

OBJECTIVES:

- ☐ To gain understanding of the basic principles of service orientation
- ☐ To learn service oriented analysis techniques
- ☐ To learn technology underlying the service design
- ☐ To learn advanced concepts such as service composition, orchestration and Choreography
- ☐ To know about various WS-* specification standards

UNIT I**9**

Roots of SOA – Characteristics of SOA - Comparing SOA to client-server and distributed internet architectures – Anatomy of SOA- How components in an SOA interrelate - Principles of service orientation.

UNIT II**9**

Web services – Service descriptions – Messaging with SOAP –Message exchange Patterns – Coordination –Atomic Transactions – Business activities – Orchestration – Choreography - Service layer abstraction – Application Service Layer – Business Service Layer – Orchestration Service Layer.

UNIT III**9**

Service oriented analysis – Business-centric SOA – Deriving business services- service modeling - Service Oriented Design – WSDL basics – SOAP basics – SOA composition guidelines – Entity-centric business service design – Application service design – Taskcentric business service design.

UNIT IV**9**

SOA platform basics – SOA support in J2EE – Java API for XML-based web services (JAX-WS) - Java architecture for XML binding (JAXB) – Java API for XML Registries (JAXR) - Java API for XML based RPC (JAX-RPC)- Web Services Interoperability Technologies (WSIT) - SOA support in .NET – Common Language Runtime - ASP.NET web forms – ASP.NET web services – Web Services Enhancements (WSE).

UNIT V**9**

WS-BPEL basics – WS-Coordination overview - WS-Choreography, WS-Policy, WSSecurity.

TOTAL = 45 PERIODS**TEXT BOOKS:**

1. Thomas Erl, “Service-Oriented Architecture: Concepts, Technology, and Design”, Pearson Education, 2005.

REFERENCES:

1. Thomas Erl, “SOA Principles of Service Design “(The Prentice Hall Service-Oriented Computing Series from Thomas Erl), 2005.
2. Newcomer, Lomow, “Understanding SOA with Web Services”, Pearson Education, 2005.
3. Sandeep Chatterjee, James Webber, “Developing Enterprise Web Services, An Architect’s Guide”, Pearson Education, 2005.
4. Dan Woods and Thomas Mattern, “Enterprise SOA Designing IT for Business Innovation” O’REILLY, First Edition, 2006

UNIT - I

1. What is architecture?

Application architecture is to an application development team what a blueprint is to a team of construction workers. Different organizations document different levels of application architecture.

2. Define Service-oriented architecture

An SOA can refer to an application architecture or the approach used to standardize technical architecture across the enterprise.

3. Define Client / Server architecture

Mainframe back-ends served thin clients, are considered an implementation of the single-tier client-server architecture. Mainframe systems natively supported both synchronous and asynchronous communication.

The latter approach was used primarily to allow the server to continuously receive characters from the terminal in response to individual key-strokes. Only upon certain conditions would the server actually respond.

4. Define Distributed Internet architecture

Distributing application logic among multiple components (some residing on the client, others on the server) reduced deployment headaches by centralizing a greater amount of the logic on servers. Server-side components, now located on dedicated application servers, would then share and manage pools of database connections, alleviating the burden of concurrent usage on the database server. A single connection could easily facilitate multiple users.

5. Define SOA Characteristics

- Services are discoverable and dynamically bound.
- Services are self-contained and modular.
- Services stress interoperability.
- Services are loosely coupled.
- Services have a network-addressable interface.
- Services have coarse-grained interfaces.
- Services are location-transparent.
- Services are composable.
- Service-oriented architecture supports self-healing.
-

6. Define Coarse-Grained Services

A service-based system controls the network access to the objects within the service through a set of coarse-grained interfaces. A service may still be implemented as a set of fine-grained objects, but the objects themselves are not accessible over a network.

connection. A service implemented as objects has one or more coarse-grained objects that act as distributed façades.

These objects are accessible over the network and provide access to the internal object state from external consumers of the service. However, objects internal to the service communicate directly with each other within a single machine, not across a network connection.

7. Define Service Component

This is the true heart of the SOA. The Service Component is that logical unit of code which implements the functionality to support the Service. The Service Component exposes one or more Services. A Service Component is also usually associated with a data store of some kind. This can contain data about a fundamental data type, control data, process, data, etc depending on the nature of the particular service component.

8. Define Service-component-level Testing

Service-component-level testing or Unit testing, is normally performed by the developers to test that the code not only successfully compiles, but the basic functionality of the components and functions within a service are working as specified.

The primary goal of Component testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each Component is tested separately before integrating it into a service or services.

9. Define Process/Orchestration-level Testing

Process/Orchestration testing ensures services are operating collectively as specified. This phase of testing would cover business logic, sequencing, exception handling and process decomposition (including service and process reuse).

10. Define Security Testing

As SOA evolves and grows within your organization, the profile and necessity of Security testing will increase. Today, many organizations perform an inadequate amount of penetration testing at the very end of a project. SOA combined with Government and Regulatory compliance, will require Security testing activities to be incorporated into the entire project life cycle.

11. Explain Legacy System Adapter

It is my personal opinion that any time a Service Component needs to interface to a legacy system, that an Adapter Pattern should be used. This Adapter performs a number of tasks. Its primary function is to convert to/from data formats spoken by the legacy system and the common data formats used by the Foundational Service Components. In order to perform the conversions, it may be necessary for an adapter to interface with many Service Components in order to perform data enrichment or break data apart into its fundamental data types.

12. Define Composite Service Component

A Composite Service Component is a component which combines the functionality of one or more other Foundational or Composite Service Components. It may also encapsulate additional functional and data enrichment, business process, etc. In some cases, a Composite Service Component may be purpose built to support one and only one business process or application, where it makes sense to encapsulate a piece of reusable functionality on the server side instead of in the application.

13. Define WSDL?

SOA services have self-describing interfaces in platform-independent XML documents. Web Services Description Language (WSDL) is the standard used to describe the services.

14. Define XSD?

SOA services communicate with messages formally defined via XML Schema (also called XSD). Communication among consumers and providers or services typically happens in heterogeneous environments, with little or no knowledge about the provider. Messages between services can be viewed as key business documents processed in an enterprise.

15. Define UDDI?

SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Definition, and Integration (UDDI) is the standard used for service registry.

16. Explain about QOS?

Each SOA service has a quality of service (QoS) associated with it. Some of the key QoS elements are security requirements, such as authentication and authorization, reliable messaging, and policies regarding who can invoke services.

17. Define Service Proxy?

The service provider supplies a service proxy to the service consumer. The service consumer executes the request by calling an API function on the proxy. It then formats the request message and executes the request on behalf of the consumer. The service proxy is a convenience entity for the service consumer. It is not required; the service consumer developer could write the necessary software for accessing the service directly.

18. List the Principles of service orientation

- Standardized Service Contracts
- Service Loose Coupling
- Service Abstraction
- Service Reusability
- Service Autonomy
- Service Statelessness

Service Discoverability
Service Composability
Service-Oriented and Interoperability

19. Define Service-Oriented and Interoperability

Service-oriented computing, stating that services must be interoperable is just about as evident as stating that services must exist. Each of the eight principles supports or contributes to interoperability in some manner.

20. Define service Loose coupling

The principle of Service Loose Coupling promotes the independent design and evolution of a service's logic and implementation while still guaranteeing baseline interoperability with consumers that have come to rely on the service's capabilities.

downloaded from rejinpaul.com

UNIT - II

1. What is Web Services?

Web Services can convert your applications into Web-applications. Web Services are published, found, and used through the Web.

2. What are Web Services?

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- XML is the basis for Web services

3. List the Web services platform elements:

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

4. Define Web API

Web API is a development in Web services (in a movement called Web 2.0) where emphasis has been moving away from SOAP based services towards Representational State Transfer (REST) based communications.^[3] REST services do not require XML, SOAP, or WSDL service-API definitions.

Web APIs allow the combination of multiple Web services into new applications known as mashups.

5. Define Agents and Services

A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided. To illustrate this distinction, you might implement a particular Web service using one agent one day, and a different agent the next day with the same functionality. Although the agent may have changed, the Web service remains the same.

6. Define Soap message.

A SOAP message is specified as an XML Information Set . While all SOAP message examples in this document are shown using XML 1.0 syntax, other representations MAY be used to transmit SOAP messages between nodes.

7. Define Request-Response

Request-Response is a pattern in which the service consumer uses configured client software to issue an invocation request to a service provided by the service provider. The request results in an optional response.

8. Define Subscribe-Push

A third pattern for interaction is called Subscribe-Push, shown in Figure 4-3. In this pattern, one or more clients register subscriptions with a service to receive messages based on some criteria.

Regardless of the criteria, the externally visible pattern remains the same. Subscriptions may remain in effect over long periods before being canceled or revoked. A subscription may, in some cases, also register another service endpoint to receive notifications.

9. Data paging

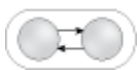
Some services automatically facilitate the paging of large data sets, enabling developers to focus on core application business logic instead of worrying about basic data management infrastructure.

10. Define Coordination Model

The Coordination model provides integrated service to each key customer group. The integration results from sharing key data across the business units to present a common fact to the customer.

11. Define Atomic Service Transaction

Atomic Service Transaction is the name of a design pattern authored by Thomas Erl and published as part of the SOA Design Patterns catalog. Within the catalog this pattern is further categorized as one of the Composition Implementation Patterns. The icon used to identify Atomic Service Transaction is:



12. List the Atomic Transaction Protocols

A Completion protocol which is typically used to initiate the commit or abort states of the transaction.

The Durable 2PC protocol for which services representing permanent data repositories should register.

The Volatile 2PC protocol to be used by services managing non-persistent (temporary) data.

13. Define atomic transaction coordinator

The atomic transaction coordinator plays a key role in managing the participants of the transaction process, and in deciding the transaction's ultimate outcome.

14. Define Business activity

Business activities consist of long-running, complex transactions involving numerous services. Hours, days, or even weeks can pass before a business activity is able to complete. During this period, the activity can perform numerous tasks that involve many participants.

15. Define Orchestration

Refers to an executable business process that may interact with both internal and external Web services. Orchestration describes how Web services can interact at the message level, including the business logic and execution order of the interactions. These interactions may span applications and/or organizations, and result in a long-lived, transactional process. With orchestration, the process is always controlled from the perspective of one of the business parties.

16. Define Choreography

More collaborative in nature, where each party involved in the process describes the part they play in the interaction. Choreography tracks the sequence of messages that may involve multiple parties and multiple sources. It is associated with the public message exchanges that occur between multiple Web services.

17. Define Service Layers

When building various types of services, it becomes evident that they can be categorized depending on:

- the type of logic they encapsulate
- the extent of reuse potential this logic has
- how this logic relates to existing domains within the enterprise

As a result, there are three common service classifications that represent the primary service models used in SOA projects:

- Entity Services
- Task Services
- Utility Services

18. Define Application Services layer.

Application services layer holds applications exposed as services, newly added services, and legacy applications wrapped by standard Web services interface. Services at Application Services layer are set of stateless Web services that perform certain task(s). Business process is the summation of tasks performed by one or more services of application services layer at the sequence maintained by orchestration and

choreography engines. Services of Application Services layer are reusable among different business processes, can be integrated in new applications, and can be extended address new business processes.

19. Define Business layer

Business layer is responsible for supporting business process life cycle. Business process lifecycle consists of five stages: design, model, simulate, monitor, manage, and optimize business processes. Business layer users are either business analysts, or business managers.

Business analysts create the initial drafts of the business processes, and business managers will manage and monitor those business processes.

20. Define Business services layer

Business services layer holds orchestration and choreography engines under governance mechanisms to map business processes to composing Web services. Orchestration and choreography engines are the mapping enablers of business processes into executable services.

Web services are stateless services that can not maintain business logic, operation flow, or user state; so, the need of an orchestration layer to include business logic is addressed. Orchestration and choreography engines maintain business process workflow logic, performance requirements, and system/user state. Business services layer has access to business rules repository.

21. Define Orchestration service

“Orchestration services encapsulate the entire business process. For example, a service containing the entire flow of the Employer Registration" business process is an orchestration service. The complete registration process identifying the employer type, determining liability and registering the employer in the database and various other steps.

UNIT - III

1. What is Service-oriented analysis?

Service-oriented analysis establishes a formal analysis process completed jointly by business analysts and technology architects. Service modeling, a sub-process of service-oriented analysis, produces conceptual service definitions called service candidates. Iterations through the service-oriented analysis and service modeling processes result in the gradual creation of a service inventory blueprint. What are Web Services?

2. Write the business-centric entry points?

These business-centric entry points are:

People—Productivity through people collaboration

Process—Business process management for continuous innovation

Information—Delivering information as a service.

3. Write the IT-centric entry points?

These IT-centric entry points are:

Reuse—Creating reusable functionality

Connectivity—Underlying connectivity to support business-centric SOA

4. Define *entity-centric business service*.

The entity-centric business service is agnostic to any one business process, they achieve "process logic independence" and therefore become highly reusable. And, because they represent a well-defined set of logic and data, they establish a level of abstraction and governance over a distinct business domain. This can significantly increase the agility with which business processes that rely on the composition of services can be altered in response to change."

5. List out the component specification in Service modeling

Data

Rules

Services

Configurable profile

Variations

6. Define SOMA - Service Oriented Modeling and Architecture

Service Oriented Modeling and Architecture refers to the general domain of service modeling that is necessary for the creation and design of Service Oriented Architecture. Service Oriented Modeling and Architecture covers a much larger scope and implements service oriented analysis and design via a specification, designation, and actualization of services, as well as the components that help realize those services and the flows that can aid in the process of building such services.

7. Define Resource and services

The XML Web services architecture defines a standard mechanism for making resources available via XML messaging. Being able to access a resource by simply transmitting XML messages over standard protocols like TCP, HTTP, or SMTP greatly

lowers the bar for potential consumers. The term "Web service" (or simply "service") typically refers to the piece of code implementing the XML interface to a resource, which may otherwise be difficult to access (see Figure 1).

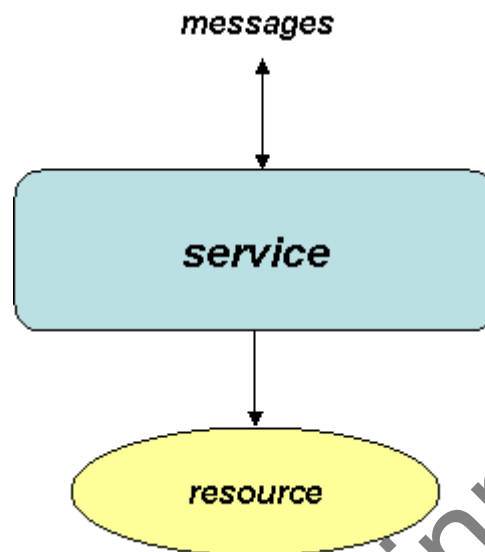


Figure : Resources and services

8. Define Messages and operations

A message exchange is also referred to as an operation. Operations are what consumers care about most since they're the focal point of interacting with the service.

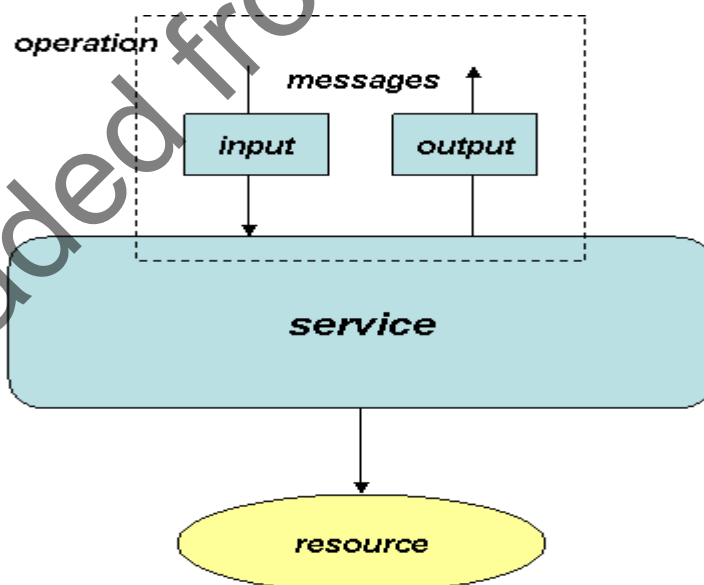


Figure: Messages and operations

A message exchange is also referred to as an operation. Operations are what consumers care about most since they're the focal point of interacting with the service.

9. Define WSDL

The Web Services Description Language (WSDL) provides an XML grammar for describing these details. WSDL picks up where XML Schema left off by providing a way to group messages into operations and operations into interfaces. It also provides a way to define bindings for each interface and protocol combination along with the endpoint address for each one. WSDL plays an important role in the overall Web services architecture since it describes the complete contract for application communication. Although other techniques exist for describing Web services, the WS-I Basic Profile Version 1.0 mandates the use of WSDL and XML Schema for describing Web services. This helps ensure interoperability at the service description layer.

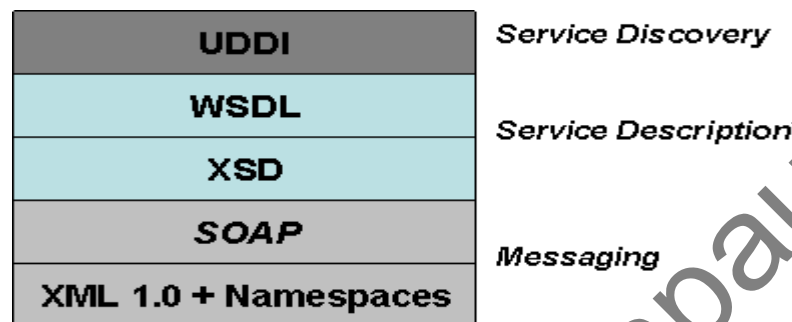


Figure: WS-I Basic Profile 1.0 technologies

10. Define SOAP

SOAP is a method of accessing remote objects by sending XML messages, which provides platform and language independence. It works over various low-level communications protocols, but the most common is HTTP.

11. Explain about the operations in entity-centric?

GetSomething
UpdateSomething
AddSomething
DeleteSomething

12. What is the responsibility of the service?

This provides the basic description of WHY the service should exist. Services are expensive. If you cannot stand in front of an executive and make a purely business-oriented case for the existence of a service, you need to rethink it. You are doing something wrong.

13. What rules does the service own?

This provides the basic scope of the service. In a well devised, Enterprise SOA, you will have a rule implemented in a relatively small number of services (hopefully in one service) which makes it easier to change that rule. This means that you need to describe the collection of rules owned by a service.

14. What style of EAI are you implementing?

The alternative is the "RPC (Remote Procedure Call)" style. If you are passing a block of self-describing and complete data to a service, and all that returns is "thanks... got it," then you are using the messaging style. If you are passing a command (with or without parameters) and are expecting either a set of data in response or an "OK... operation complete" message back, then you are using the RPC style. Note that RPC style services are more typical but, IMHO, less powerful because they assume a real-time binding between the interacting systems.

15. Is the service idempotent?

In other words, if a call to the service is duplicated, and instead of being called once, the service is called twice with identical parameters/payload, will the service detect the duplication and prevent any effects on the underlying systems? This is very important in messaging style services, but it turns up in RPC services as well. A service that provides idempotency is more loosely coupled than one that does not, but it also adds to the complexity of the service implementation.

16. What preconditions and postconditions apply to this service?

Just as in use case development, you must be able to describe the factors that must be in place for this service to be used. Unlike typical use case development, however, you must describe the behavior of the service when these preconditions are not met. You also must describe the limitations and constraints of the service. For example, if a service is designed to be used only during a specific business process, then this must be described and included in the service design.

17. What actors may use this service and how will they be authenticated?

This is an optimistic statement, because you (a) may not know, and (b) may not want to limit your implementation. However, you need to consider all of the actors who can use the business rule that you are encapsulating. If one of those actors cannot use your service, you need to either find a suitable interaction where that actor can use the rule, or create another service that meets that actors needs. Even within the firewalls of the data center, it is imperative that the communications between systems be understood to be secure from mal-intentioned people. If your answer is "pray," then you may want to consider a new line of work.

18. What data elements will be required in order to call the service?

Do not define the format of the calling sequence. Define the semantics of the data element itself. Is there an Enterprise-wide Unique identifier for the data item you are submitting? If there is a numeric value, what does it mean? At what point in a process is it meaningful?

19. What data elements will be returned by the service in its acknowledgement / receipt / return?

Think of these questions as the services "data dictionary" but with more constraints. Data dictionaries describe data at rest. These points describe data in process.

20. Define task service

A *task service* is a form of business service with a functional context based on a specific business process. As a result, task services are not generally considered agnostic and therefore have less reuse potential than other service models.

downloaded from rejinpaul.com

UNIT - IV

1. Write the Basic Platform Blocks?

- Development environment
- Runtime
- APIs
- Operating system

2. Write the layers required by a development and run-time platform for building SOA?

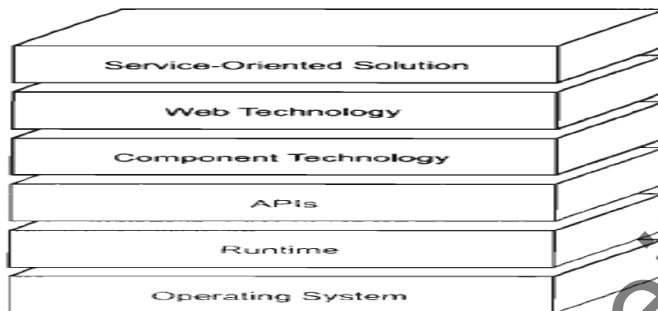


Figure 18.2
The common layers required by a development and runtime platform for building SOA.

3. Define WS-Coordination framework

The WS-Coordination framework exposes an Activation Service which supports the creation of coordinators for specific protocols and their associated contexts. The process of invoking

4. Define Service Processing Logic.

Message Processing Logic The part of a Web service and its surrounding environment that executes a variety of SOAP message processing tasks. Message processing logic is performed by a combination of runtime services, service agents, as well as service logic related to the processing of the WSDL definition.

Business Logic The back-end part of a Web service that performs tasks in response to the receipt of SOAP message contents. Business logic is application-specific and can range dramatically in scope, depending on the functionality exposed by the WSDL definition. For example, business logic can consist of a single component providing service-specific functions, or it can be represented by a legacy application that offers only some of its functions via the Web service.

5. Define business logic

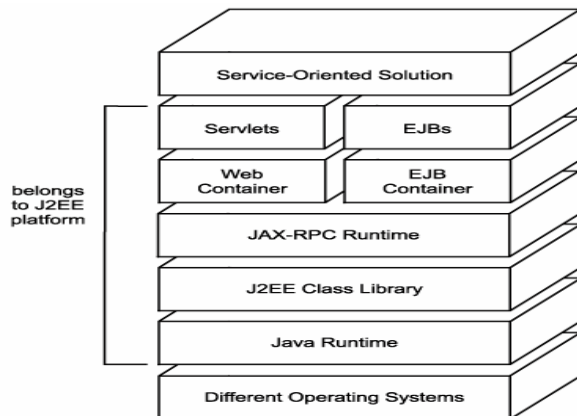
Business logic can exist as a standalone component, housing the intelligence required to either invoke a service provider as part of a business activity or to respond to a request in order to participate in such an activity.

As an independent unit of logic, it is free to act in different roles.

6. Define Service agents

A type of software program commonly found within the message processing logic of SOA platforms is the *service agent*. Its primary role is to perform some form of automated processing prior to the transmission and receipt of SOAP messages. As such, service agents are a form of intermediary service.

7. List the layers of the J2EE platform as they relate to SOA.



8. List the layers of the J2EE platform as they relate to SOA.

J2EE solutions inherently are distributed and therefore componentized. The following types of components can be used to build J2EE Web applications:

- *Java Server Pages (JSPs)* Dynamically generated Web pages hosted by the Web server. JSPs exist as text files comprised of code interspersed with HTML.
- *Struts* An extension to J2EE that allows for the development of Web applications with sophisticated user -interfaces and navigation.
- *Java Servlets* These components also reside on the Web server and are used to process HTTP request and response exchanges. Unlike JSPs, servlets are compiled programs.
- *Enterprise JavaBeans (EJBs)* The business components that perform the bulk of the processing within enterprise solution environments. They are deployed on dedicated application servers and can therefore leverage middleware features, such as transaction support.

While the first two components are of more relevance to establishing the presentation layer of a service-oriented solution, the latter two commonly are used to realize Web services.

9. Define EJB Container and Web Container.

- *EJB container* This container is designed specifically to host EJB components, and it provides a series of enterprise-level services that can be used collectively by EJBs participating in the distributed execution of a business task. Examples of these services include transaction management, concurrency management, operation-level security, and object pooling.
- *Web container* A Web container can be considered an extension to a Web server and is used to host Java Web applications consisting of JSP or Java servlet components. Web containers provide runtime services geared toward the processing of JSP requests and servlet instances.

10. List Support for service-orientation principles

Autonomy
Reusability
Statelessness
Discoverability

11. Define JAX-WS.

JAX-WS stands for Java API for XML Web Services. JAX-WS is a technology for building web services and clients that communicate using XML. JAX-WS allows developers to write message-oriented as well as RPC-oriented web services.

In JAX-WS, a web service operation invocation is represented by an XML-based protocol such as SOAP. The SOAP specification defines the envelope structure, encoding rules, and conventions for representing web service invocations and responses. These calls and responses are transmitted as SOAP messages (XML files) over HTTP.

12. Define Service endpoint interface

A service endpoint interface or service endpoint implementation (SEI) is a Java interface or class, respectively, that declares the methods that a client can invoke on the service. An interface is not required when building a JAX-WS endpoint. The web service implementation class implicitly defines an SEI.

13. Define JAXB

The Java Architecture for XML Binding (JAXB) provides a fast and convenient way to bind between XML schemas and Java representations, making it easy for Java developers to incorporate XML data and processing functions in Java applications. As part of this process, JAXB provides methods for unmarshalling XML instance documents into Java content trees, and then marshalling Java content trees back into XML instance documents. JAXB also provides a way to generate XML schema from Java objects.

14. Define Marshalling and Unmarshalling

Marshalling provides a client application the ability to convert a JAXB-derived Java object tree back into XML data.

Unmarshalling provides a client application the ability to convert XML data into JAXB-derived Java objects.

15. Define Schema-to-Java

Custom JAXB binding declarations allow you to customize your generated JAXB classes beyond the XML-specific constraints in an XML schema to include Java-specific refinements, such as class and package name mappings.

JAXB provides two ways to customize an XML schema:

- As inline annotations in a source XML schema
- As declarations in an external binding customization file that is passed to the JAXB binding compiler

Code examples that show how to customize JAXB bindings are provided later in this chapter.

16. Define Java-to-Schema

The JAXB annotations defined in the `javax.xml.bind.annotations` package can be used to customize Java program elements to XML schema mapping. [Table 17-3](#) summarizes the JAXB annotations that can be used with a Java package.

17. Define Scope, Inheritance, and Precedence

Default JAXB bindings can be customized or overridden at four different levels, or scopes.

The fig illustrates the inheritance and precedence of customization declarations. Specifically, declarations towards the top of the pyramid inherit and supersede declarations below them. For example, Component declarations inherit from and supersede Definition declarations; Definition declarations inherit and supersede Schema declarations; and Schema declarations inherit and supersede Global declarations.

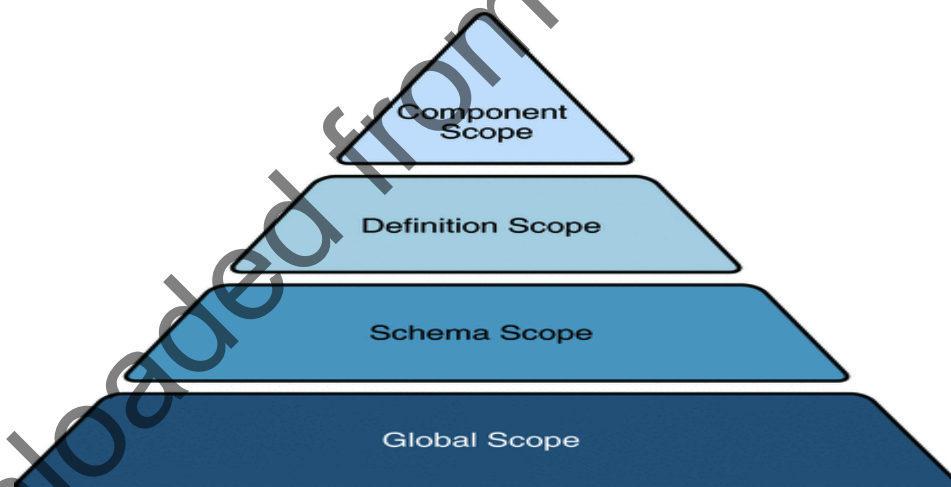


Figure : Customization Scope Inheritance and Precedence

18. Define JAXR

The Java API for XML Registries (JAXR) provides a uniform and standard Java API for accessing various kinds of XML registries.

After providing a brief overview of JAXR, this chapter describes how to implement a JAXR client to publish an organization and its web services to a registry and to query a registry to find organizations and services. Finally, it explains how to run the examples provided with this tutorial and offers links to more information on JAXR.

19. What Is a Registry?

An XML *registry* is an infrastructure that enables the building, deployment, and discovery of web services. It is a neutral third party that facilitates dynamic and loosely coupled business-to-business (B2B) interactions. A registry is available to organizations as a shared resource, often in the form of a web-based service.

Currently there are a variety of specifications for XML registries. These include

- The ebXML Registry and Repository standard, which is sponsored by the Organization for the Advancement of Structured Information Standards (OASIS) and the United Nations Centre for the Facilitation of Procedures and Practices in Administration, Commerce and Transport (U.N./CEFACT)
- The Universal Description, Discovery, and Integration (UDDI) project, which is being developed by a vendor consortium

A *registry provider* is an implementation of a business registry that conforms to a specification for XML registries.

20. Define Querying a Registry

The simplest way for a client to use a registry is to query it for information about the organizations that have submitted data to it. The BusinessQueryManager interface supports a number of find methods that allow clients to search for data using the JAXR information model. Many of these methods return a BulkResponse (a collection of objects) that meets a set of criteria specified in the method arguments. The most useful of these methods are as follows:

- findOrganizations, which returns a list of organizations that meet the specified criteria--often a name pattern or a classification within a classification scheme
- findServices, which returns a set of services offered by a specified organization
- findServiceBindings, which returns the *service bindings* (information about how to access the service) that are supported by a specified service

21. List the benefits of JAX_RPC

- Portable and interoperable web services
- Ease of development of web service endpoints and clients
- Increased developer productivity
- Support for open standards: XML, SOAP, WSDL
- Standard API developed under Java Community Process (JCP)
- Support for tools
- RPC programming model with support for attachments
- Support for SOAP message processing model and extensions
- Secure web services
- Extensible type mapping

22. Define WSIT

WSIT is an implementation of a number of open web services specifications to support enterprise features. In addition to message optimization, reliable messaging, and security, WSIT includes a bootstrapping and configuration technology. Starting with

the core XML support currently built into the Java platform, WSIT uses or extends existing features and adds new support for interoperable web services, including:

- Bootstrapping and Configuration
- Message Optimization Technology
- Reliable Messaging Technology
- Security Technology

23. List the layers of the J2EE platform as they relate to SOA.

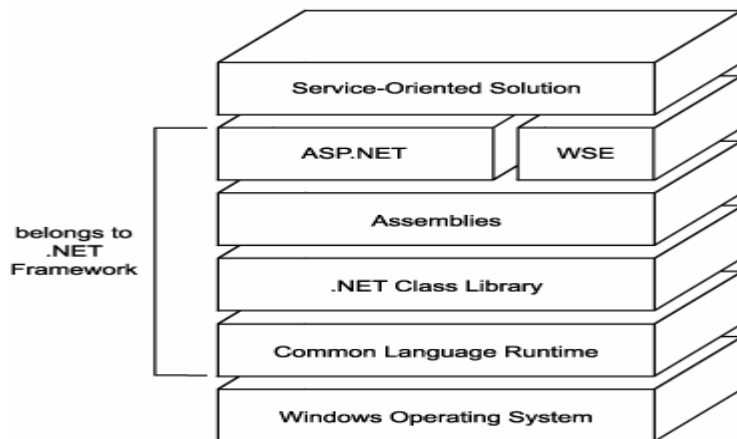


Figure: Relevant layers of the .NET framework, as they relate to SOA.

24. Define CLR.

The *Common Language Runtime (CLR)* provided by the .NET framework. CLR supplies a collection of runtime agents that provide a number of services for managing .NET applications, including cross-language support, central data typing, and object lifecycle and memory management.

25. Write the features of CLR.

- Cross-language integration, especially cross-language inheritance.
- Garbage collection, which manages object lifetime so that reference counting is unnecessary.
- Self-describing objects, which make using Interface Definition Language (IDL) unnecessary.
- The ability to compile once and run on any CPU and operating system that supports the runtime.

26. Define ASP.NET Web Forms.

All server controls must appear within a <form> tag, and the <form> tag must contain the runat="server" attribute.

27. Define Web Services Enhancement.

The Web Services Enhancements (WSE) 3.0 for Microsoft® .NET is an add-on to Microsoft Visual Studio® 2005 and the Microsoft .NET Framework 2.0 that enables developers to build secure Web services based on the latest Web services protocol specifications.

UNIT - V

1. List out the stages to develop applications in WS-BPEL

<i>Denial</i>	- don't need it
<i>Coercion</i>	- management says we have to use it
<i>Elation</i>	- realization that it will solve all our enterprise application problems
<i>Depression</i>	- realization that it will <i>not</i> solve all our enterprise application problems... yet
<i>Enlightenment</i>	- understanding how - and when - to leverage the advantages of SOA, via BPEL (in this case, using your Java skills as a guide)

2. Service Orchestration

Service Orchestration is nothing more than a fancy title for a program that calls web services during its execution. There's nothing magical or omnipotent about a "service orchestration". In fact we'll learn later that any service orchestration implemented using BPEL is *also* just a standard, WSDL-defined web service in its own right.

In the context of Java, a service orchestration might be a Java class that not only constructs and calls methods on other classes but which *also* invokes web services using special Java classes and interfaces designed specifically for that purpose. \

3. Define the Reply activity.

A Reply is responsible for extracting data from a process variable and placing it into a WSDL message that is then sent back in response to the one accepted by the preceding Receive. In fact syntactically, a Reply activity requires that the process have a Receive (or equivalent) somewhere earlier in the flow of the process. Here's the XML syntax our BPEL Process uses to reply to the loan approval

4. ACID transactions

They preserve the *atomicity*, *consistency*, *isolation*, and *durability* of the operation(s) encompassed by the transaction. This helps preserve the integrity of our programs' logic execution as well as the integrity of the data managed by those programs. Java transaction managers are specifically designed to provide this functionality.

5. Define WS-Coordination framework

The WS-Coordination framework exposes an Activation Service which supports the creation of coordinators for specific protocols and their associated contexts. The process of invoking an activation service is done asynchronously, and so the specification defines both the interface of the activation service itself, and that of the invoking service, so that the activation service can call back to deliver the results of the activation - namely a context that identifies the protocol type and coordinator location.

6. List the stages of WS-Coordination

Instantiation (or activation) of a new coordinator for the specific coordination protocol, for a particular application instance

Registration of participants with the coordinator, such that they will receive that coordinator's protocol messages during (some part of) the application's lifetime.

Propagation of contextual information between Web services that comprise the application. An entity to drive the coordination protocol through to completion

7. Define WS-Coordination context

A coordination identifier with guaranteed global uniqueness for an individual coordinator in the form of a URI

An address of a registration service endpoint where parties receiving a context can register participants into the protocol

A TTL value which indicates for how long the context should be considered valid

Extensible protocol-specific information particular to the actual coordination protocol supported by the coordinator

8. List the types of Choreography.

- Abstract Choreography
- "portable" choreography
- Concrete Choreography

9. Define Abstract Choreography.

The types of information that is exchanged, for example an order sent between a buyer and a seller

The sequence and conditions under which the information is sent.

10. Define Portable Choreography.

Detailed definitions of the physical structure of the information that is exchanged including the WSDL port types and operations

Details of the technology to be used, for example, how to secure the messages and send them reliably

Rules that express, as far as possible, the conditions that are used to control the sequence of exchange of information, in terms of, for example XPath expressions that reference data in the messages.

11. Define Concrete Choreography.

Choreography, where all the details are specified that are required to send a message. This extends the definition in a Portable Choreography to include information about the "endpoints". This can include information such as:

- The URLs that are the destinations of the messages that are sent, and
- Other "endpoint" specific rules such as digital certificates to be used for securing messages.

12. Define WS-Policy

The Web Services Policy Framework (WS-Policy) provides a general purpose model and corresponding syntax to describe the policies of a Web Service.

WS-Policy defines a base set of constructs that can be used and extended by other Web services specifications to describe a broad range of service requirements and capabilities.

13. Define Composable Architecture

The Web service specifications (WS*) are designed to be composed with each other to provide a rich set of tools for secure, reliable, and/or transacted Web services. WS-Policy by itself does not provide a negotiation solution for Web services. WS-Policy is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of policy exchange models.

14. Define Compact Policy Expression

To express a policy in a more compact form while still using the XML Infoset, this specification defines three constructs: an attribute to decorate an assertion, semantics for recursively nested policy operators, and a policy reference/inclusion mechanism.

15. Define Policy Expression

To convey policy in an interoperable form, a policy expression is an XML Infoset representation of a policy. The normal form policy expression is the most straightforward Infoset; equivalent, alternative Infosets allow compactly expressing a policy through a number of constructs.

16. Define Policy Expression

here are three fundamental concepts related to security: the resources that must be secured; the mechanisms by which these resources are secured (i.e., policy guards); and policies, which are machine-processable documents describing constraints on these resources.

Policies can be logically broken down into two main types: permission policies and obligatory policies. A permission policy concerns those actions and accesses that entities are permitted to perform and an obligation policy concerns those actions and states that entities are required to perform. These are closely related, and dependent: it is not consistent to be obliged to perform some action that one does not have

permission to perform. A given policy document is likely to contain a mix of obligation and permission policy statements.

17. List the Security Threads

- Confidentiality
- Man-in-the-middle
- Spoofing
- Denial of Service
- Replay Attacks

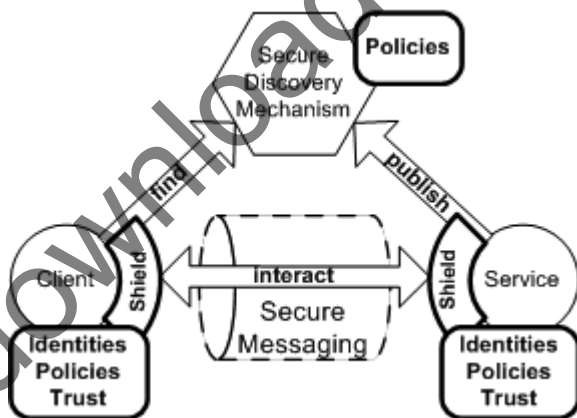
18. List the Web Services Security Requirements

- Authorization
- Data Integrity and Data Confidentiality
- Integrity of Transactions and Communications
- Non-Repudiation
- End-to-End Integrity and Confidentiality of Messages
- Audit Trails
- Distributed Enforcement of Security Policies

19. Define Secure Messaging

Secure Messaging ensures privacy, confidentiality and integrity of interactions. Digital signatures techniques can be used to help ensure non-repudiation.

Techniques that ensure channel security can be used for securing messages. However, such techniques are applicable in a few limited cases. Examples include a static direct connection between a requester agent and a provider agent. For some applications, such mechanisms can be appropriate. However, in the general case, message security techniques such as encryption and signing of the message payload can be used in routing and reliable messaging.



IT2401 SERVICE ORIENTED ARCHITECTURE

16 – MARK QUESTIONS

UNIT – I

1. Explain briefly about Web services as component wrappers
2. Explain briefly about Multi-Grained Services
3. Explain briefly about Client / Server architecture
4. Explain briefly about Distributed Internet architecture
5. Explain briefly about Anatomy of SOA.
6. How components in an SOA interrelate?

UNIT- II

1. Explain briefly about Processing SOAP Messages
2. Explain briefly about Data and Message Exchange Patterns for Enterprise SOA
3. Explain briefly about SOA and BPM in the Coordination Model
4. Explain briefly about Business activity states.
5. Explain briefly about Technical Requirements for Orchestration and Choreography

UNIT- III

1. Explain briefly about Business-Centric SOA.
2. Explain briefly about Deriving Business-services.
3. Explain briefly about Service modeling.
4. Explain briefly about Service Oriented Design and Development
5. Explain briefly about WSDL.
6. Explain briefly about SOAP
7. Explain briefly about Service Composition
8. Explain briefly about SOA Composition guidelines
9. Explain briefly about entity-centric business service design
10. Explain about data points you need for the service contract are.
11. Explain briefly about task-centric business service design

UNIT- IV

1. Explain briefly about Service agents processing incoming and outgoing SOAP message headers.
2. Discuss about Contemporary SOA support
3. Discuss about J2EE handlers as service agents.
4. Discuss about typical J2EE service provider.
5. Discuss about typical J2EE service requester.
6. Explain briefly about Creating a Simple Web Service and Client with JAX-WS.
7. Explain briefly about JAXB Architecture..
8. Explain briefly about JAXR Architecture.
9. Explain briefly about WSIT.
10. Discuss about .NET handlers as service agents.
11. Discuss about typical .NET service provider.
12. Discuss about typical .NET service requester.

13. Explain briefly about CLR.
14. Explain briefly about ASP.NET web services.
15. Explain briefly about Web Services Enhancement.

UNIT- V

1. Explain briefly about WS-BPEL.
2. Explain briefly about WS-Coordination.
3. Explain briefly about WS- Choreography Model Description
4. Explain briefly about WS- Policy
5. Explain briefly about WS- Security