the integrated system. This layer of services provides the external interaction services for uses (including browser and voice interaction) and pervasive device integration. The following services are provided as part of interaction services:
  - **Delivery services** enable a run time interaction framework for users to interact with the integration components via portlets, voice, and pervasive messaging; this set of capabilities includes specialized technologies such as multi-device support, page aggregation, markup transcoding, language translation, and internationalization, as well as integration to wireless communication technologies to support mobile/remote user/device interactions.
  - **Experience services** provide the user-centric services responsible for the delivery of a robust user experience, including personalization, collaboration, authentication, authorization, self-service functions (customization and registration), and single-sign-on functions.
  - **Resource services** provide for the run time management of the interaction components supporting, for example, security and entitlements via components such as pages, themes/skins, principles, and portlets.

- **Process services** provide the control services required to manage the flow and interactions of services that implement business processes. This layer of services provides the ability to broker process execution through the aggregation of integration components to support coarse-grained business functions. Within the WebSphere Integration Reference Architecture, the BPEL4WS standard is used to describe the orchestration of services. The following services are provided within process services:
  - **Choreography services** are composed of other services, and so choreography services provide the ability to execute these other services in a defined sequence and to recover from errors.
• **Transaction services** provide error recovery for both short running and long running processes. An entire process may be transactional (short running). In a long running process, each service is typically treated as a transaction if it changes data.

• **Staff services** enable the integration of people into a process, providing the creation of work items based on rules and information in a staffing directory. It supports task assignment, delegation, and administration in conjunction with interaction services.

• **Information services** provide the capabilities to federate, replicate, query, analyze, and transform data sources which may be structured (RDBMS or other data sources, such as IDMS or VSAM) or non-structured (such as spreadsheets, text files, document formats, like Adobe PDF). This layer of services provides data integration and aggregation for heterogeneous data sources, thereby enabling transparent access to multiple data contexts using advanced techniques for distributed optimization. The following services are part of the information services layer:
  - **Federation services** enable the ability to aggregate data from traditional (such as RDBMS) and non-traditional sources (such as XML data stores, text data, and content stores) while leaving the data under the control of its native store.
  - **Replication services** provide support for automated real-time data synchronization and source/target transformation of structured data sources, enabling locality of access for data access regardless of source implementation.
  - **Transformation services** transform data from source formats into target formats for batch and record-oriented access/transformation.
  - **Search services** enable full query and search integration through conventional (database and structured data sources) and non-conventional (such as PDF, spreadsheets, word processing documents) data sources.

### Development services

The WebSphere Integration Reference Architecture rigorously recognizes the requirement for a comprehensive software development platform as a core competency. It is important that the development platform encompass the entire lifecycle of software development, including requirements analysis, modeling and design, component development, testing, and code maintenance. The tooling must be compatible with the concepts of MDA (model driven architecture) and support use of the emerging best practices evolving via SOMA (service-oriented modeling and architecture) and SODA (service-oriented development of applications) methodologies. These characteristics are incumbent in the IBM Software Development Platform.

At a high level, development services within the WebSphere Integration Reference Architecture enable people to complete specific tasks and create specific output based on their skills, their expertise, and their role within the enterprise:

- **Business Analysts**, who analyze business process requirements, need modeling tools to have business processes be designed and simulated.
- **Software Architects** need tools to enable them to model data, functional flows, and system interactions, and to develop system topologies.
- **Integration Specialists** require capabilities to configure and orchestrate components in the development of integration solutions.
• Programmers need tools to develop new business logic, such as J2EE components, portlets, and other custom service components.

Most importantly, the integration tooling environment promotes joint development, asset management, and deep collaboration between development roles through asset access and asset sharing. It is important to note that an organization's tool technologies and competencies will come from multiple vendors and, as a result, the presence of a multi-vendor framework like Eclipse is an imperative to reduce the learning curve for the disparate roles in the development process. Additionally, a standard tool framework such as Eclipse provides common repository and base functions common across all the developer perspectives (for example, version control functions such as CVS and ClearCase, and utility functions such as edit, file, and print services). The development services provided through the WebSphere Integration Reference Architecture leverage the Eclipse base for their implementation.

Regardless of specific development roles, a high degree of collaboration is required in software development to enable each role to be productive and efficient. The development tool platform provides an integrated set of tools that addresses the scope of integration development through role-based activities and across a multi-vendor tool environment. By applying separation of concerns to the development process, each role can design, develop, and deploy artifacts specific to an individual's skills and responsibilities. A number of service functions are exposed as part of the development framework:

• **Model services** provide the ability for analysts to build visual models that are representative of business processes.
• **Design services** enable models to be staged into design, and include the ability to attribute the process with service components. Additionally, this set of functions enable the design and development of new integration components.
• **Implementation services** enable the ability to move developed artifacts into production as part of an organization's configuration management standards.
• **Test services** provide the ability to support unit test as well as integrated test capabilities as part of the overall development lifecycle.

**Business innovation and optimization services**

In combination with many of the services mention above, business innovation and optimization services provide an infrastructure for continuous improvement and innovation, enabling businesses to adapt to changing market dynamics and everyday operational disruptions. Well-engineered BPM solutions support a holistic approach to business management, enabling aligned objectives, role-based visibility, contextual insight, and in-time actions. It is critical to note that BPM requires a set of differentiated capabilities that supports and incorporates the needs of both business and IT professionals.

The common base event (CBE) specification provides the common base event model for the WebSphere Integration Reference Architecture. CBEs, originally developed by the IBM Autonomic team in collaboration with partners, is an emerging OASIS standard that defines a common XML schema-based representation of events, supporting encoding of logging, tracing, management, and business events.
Within the WebSphere Integration Reference Architecture, the BPM services consist of three primary groups, each of which supports both IT and business events:

- **Common event infrastructure (CEI) services** provide emission services for filtering and converting native events into CBE form, event store services for storing, querying, and managing the events, and event catalog services for storing, querying, and managing the event schemes.
- **Correlation services** provide policy-driven filtering and correlation of events to detect situations of interest.
- **Monitoring services** use an observation model to define the appropriate monitoring context, map events to the context, and compute and manage the associated performance measures and key performance indicators (KPIs).

Presentation of metrics and KPI is done using the interaction services. Likewise, any analytic and data integration that is required to support a BPM solution is provided using the information services.

The linkage between the development platform and the business innovation and optimization services is a key aspect of the WebSphere Integration Reference Architecture. The ability to characterize key performance indicators as part of the modeling environment and to generate specific event flags as part of the process model enable analysts to build management functionality into their business processes. Following implementation of integration components, the BPM layer provides capture and delivery of event data and statistics, which can also be input back into the modeling environment. This approach enables organizations to support iterative process re-engineering through a continuous business process improvement cycle.

**IT services management**

Beneath all the capabilities of the WebSphere Integration Reference Architecture are the management services for security, directory, system management, and resource virtualization. The security and directory services include authentication and authorization functions required for implementing; for example, single sign-on capabilities across a distributed and heterogeneous system. System management and virtualization services include functions across the operating environment for management of server, storage, network, and other resources; for example, clustering and virtualization services enable the efficient use of computing resources based on load patterns and other factors. The ability to leverage grids as part of a grid computing platform is an integral part of IT services management. While many management services perform functions tied directly to hardware or system implementations, others provide functions to interact directly with integration services provided in other elements of the architecture through the ESB. These interactions typically involve services related to security, directory, and IT operational systems management, as part of the support operating environment.

Hardware and software management services provide the capabilities required to effectively run and operate enterprise systems. Many of these services are independent from the other integrated services; others provide capabilities and data to the other integrated services to enable effective business performance management and system operation.
WebSphere Integration Reference Architecture in action

In Figure 3, purchase order processing is an end-to-end composite integration process. The solution represents a sequence of services which are orchestrated through the components in the WebSphere Integration Reference Architecture.

Figure 3. Service invocation in the WebSphere Integration Reference Architecture

The development tools and IT services management are not shown. The development tools and management services provide the support for development of the integration components and the management of the underlying operational runtime framework, respectively. Through an SOA-based approach to business integration, individual integration artifacts can be developed independently and then orchestrated to provide an overall solution. This provides a key differentiator to the traditional development of integration solutions as tightly-bound applications. More importantly, any of the developed components (such as the adapter services or partner profile services) can be re-used without affecting the operation of the purchase order process. The WebSphere Integration Reference Architecture enables organizations to be more strategic in the application of integration techniques to solve the challenges of on-demand computing.

As discussed in Patterns: Service-oriented Architecture and Web Services (M. Endrei and others, 2004), "an on-demand operating environment provides the infrastructure needed to allow applications to be integrated using common standards and open technologies." This is a key tenet of the integration framework enabled via the WebSphere Integration Reference Architecture. The adherence to common standards and open technologies are the foundation of the IBM integration approach. The table below identifies some of the major standards and technologies that are present in the WebSphere Reference Architecture:

Table 1. WebSphere Integration Reference Architecture standards and technologies

<table>
<thead>
<tr>
<th>Service function</th>
<th>Relevant standards</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Enterprise service bus | JMS, J2EE, SOAP, XSLT, WSDL, UDDI
--- | ---
Development tools | Eclipse, J2EE, J2SE, J2ME, XML, UML, Java Server Faces, SWT, XMI, WS BPEL, SQLJ, JDBC, XSLT, WSDL, UDDI
Business performance management tools | W3 Common Log Format, WS DM initiatives, CEI/CBE
Interaction services | WSRP, JSR 168, Java Server Faces, VoiceXML, J2EE
Process services | J2EE, BPEL4WS, WSDL, UDDI
Information services | XQuery, SQL, JDBC/ODBC
Partner services | FTP, sFTP, HTTP, HTTP/S, RosettaNet, SMTP, JMS, SOAP/HTTP, WMQ, cXML, EDI (X12, EDIFACT and others)
Business application services | J2EE (JNDI, EJB, JSP, JTA, JAAS, JAXP, JAXR, JMX and others)
Application and information assets | J2C, JMS, IIOP, JDBC, CICS, IMS, 3270/5250

Additionally, industry standards are reflected within the solution framework, such as ACORD, SWIFT, HIPAA, UCCNET, and other key vertical frameworks and initiatives. The adherence to standards represents IBM's approach to integration as a key enabler in building flexible and sustainable enterprise integration architectures.

IBM's WebSphere Integration Reference Architecture provides the most comprehensive integration framework in the industry. In addition to supporting traditional programming and development solutions, the architecture has extensive support for developing service-oriented solutions for integration. This support for SOA begins with the industry-leading support for MDA in the IBM Software Development Platform, and is then further implemented in the WebSphere foundation technologies within IBM WebSphere Application Server and IBM WebSphere MQ. The overall WebSphere Integration Reference Architecture is tightly tied to existing and emerging standards that support SOA development. Most importantly, the WebSphere Integration Reference Architecture provides support for all major styles of integration, enabling the linkage of people, processes, and information on a standards-based foundation. The solution architecture provides rich support for lifecycle modeling through development and implementation, and through enabling the monitoring of business and IT metrics as part of a complete development platform.

**Conclusion**

The WebSphere Integration Reference Architecture enables an enterprise to form a tight linkage between business requirements and technology solutions through a service-oriented architecture foundation. Applying this architecture and using the concepts of "separation of concerns" provides a clear alternative to traditional integration approaches.

Three key concepts -- MDA, SOA, and BPM -- provide the underpinnings to the WebSphere Integration Reference Architecture. A common framework of role-based development tooling embodying MDA enables the design and development of integration artifacts. These artifacts are tested and deployed into a run time environment with a communication infrastructure provided via an enterprise service bus architecture. These integration components leverage a common set of core infrastructure services (for performance, scalability, security, and so on) based on SOA. Finally, overlaying this run time structure is an extensive monitoring and management environment embodying BPM.
In summary, the WebSphere Integration Reference Architecture provides a complete and comprehensive architecture that covers the breadth of integration needs within an enterprise. Integration services are clearly defined and delivered in a modular way promoting reuse and shared competencies across the organization. As new projects are implemented, services are easily added or extended, enhancing the scope and the efficiency of enterprise integration efforts. The WebSphere Integration Reference Architecture enables organizations to take a service-oriented approach to integration and to avoid the pitfalls associated with traditional integration approaches.

Acknowledgements

The author would like to acknowledge the work of the Worldwide WebSphere Technical Sales team in the creation and evolution of the WebSphere Integration Reference Architecture, and particularly the contributions of Bill Hassell, Bob Liburdi, and Bob Knaus. Additionally, the author would like to acknowledge Dan Wolfson, IBM CTO for Business Integration, for his thorough review of this article.
Related topics


© Copyright IBM Corporation 2005
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)