



## **Enterprise Reference Architecture**

*Prepared by*

*Enterprise Planning and Architecture Strategies Team*

**Control Page:**

<b>Revision History:</b>			
<b>Version No</b>	<b>Revised Date</b>	<b>Author</b>	<b>Comments</b>
	03/18/2011	Anitha Ramakrishnan	Initial Version

**Table of Contents:**

**1 Introduction:..... 4**

    1.1 Purpose: ..... 4

**2 Major Architectural Requirements: ..... 4**

**3 NYSDEPARTMENT OF LABOR SOA Solution:..... 5**

**4 Layers of our SOA Solution (Reference Architecture):..... 6**

    4.1 Consumer Layer:..... 7

    4.2 Services (Interaction) Layer:..... 9

    4.3 Process Services Layer:..... 9

    4.4 Business Services / Components Layer: ..... 10

    4.5 Operational / Persistence Layer: ..... 11

    4.6 Foundational Services: ..... 12

    4.7 Infrastructure Services: ..... 12

    4.8 Integration Services:..... 12

    4.9 Governance: ..... 13

**5 Conceptual Operational Model..... 13**

**6 Logical layers to Operational model mapping ..... 14**

**Consumer Layer: ..... 14**

**Services Layer: ..... 14**

**Process Services Layer: ..... 15**

**Business Services/Components Layer: ..... 15**

**Operational Layer: ..... 15**

**7 Architectural Guidelines ..... 15**

    7.1 Information Architecture: ..... 15

    7.2 Service Exposure: ..... 15

    7.3 Business Rules: ..... 16

    7.4 Synchronous / Asynchronous Invocations: ..... 16

    7.5 General guidelines for Composite Application Design/Development:..... 16

    7.6 Development Tools: ..... 17

**8 Appendix: ..... 18**

    8.1 Examples / Sample Diagrams: ..... 18

    8.2 Abbreviations Used: ..... 19

    8.3 Acronyms Used:..... 19

**9 References:..... 19**

    9.1 Bibliography:..... 19

## 1 Introduction:

Enterprise reference architecture provides a framework or set of guidelines and practices for a technology environment / application development. It serves as the basis for the design and construction of various composite applications in multiple domains within the Department of Labor. Our enterprise architecture is based on Service Oriented Architecture (SOA).

### 1.1 Purpose:

The purpose of the document is to outline enterprise architecture and design of the SOA solution at the Department of Labor. This document defines the different layers of our SOA solution, highlighting each layer's physical and logical aspects such as the architectural building blocks (ABBs), relations between ABBs and layers, interaction patterns, architectural design decisions, technology and product selections.

This document is intended for Business Managers, IT Managers, Enterprise Architects, Application Architects, Technical leads and Developers.

## 2 Major Architectural Requirements:

Following are some of the major architectural requirements that steer decisions outlined in the document.

- Build new componentized, SOA based composite applications that promote better interoperability and reuse of services and components, by integrating with legacy, and Commercial Off-The-Shelf (COTS) systems.
- Support for multiple communication protocols (SOAP/HTTP, SOAP/MQ etc), especially at the edge of the network for external customers.
- Support for reliable messaging with synchronous and asynchronous invocation.
- Aid process oriented programming model with the ability to run micro flows and macro flows (long running processes) and support process integration with service choreography.
- Complex work flow support, scheduling and timing of interactions, event correlation and workflow involving human interaction.
- Transaction support: Some use cases require compensation and rollback of committed service requests (transaction support in composite services). There should also be support for rule-based, atomic and process-driven transactions.

- Ability to integrate COTS products such as xPression, PeopleSoft and IBM Enterprise Content Manager (ECM).
- High performance, scalable applications with minimal latency and fail-over support.
- Centralized policy enforcement: Some services may have to be deployed in multiple copies within the infrastructure to facilitate reliability, serviceability and availability, as internal (Department of Labor staff) and external users could use the same service. This raises the need for centralized policy enforcement and workload management, distributing requests to identical service instances based on policies.
- Need for service level monitoring, auditing for data accessibility, fault detection, root-cause analysis, reporting and historical analysis.
- Support for WS\* standards.
- Message validation (WS-I and XSD) and support for performance / transformation acceleration.
- Securely support SOA interactions with other agencies like DTF, GORR, and DMV etc.
- Provide location transparency, dynamic discovery of services and service endpoints.
- Provide single sign-on from/to external agency applications for external users.
- Provide a single interactive user experience both for internal (Department of Labor staff) and external users (customers) for all applications.

### **3 Department of Labor SOA Solution:**

Our enterprise architecture is based on SOA with a process centric programming model, and Business Process Management (BPM), using an Enterprise Service Bus (ESB) for integration and BPEL for process orchestration. The BPM-SOA combination allows services to be used as reusable components throughout the enterprise that can be orchestrated to support the needs of business processes.

The combination approach enables teams to iteratively design and optimize business processes and implement them based on SOA, so that the processes can be changed

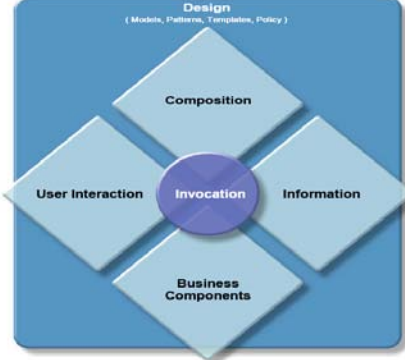
quickly, instead of being 'hard-wired'. In other words, BPM helps in optimizing the business processes and SOA decouples the business process from implementation specifics by providing a layer of control and governance for IT underneath BPM.

The BPM-SOA combination was chosen to meet our objectives and requirements (listed in section 2) proficiently, with ample room to grow.

#### 4 Layers of the SOA Solution (Reference Architecture):

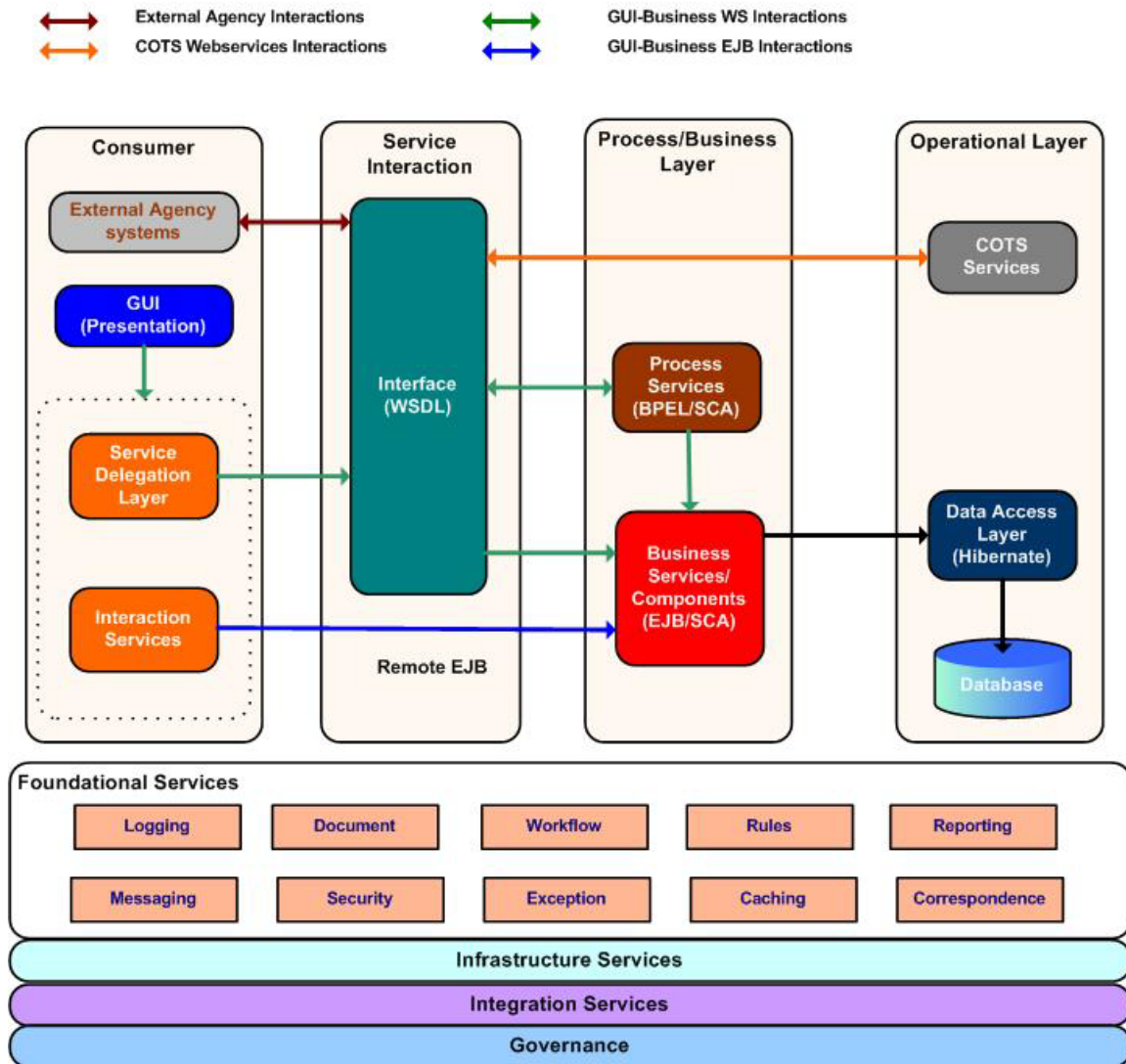
This section defines the logical architectural layers of our enterprise SOA solution and outlines the building blocks, product mappings and major architectural design decisions of each layer. Note that the detailed architecture decisions with different options are captured separately in our consolidated Architecture Decisions document.

These layers were identified considering the following major aspects of application design:

- User Interaction (Presentation) logic
  - Business Logic
  - Information (data) Logic
  - Composition Logic
  - Service Interaction
- 
- the backplane of design and policy metadata from which schema, relationships, data types, and constraints are derived dynamically (**the first five aspects rest on this aspect**)

The following diagram depicts the layers of the Department of Labor's Reference Architecture and includes the consumer/presentation layer, services layer, Process layer, Business services/components layer and the operational / persistence layer.

### Reference Architecture Layers



#### 4.1 Consumer Layer:

This layer is the presentation layer of SOA and it lets users (internal and external) or machines to interact with the Department of Labor’s IT assets (exposed business functions) and provides data to end users to meet specific usage preferences. It enables channel-independent access to those business processes supported by various application and platforms.

Our architectural design decision for this layer is to have a single unified view of knowledge presentation as well as a single unified entry point to the business processes and applications. This unified entry point will be integrated with other foundational services, such as security (single sign-on, access control and trust), to significantly improve the usability of the business process and application. Also, we do not have a portal (product) in this layer. Our decision is to build custom interfaces using JSF and other web 2.0 technologies.

Our consumers are placed into three groups, namely: **internal staff**, **external users** (customers like employers and claimants) and **external systems** from other agencies. These consumers could be using multiple channels such as web, Interactive Voice Response (IVR), batch processes, and external agency systems to access the business services. The presentation layer is responsible to catering to various channels, hence leaving the business layers to be independent of the channels.

The GUI presentation layer includes functionality from basic rendering through to aggregation, visual composition, populating data in GUI elements, addressing navigational aspects of screens etc. The view is the representation of data and should be clearly separated from other layers.

### **Interaction with other layers:**

The presentation layer interacts with the business layers using the following methods.

#### **1) Service Delegation Layer**

The service delegation layer is used when consuming process/business services exposed as web services from the presentation layer. The Service Delegation Layer provides an indirection between GUI and business layers by providing a pattern to invoke web services from the presentation layer in a consistent and abstracted fashion. These services are registered in WSRR and would be accessed through the gateway (router).

#### **2) Interaction Services (EJB 3 invocations)**

The interaction services cater to the GUI intensive processing (based on user actions) that are less linear. These are used in scenarios where the specific collection of data that the user requires is so specific to the GUI that this does not form a re-usable data /data aggregation service. In such scenarios, it is better to interact with business layers directly using context sensitive GUI components/screens.

The interaction services directly invoke business components (remote EJB) to populate data in screens, handling reports/ large volume query, fine-grained database interactions and other GUI intensive interactions that are not dependent on business processes / services. Such a separation will help in abstracting the GUI specific needs (navigation/interaction) from the business components, so that the business services can be built with no knowledge of the GUI that is calling them or its navigational state.

#### **Interaction with other layers:**

**4.2 *These interaction services communicate with the business layers using a façade (session façade) that is responsible for catering to the needs of the GUI. These are implemented using EJB and invoked remotely (remote EJB calls) from the presentation layer.***

#### **4.3 Services (Interaction) Layer:**

This layer consists of all the exposed services defined within our enterprise service portfolio. The definition of each service, which constitutes both its syntactic and semantic information, is defined in this layer by providing a specification for the service consumers with sufficient detail to invoke the functions exposed by a service provider.

The specifications are written using WSDL and XSD and should be platform independent. The specification may also include policy documents (defined using WS-policy), attachments and SOA management descriptions. These services are also registered with WSRR. These services are exposed on the ESB and could be process services, business services/components, COTS services and partner (external agency) services.

#### **Interaction with other layers:**

The consumer layer accesses the process services and other exposed business services through the services layer. Also, the Process layer accesses COTS and partner services through the services layer. The gateway (router) and WSRR plays a major role in this layer by providing location transparency, logging, security and other Quality of Service (QoS) features.

#### **4.4 Process Services Layer:**

The Process Services layer is made of composite services which are comprised from the orchestration of the different business services/components. Process Services are very coarse grained, model the business processes and are implemented using BPEL. Process services would in most cases directly fulfill high level business goals and are coarsely granulated.

These services truly represent the business processes and are independent of any consumers / consuming channels. The Process services orchestrate the lower level business services and components using Service Component Architecture (SCA). These services also maintain state and transactions.

**Interaction with other layers:**

The process services are exposed as web services in the services layer for the consumer layer to consume. The process layer interacts with the Business service/Components layer using SCA.

**4.5 Business Services / Components Layer:**

This layer includes services and components that capture and encapsulate the business logic and associated data (entities) that the business domain / application are dealing with. It would represent the domain model of domain through domain objects along with their state, behavior and associations.

These services deal with managing the lifecycle of the business entities and the business rules associated with them. In other words, it includes different types of services such as Task, Entity and other utility services. These services could be fine grained and will be orchestrated by the Process services.

These services are implemented as EJB / SCA components. These components should be transaction enabled components where it would participate in transactions that are initiated by other higher layers or would have to initiate their own transactions in the absence of them. Components that need to be consumed by process services are exposed as SCA components. Components that need to be consumed by the consumer layer are exposed as web services in the services layer. The components that need to be consumed by interaction services of the consumer layer are built as plain EJB components and are invoked by the presentation layer using remote EJB calls (through a session façade for interaction layer). Interaction services are used to meet very GUI specific needs. The decision could be made by the application architects on a case by case basis.

**Interaction with other layers:**

The process services invoke these services through SCA.

The services of this layer that are exposed as web services are invoked by the consumer layer using the Service Delegation Layer.

The components of this layer that are built for interaction services are invoked as remote EJB calls by the consumer layer.

The services/components of this layer talk to the persistence layer thru O-R mapping (data access layer) for persistence of business objects in database tables.

#### 4.6 Operational / Persistence Layer:

The operational layer includes the Enterprise Information Systems (EIS) / resources of the enterprise. This includes the enterprise database and some of the packaged applications that are built using COTS systems such as PeopleSoft, XPression, ECM and Cognos.

Oracle is the enterprise data store. All enterprise data that are currently distributed in various formats like relational and VSAM and other flat files will be consolidated in a relational format in Oracle.

PeopleSoft is used to build our modernized standalone financial applications.

IBM Enterprise Content Manager (ECM) is a content management tool used to store various forms of content (images, documents etc) in a single repository, such that they can be searched and retrieved easily by authorized system or users. Due to legal and legislation requirements, most of the forms filled by customers (employers & claimants) and correspondences exchanged with customers will have to be stored as documents for future needs.

The packaged applications built using the above mentioned third party products will be service enabled /exposed as web services in the services layer using WebSphere ESB (WESB). Although most of these COTS products offer web services integration out of the box, WESB will be used to mediate and expose them as services in order to abstract product specific interface definitions. It also provides a more generic business interface (by performing XML transformations, if necessary) to access these packaged applications. For example, the data types of the elements of the interface provided by the third party products may not be the same as our enterprise data representation for those elements. WESB will also abstract any changes in packaged applications' web service definitions due to product upgrade (or if the enterprise decides to change the vendor of the product in future). The preferred choice of protocol between the ESB and the operational layer products is SOAP over JMS with WMQ as the JMS provider.

The database resources are accessed through the data access layer (DAL -foundational service), that uses the hibernate Object Relational Mapping (ORM) tool.

#### **Interaction with other layers:**

The business services/components interact with the database using plain java calls (JNDI) through the DAL. The Process services and the consumer layer consume the COTS services through web services. The COTS services are exposed as web services in the Services layer and are registered with WSRR.

#### 4.7 Foundational Services:

The foundational services provide a reusable abstraction of some of the common technical functions that are used / extended to build SOA based composite applications. They establish the standards, patterns, base components and utilities that accelerate the development of business specific implementations. The major advantages of the foundational services include:

- Increased consistency and reusability across all applications
- Enable development teams to focus on business logic rather than the complexities of implementation.
- Provide abstraction from products and technologies used in SOA implementation.

The library of foundational services include the following.

Workflow  
Messaging  
Logging  
Business Rules  
Correspondence  
Security – Identity and Access Management (IAM)  
Exception Handling  
Frameworks and Patterns  
Caching  
Technical components / utility services

There are separate documents for each of these foundational services.

#### 4.8 Infrastructure Services:

As with software, infrastructure is also a service in SOA and provides the foundation for IT services. Major infrastructure services include security / access (authentication and authorization) services, service level management (SLM) services and monitoring services that are shared across the enterprise and are used to optimize throughput, availability and performance.

The runtime product choices in this layer include DataPower, Siteminder (policy server), LDAP, WebSphere Application Server (WAS), WebSphere Process Server (WPS), WMB, WSRR and our EIS systems. ITCAM is used for SLM services.

#### 4.9 Integration Services:

The integration layer is the key enabler of our SOA as it provides the capability to mediate, route and transport service requests from the service consumer to the correct service provider. In other words, it decouples the service consumer and providers. This layer provides the capability for service consumers to locate service providers and initiate service invocations securely.

Our integration layer's building block is the ESB. It, along with WebSphere Service Registry and Repository (WSRR) provides a location- independent mechanism for integration.

Our ESB is a *hybrid* built on an **ESB Gateway Pattern**. The gateway (router) is currently implemented on WebSphere Message Broker (WMB). WESB is used to expose/integrate COTS services. The ESB also provides other features such as logging, synchronous and asynchronous invocation of services, reliable delivery and transaction management.

#### **4.10 SOA Governance:**

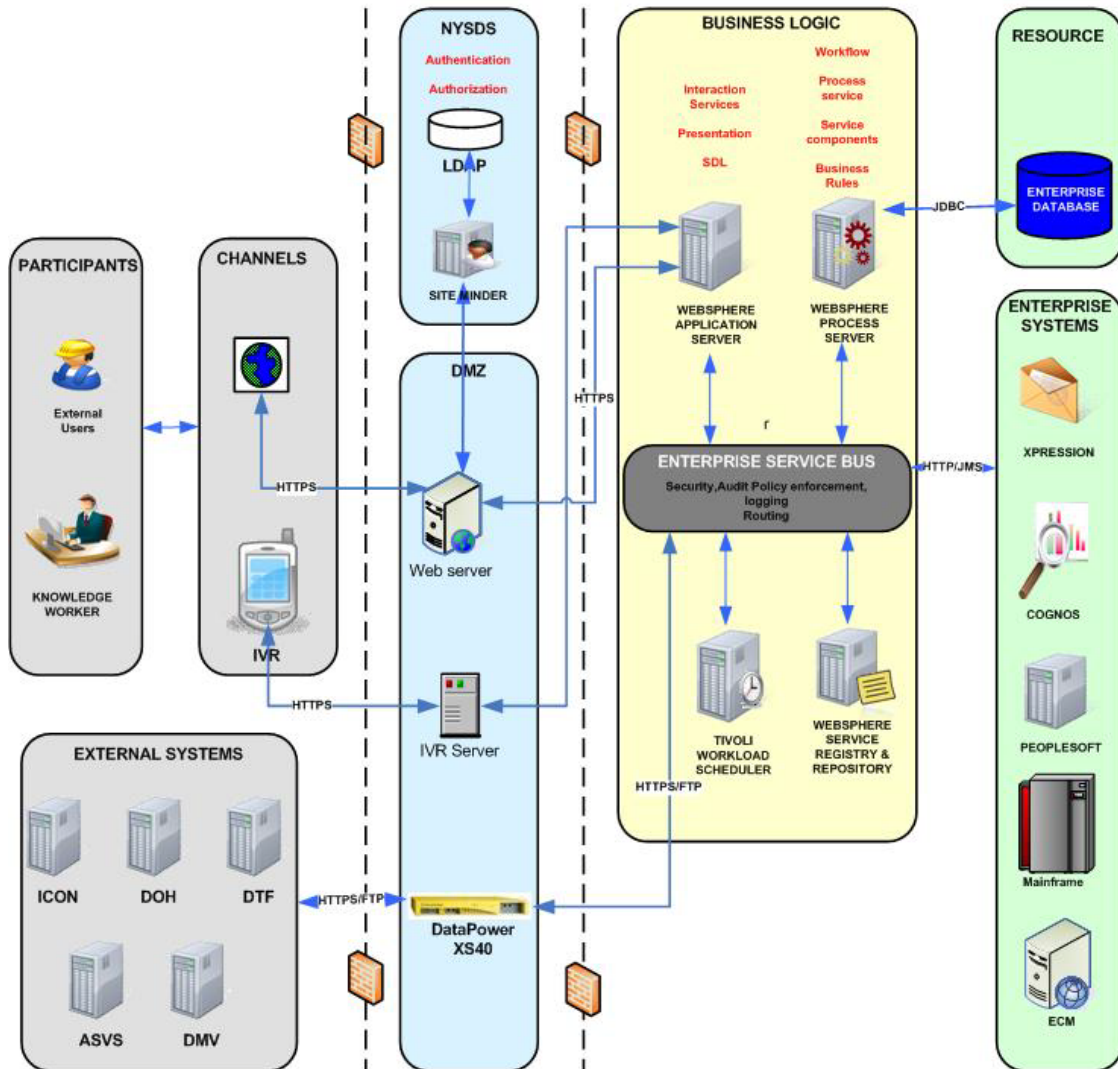
The SOA governance layer covers all aspects of business operational life-cycle management in SOA. It provides guidance and policies for making decisions about an SOA and managing all aspects of an SOA solution, including capacity, performance, security, and monitoring.

This is a very wide layer and involves managing all layers in our SOA solution stack. Service life cycle management is done using WSRR (our enterprise service repository).

## **5 Conceptual Operational Model**

The following diagram depicts the conceptual operational model of the SOA environment.

### Conceptual Operation Model



## 6 Logical layers to Operational model mapping

### Consumer Layer:

The consumer layer’s presentation logic code, Service delegation layer and interaction services (remote EJB invocations code alone, not the actual service component / session facade) are deployed to (WAS) 7.0 (GUI server).

### Services Layer:

This is a logical layer to represent service interactions. This layer includes the router and is currently implanted in WMB.

**Process Services Layer:**

Process Services are implemented using BPEL and are deployed to WPS.

**Business Services/Components Layer:**

Business services are implemented as EJB / SCA components and are deployed to WPS. Note that the actual EJB component that supports the interaction services of the consumer layer are deployed to WPS and invoked through remote EJB calls from the consumer layer. The DAL (foundational service) is also deployed to WPS.

**Operational Layer:**

The database is Oracle 10g. The COTS products based custom applications are deployed to respective servers that host the COTS products. They are integrated using WESB to be exposed as services in the services layer.

**Note : To support EJB 3, WPS is to be upgraded to version 7.0**

## 7 Architectural Guidelines

### 7.1 Information Architecture:

A proper representation of data and information is required in an SOA. The information across our systems is centralized using schema representation (XSD). XSDs are the technical representation of business items and are abstract and independent from technologies. The schema centralization helps in defining a business item in only one representation, no matter how it is stored in the repositories / databases.

### 7.2 Service Exposure:

Following are the general guidelines for exposing a service and the decision could be made on a case by case basis, considering the actual scenario.

- All Process services will be exposed as web services.
- Business services are exposed as SCA components for Process services to consume. (The process service could be from the same business domain or from other domains).
- Business components that need to be consumed by the Consumer layer (of the same domain or another domain) are exposed as web services in the service layer. The exception is the interaction services that are specific to GUI needs and very fine grained services that don't hold any reusable value.
- Business components that need to be shared with other domains are exposed as SCA (for a Process from other domain) and web services (from consumer layer of other domain).
- COTS applications built using third party products and partner services are exposed as web services in Services layer using WESB.

- All exposed services are registered with WSRR and accessed through the router (ESB).

### 7.3 Business Rules:

Business rules cut across all the layers. For example, business process and governance layers intersect in defining the rules and policies for the business process. Consumer layer validation, and input and output transformations from and to that layer, must abide by some rules. These lie at the intersection point between the consumer and governance and policy layer. Our process and service layer business rules are stored as rulesets in Ilog JRules.

### 7.4 Synchronous / Asynchronous Invocations:

Asynchronous invocation is recommended on the edges of modules but usually not within a module / process. Business processes can be either long-running processes or microflows. Following are some guidelines for deciding the type of invocations to use in WPS.

- Whenever possible, use synchronous interactions for non-interruptible processes (microflows). A non-interruptible process is much more efficient than an interruptible process.
- Invocations to request-response operations from a short running process should always be synchronous. All asynchronous invocations for one-way operations are made using the `invokeAsyncWithCallback` invocation style, never asynchronous deferred response.
- Transactions can span across synchronous invocations but cannot span asynchronous invocations.
- The following components/imports are considered "asynchronous":
  - Long running BPEL
  - Human tasks
  - MQ/MQ JMS/Generic JMS/JMS imports
  - Parallel Sub processes / tasks

The consumer layer has a choice of invocation. If the consumer / application expects an immediate response to proceed with the next action, invoke the service synchronously. On the other hand, if the consumer is just submitting a request (e.g. form submission), invoke the service / process asynchronously. The decision should be made based on the requirements that best suits the business scenario.

### 7.5 General guidelines for Composite Application Design/Development:

- Design service interfaces for stateless interactions. The request message passed in to the operation should contain all information necessary to complete that operation, regardless of the sequence in which other interface operations are invoked.

- Build a topology of services that reflects the business processes (business driven) and not the systems in our enterprise.
- Document non functional requirements (like performance, security, availability etc) as well while modeling and analyzing the requirements for building SOA based composite applications. These non functional requirements should be externalized from the actual services implementation and achieved using ESB, DataPower appliances and ITCAM.
- Follow a top down approach in analyzing business processes and identify services to expose the legacy or custom applications based on the need of the processes.
- Avoid implementing any business logic in the ESB. Business logic should be implemented in the business services/components or in the process using BPEL.
- ESB (WESB/ WMB) should log exceptions (if any). All services should throw a predefined custom exception (*DOLCustomException*).
- Analyze existing services before creating new services/components as an already existing service could be reused with no or little modifications. All services identified and developed along with their state of the service should be registered in WSRR.
- Identify business rules and separate them from the process / service logic for easy maintenance as business rules could change over time.
- Follow the standard message format defined by EPAS that includes the *DOLCustomheader* in the SOAP header.
- For web services, use JMS within the network. The edge (DPXS40) will provide multiple protocol options for external agency communication.

**7.6 Development Tools:**

A major portion of an SOA platform is devoted to the tools and services used to develop services as applications, develop compositions of these services, and govern the services architecture they form.

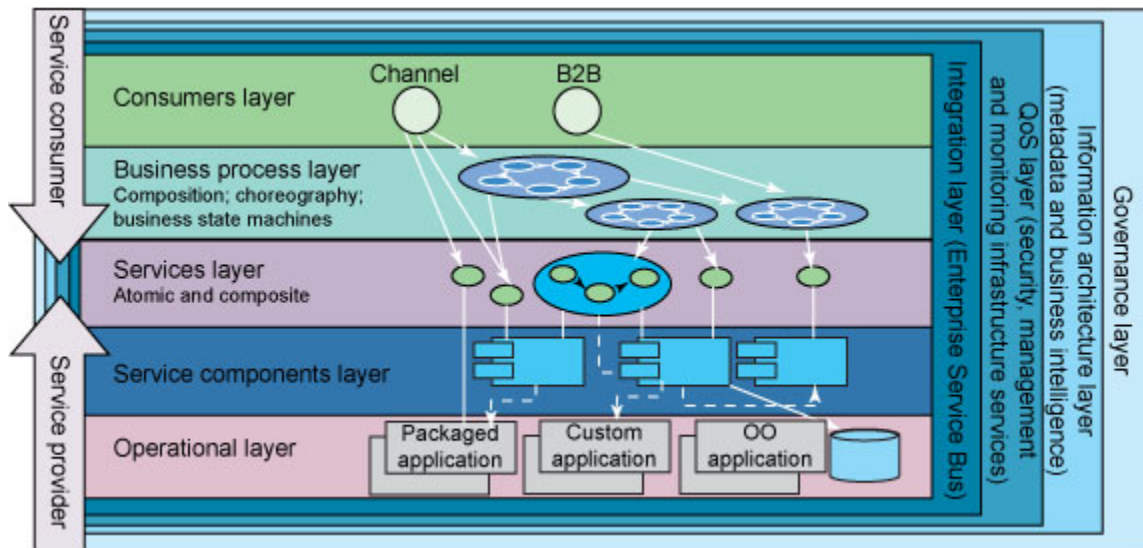
The following is a list of the development tools that could be used during different phases of SOA.

Phase	Tools
Requirements Gathering	Rational RequisitePro Word
Business Modeling	WebSphere Business Modeler WebSphere Publishing Server
Service oriented Analysis and Design	Rational Software Architect Erwin
Service Development	Rational Application Developer (GUI) WebSphere Integration Developer ILog JRules

	Third party development IDEs for xPression, PeopleSoft, and Enterprise Content Manager. COBOL and CICS for legacy
Service Orchestration / Integration	Message Broker Toolkit WebSphere DataPower appliances
Service Choreography	WebSphere Integration Developer ILog JRules
Service Testing	Rational Functional Tester Rational Test Manager
Service Discovery	WebSphere Service Registry and Repository
Service Level Management, and Policy enforcement	ITCAM for SOA WebSphere DataPower appliances WebSphere Service Registry and Repository

**8 Appendix:**

**8.1 Examples / Sample Diagrams:**



**Fig 1. IBM's SOA Reference Architecture**

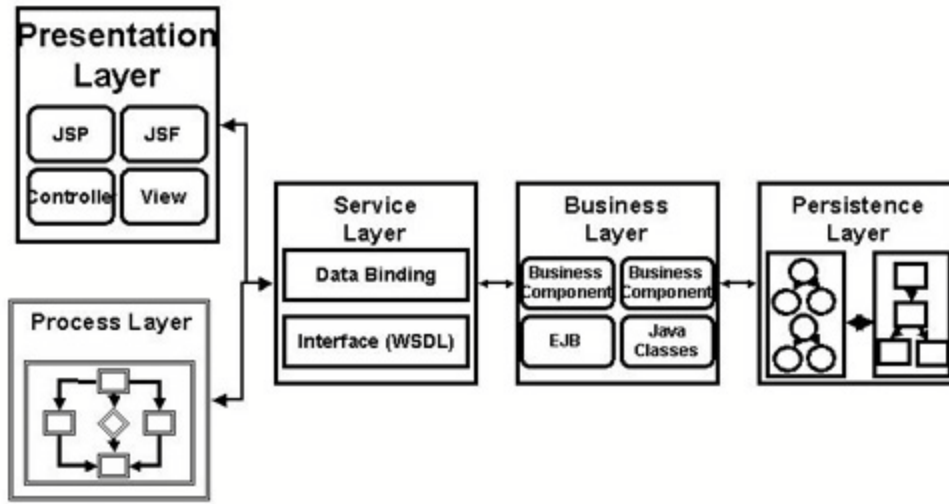


Fig 2. Different Layers of Service Oriented Applications

**8.2 Abbreviations Used:**

None.

**8.3 Acronyms Used:**

COTS	Commercial Off-the-shelf
EIS	Enterprise Information Systems
SDL	Service Delegation Layer
WAS	Websphere Application Server
WESB	Websphere Enterprise Service Bus
WPS	Websphere Process Server
WSDL	Web Services Definition Language
XSD	XML Schema Definition

**9 References:**

None.

**9.1 Bibliography:**

None.