SOA Model

D. R. Ingle[#1], Dr. B.B. Meshram[*2]

[#]*Department of computer Engineeringt, University of Mumbai*
Bharati Vidyapeeth College of Engineerig, Navi Mumbai, India
[1]dringleus@yahoo.com

[*]*Department of Computer Technology, VJTI*
Matunga, Mumbai, India
[2]*bbmeshram@vjti.org.in*

Abstract— With the growing amount of information in various domains, retrieval and analysis is the most frequently used operation. Various institutions/organizations generate valuable information in various domains which is queried and analyzed by users for various purposes. Most of the applications performing these tasks are predominantly database driven and tightly coupled with the system. This limits the possibilities of seamless database integration with other sources of knowledge and also their ability to adopt to changes in the information structures. We describe a generic approach comprising a loosely coupled system with an ability to perform complex querying, analysis and seamless integration with other systems, both off line and over internet.

Index Terms— Web Services, XML, WSDL, BPEL, UDDI, ESB.

I. INTRODUCTION

The Web Services architecture describes the principles behind the next generation of e-business architectures, presenting a logical evolution from object-oriented systems to systems of services[1]. Web Services systems promote significant decoupling and dynamic binding of components[2]. All components in a system are services, in that they encapsulate behavior and publish a messaging API [3]to other collaborating components on the network. Services are marshaled by applications using service discovery for dynamic binding of collaborations[4]. These new applications, themselves, become services, thus creating aggregated services available for discovery and collaboration[5].

Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the Web[6]. The Web Services architecture is the logical evolution of object-oriented analysis and design, and the logical evolution of components geared towards the architecture[7], design, implementation, and deployment of e-business solutions[8].

A Web service is a software system designed to support interoperable[9] machine-to-machine interaction over a network[10]. It has an interface described in a machine-processable format (specifically WSDL)[11]. Other systems interact with the Web service in a manner prescribed by its description using SOAP[12] messages, typically conveyed using HTTP[13] with an XML[14] serialization in conjunction with other Web-related standards[15].

The rest of the paper is organized as follows. Section 2 deals with proposed SOA model, Section 3. gives the conclusion

II. PROPOSED MODEL

A. Categories of Web services

Web services can be categorized into three categories like:

1. **Business information**: Information about the particular business can be shared with consumers or other businesses.

2. **Business integration**: For the business promotion, the business becomes part of a global network of value-added suppliers that can be used to conduct commerce.

B. Web Services components

For the execution of the web service following activities need to be taken:

• A Web service needs to be create its interfaces and invocation methods must be defined.
• A Web service needs to be published to one or more intranet or Internet repositories for potential users to locate.
• A Web service needs to be located to be invoked by potential users.
• A Web service needs to be invoked to be of any benefit.
• A Web service may need to be unpublished when it is no longer available or needed.

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.

1. Requesters and Providers: The provider entity is the person or organization that provides an appropriate agent to implement a particular service. A requester entity is a person or organization that wishes to make use of a provider entity's Web service.

2.Service Description: The Web Service Description (WSD) is a machine-processable specification of the Web service's interface, written in WSDL. It defines the message formats, datatypes, transport protocols, and transport serialization formats that should be used between the requester agent and the provider agent.

3.Semantics: The semantics of a Web service is the shared expectation about the behavior of the service. In effect, this is the "contract" between the requester entity and the provider entity regarding the purpose and consequences of the interaction. The semantics represents a contract governing the meaning and purpose of that interaction.

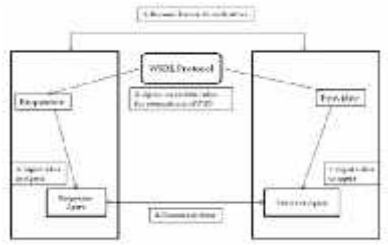

Fig 1: Working of Web Service

C. Overview of Engaging a Web Service

1) There are many ways that a requester entity might engage and use a Web service as shown in fig.

2) 1. The requester and provider entities become known to each other

3) 2. The requester and provider entities somehow agree on the service description and

4) semantics that will govern the interaction between both of them.

5) 3. The service description and semantics are realized by requester and provider agents.

4. The requester and provider agents exchange messages, thus performing some task on behalf of the requester and provider entities.

D. Web services task properties

• Discoverable: It has to be discovered and accessed by consumers.
• Communicable: often asynchronous messaging as opposed to synchronous messaging.
• Conversational: A conversation involves sending and receiving documents in a context.
• Secure and Manageable: Security, manageability, availability, and fault tolerance are critical for a commercial web-service.

E. Characteristics of web services:

1. Web services are self-contained. On the client side, without any additional software
using programming language with XML and HTTP , client request is enough to get start. While On the server side, a Web server and servlet engine are required.

2. Web services are self-describing. The client and server need to recognize only the
format and content of request and response messages.
3. Web services are modular. More complex Web services can be created using number of simple web services either by using workflow techniques or by calling lower layer Web services from a Web service implementation.
4. Web Services are platform independent .Output of the web services are XML-based standards which are designed to promote interoperability

F. Advantages of web services

• **exposing the function on to network:** A Web service is a unit of managed code that can be remotely invoked using HTTP. So, Web Services allows exposing the functionality of existing code over the network..
• **Connecting Different Applications:** Web Services allows different applications to talk to each other and share data and services among themselves. So, Web services are used to make the application platform and technology independent.
• **Standardized Protocol:** Web Services uses standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) use the well defined protocol in the Web Services protocol stack.
• **Low Cost of communication:** Web Services uses SOAP over HTTP protocol for the communication, so can use existing low cost internet for implementing Web Services.
• **Support for Other communication means:** Beside SOAP over HTTP, Web Services can also be implemented on other reliable transport mechanisms. E.g. Web Services can also be implemented using ftp protocol (Web services over FTP).
• **Loosely Coupled Applications:** Web Services are self-describing software modules which encapsulates discrete functionality. Web Services are accessible via standard Internet communication protocols like XML and SOAP.
• **Web Services Sharing:** Web Services supports all the technologies like Enterprise Architecture Interface, Business to Business (B2B) thus helping the business to use existing investments in other technologies.
• **Web Services are Self Describing:** Web Services are self describing applications, which reduces the software development time.
• **Automatic Discovery:** Web Services automatic discovery mechanism helps the business to easy find the Service Providers.
• **Business Opportunity:** Web Services has opened the door to new business opportunities by making it easy to connect with partners.

G. Tools for Web services development

Tools are provided to assist with the following tasks of Web services development.

• Discover. Browse the UDDI Business Registries or WSIL documents to locate existing Web services for integration.
• Create or Transform. Create bottom-up Web services from existing artifacts. Create top-down Web services from WSDL discovered from others or created using the WSDL Editor.
• Build. Wrap existing artifacts as SOAP accessible services and describe them in WSDL.
• Deploy. Deploy Web services into a variety of test environments.
• Test. Test Web services running locally or remotely in order to get instant feedback.
• Develop. Generate sample applications to assist you in creating your own Web service client application.
• Publish. Publish Web services to a UDDI v2 or v3 Business Registry, advertising your Web services so that other businesses and clients can access them.

H. Web Service Architecture

Web Services architecture requires three fundamental operations: publish, find, and bind. Service providers' publish services to a service broker. Service requesters find required services using a service broker and bind to them.

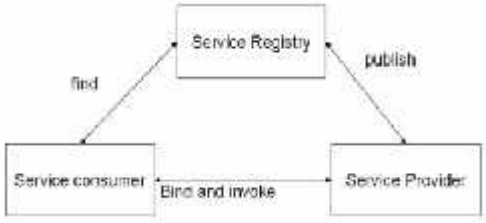

Fig 2: Web service basic architecture

Web services provide a standard means of interoperating between different software applications which are running on different platforms or on different frameworks. A Web service is a software system which is designed to support interoperable machine-to-machine interaction in a network. It has an interface described using Web Service Description Language. Other systems interact with the Web service using SOAP messages, typically conveyed using HTTP with an XML serialization.

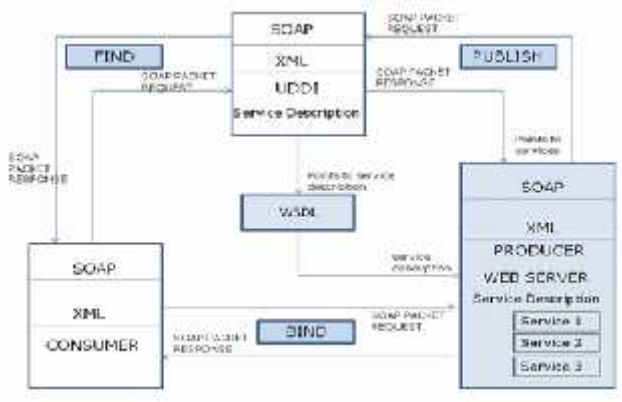

Fig 3 Web Service Architecture

• **Universal Description, Discovery, and Integration (UDDI)** allows businesses to register with an Internet directory that will help them advertise their services.
• **Simple Object Access Protocol (SOAP)** is standard protocol for initiating conversations with a UDDI Service.
• **Web Service Description Language (WSDL)** is the standard for describing Web services based on XML service IDL (Interface Definition Language). This defines the service interface and its implementation characteristics.
• **ebXML (e-business XML)** defines core components, business processes, registry and repository, messaging services, trading partner agreements, and security. ebXML is a B2B framework, which enables enterprises of any size to conduct business over the Internet from any geographical location.
• **Service Description**: Message exchanges are documented in a Web service description (WSD).It defines the message formats, datatypes, transport protocols, and transport serialization formats that should be used between the requester agent and the provider.
• **Service roles and interactions**: A network component in a Web Services architecture can plays fundamental roles: service provider, service broker, and service client.
• **Service providers** create and deploy their Web services and can publish the availability of their WSDL-described services through a service registry, such as a Business Registry.
• **Service brokers** register and categorize published services and provide search services.
For example, UDDI acts as a service broker for WSDL-described Web services.
• **Service clients** use broker services such as the UDDI Business Registry to discover a needed WSDL-described service and then bind to and call the service provider.

I. Service life cycle of the web Services

A service life cycle is expressed in the state transition diagrams below. There are two separate transition paths: service itself and request processing.

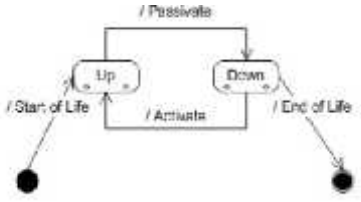

Fig.4. Service life cycle of web services

States
• UP (compound) — (i.e. the service is available).
• DOWN (compound) — (i.e. the service is not available).

Transitions
• Start of Life (SOL) — the service starts its life in UP state.
• End of Life (EOL) — the service ends its life from DOWN state.
• Activate — the service can become available which transitions it from DOWN to UP
• Passivate — the service can become unavailable which transitions it

from UP to DOWN

Universal Description, Discovery, and Integration (UDDI)

Universal Description, Discovery, and Integration (UDDI) is a standard protocol used for Describing available Web services components. This allows businesses to register with an Internet directory that will help them to advertise their services, so companies can find one another and conduct transactions over the Web. UDDI uses registration and lookup task using XML and HTTP(S)-based mechanisms. UDDI includes an XML schema for SOAP messages that defines a set of documents to describe business and services information.UDDI has two functions:

- It is a SOAP-based protocol that defines how clients communicate with UDDI registries.
- It is a particular set of global replicated registries.

J. Technical Architecture of UDDI

Content inserted into the UBR is done at a single node, and that operator node becomes the master owner of that content. Any subsequent updates or deletes of the data must occur at the operator node where the data was inserted.

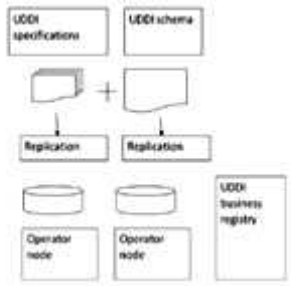

Fig. 5 UDDI initiative

The UBR allows inquiry services, but services may be published only by authenticated entities. Private operator nodes can define the access rules for their nodes on a case-by-case basis. They can follow the same model as the UBR or make the restrictions looser or tighter.

- **UDDI Specifications:** The UDDI project defines a set of XML Schema definitions that describe the data formats used by the various specification APIs.
- **UDDI replication:** describes the data replication processes and interfaces to which a registry operator must conform to achieve data replication between sites.
- **UDDI operators:** this outlines the behavior and operational parameters required by UDDI node operators.
- **UDDI Programmer's API:** This specification defines a set of functions that all UDDI registries support for inquiring about services hosted in a registry and for publishing information about a business or a service to a registry.
- **UDDI registries:**. UDDI registries come in two forms: public and private. A private registry enables to publish and test internal e-business applications in a secure, private environment and a public registry is a collection of peer directories that contain information about businesses and services.
- **UDDI Business Registry:**UDDI assigns a unique identifier to each service description and business registration. These become the service and business keys respectively. UDDI servers are a directory of available services and service providers.

K. Data structure types (UDDI registry)

Registration of a service involves four core data structure types: business information, service information, binding information, and information describing the specifications for services. The relationship between these data types is described in Figure 1.

- **Business information:** Information that is contained in a business Entity structure.

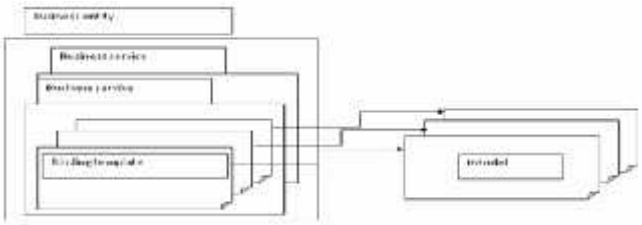

Fig 6 Relationship between data types for UDDI

- **Service information:** The businessService structure contains information about families of technical services. It groups a set of Web services related to either a business process or group of services.

- **Information describing the specifications for services:** Metadata about the various specifications implemented by a given Web service represented by the tModel.
- Each child structure has a unique parent structure. This means that each businessService structure is owned by a specific businessEntity. In turn, each bindingTemplate is owned by a specific businessService.
- **Publisher assertions:** this is a way in UDDI to associate businessEntity structures. The publisher assertion defines a group of businessEntity structures.
- **Service projections:** This enables a business entity to reference a service that was published by another business entity. By using the businessService structure as a projection to an already published businessService, businesses can share or reuse services. Service projections are managed centrally as part of the referencing business Entity.

L. tModel of the UDDI

The tModel structure, short for "Technology Model", represents technical fingerprints, interfaces and abstract types of meta-data. A tModel is a data structure representing a service type type (a generic representation of a registered service) in the UDDI registry, it organizes the service type's information and makes it accessible in the registry database. Corollary with tModels are binding templates, which are the concrete implementation of one or more tModels. Inside a binding

template, one registers the access point for a particular implementation of a tModel. WSDL files are perfect examples of a UDDI tModel. Each business registered with UDDI categorizes all of its Web services according to a defined list of service types. Businesses can search the registry's listed service types to find service providers. Each tModel should contain an overview URL, which references a document that describes the tModel and its use in more detail.

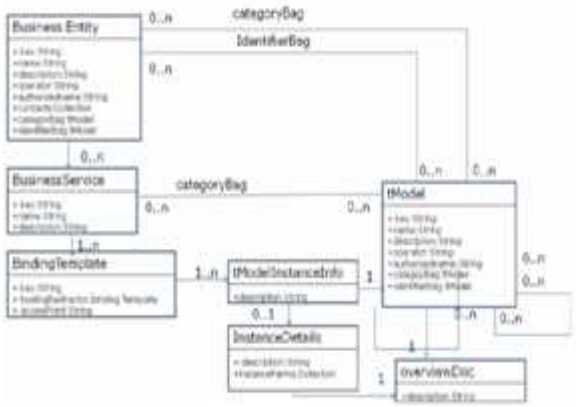

Fig 7 tModel for the UDDI

M. Data structures used in UDDI

• <businessEntity>: The businessEntity structure contains all descriptive information about the business and the services it offers.

• <businessService> : Each businessEntity structure contains one or more businessService structure. A businessService structure describes a categorized set of services offered by a business.

• <bindingTemplate>: The bindingTemplate structure contains a technical description of a service. Each bindingTemplate belongs to a single businessService element.

2. **SOAP- (Simple Object Access Protocol)**

SOAP is a lightweight protocol for the exchange of information in a decentralized, distributed environment. A SOAP message is a transmission of information from a ender to a receiver. SOAP messages can be combined to perform request/response patterns. SOAP is transport independent and most commonly carried over HTTP in order to run with the existing Internet infrastructure. The SOAP standard defines the rules for how data types can be constructed.

2.1. SOAP structure

SOAP is an XML-based protocol that defines three parts to every message:

• **Envelope.**envelope describes what is in a message and how to process it. The envelope is the top element of the XML document, providing a container for control information, the address of a message, and the message itself. Headers transport any control information such as quality-of-service attributes. The body contains the message identification and its parameters. Both the headers and the body are child elements of the envelope.

• **Encoding rules.** The set of encoding rules expresses instances of application-defined
data types. Encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types.

• **Communication styles.** Communications can follow a remote procedure call (RPC) or message-oriented (Document) format.

Fig 8. SOAP structure

2.2. Binding styles of SOAP

SOAP supports two different communication styles:

• **Remote procedures call (RPC):** Invocation of an operation returning a result. Typically used with SOAP encoding, this is not WS-I compliant.

• **Document Style:** Also known as document-oriented or message-oriented style. This style provides a lower layer of abstraction, and requires more programming work.

2.3 Encoding styles of SOAP

In distributed computing environments, encoding styles define how data values defined in the application can be translated to and from a particular protocol format. The translation process is known as serialization and deserialization.

• **SOAP encoding:** The SOAP encoding style allows to serialize/deserialize values of data types from the SOAP data model. This encoding style is defined in the SOAP 1.1 standard, and is not WS-I compliant. WSDL defines the Literal XML encoding style:

- **Literal XML:** Literal refers to the fact that the document should be read as-is, or unencoded. The document is serialized as XMI, meaning that the message XML complies with the Schema in the WSDL.

3. Eb-XML (Electronic Business using eXML)

ebXML as it is typically referred to, is a family of XML based standards sponsored by OASIS and UN/CEFACT whose mission is to provide an open, XML-based infrastructure that enables the global use of electronic business information in an interoperable, secure, and consistent manner. The ebXML architecture is a unique set of concepts; part theoretical and part implemented in the existing ebXML standards work.

4. WSDL (Web Service Description Language)

The WSDL describes services as collections of network endpoints, or ports. The WSDL
specification provides an XML format for documents for this purpose. The abstract definitions of ports and messages are separated from their concrete use or instance, allowing the reuse of these definitions.

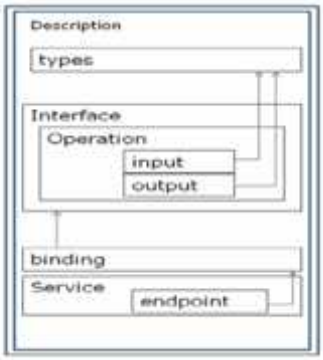

Fig 9 WSDL Structure

• **Service/Service** The service can be thought of as a container for a set of system functions that have been exposed to the Web-based protocols.
• **Port/Endpoint:** The port/endpoint does nothing more than define the address or
connection point to a Web service. It is represented by
a simple HTTP URL string.
• **Binding/Binding:** The binding specifies the interface as well as defining the SOAP
binding style (RPC/Document) and transport (SOAP Protocol) and operations.
• **PortType/Interface:** The <portType> element, renamed to <interface>, defines a Web
service, the operations that can be performed, and the messages that are used to perform the operation.
• **Operation:** Each operation can be compared to a method or function call in a
traditional programming language.
• **Message:** The message contains the information needed to perform the operation. Each message consists of one or more logical parts. Each part is associated with a message-typing attribute. The part name attribute provides a unique name among all the parts of the enclosing message. Parts are a description of the logical content of a message.
• **Types:** The purpose of the types in WSDL is to describe the data. XML Schema is used (inline or referenced) for this purpose.

4.1 Working of WSDL for service Operation
The steps involved in providing and consuming a service are:
1. A service provider describes its service using WSDL. This definition is published to a directory of services. The directory could use Universal Description, Discovery, and Integration (UDDI). Other forms of directories can also be used.
2. A service consumer issues one or more queries to the directory to locate a service and determine how to communicate with that service.
3. Part of the WSDL provided by the service provider is passed to the service consumer. That is the service consumer what the requests and responses are for the service provider.
4. The service consumer uses the WSDL to send a request to the service provider.
5. The service provider provides the expected response to the service consumer.

4.2 Mapping WSDL to UDDI
Both WSDL and UDDI were designed to clearly delineate between abstract meta-data and concrete implementations, and understanding the implications of the division is essential to understanding WSDL and UDDI. WSDL makes a clear distinction between messages and ports: Messages, the required syntax and semantics of a Web Service, are always abstract, while ports, the network address where the Web Service can be invoked, are always concrete. A WSDL can contain solely abstract interface information and not provide any concrete implementation data.
4.3 Quality of service in composite web service
Composition of the web services means number of the web services are integrated together to perform certain task. Instead of being developed from scratch existing services can be used and composed into other web services in order to provide a new and more complex service.The building and execution of this web services can be carried out using:
1) The individual tasks of the composite service are identified
2) The suitable web services for each task are discovered
3) The optimal composition of these web services is identified
4) The execution phase executes the assigned web services.
In four basic workflow patterns were used: sequential, parallel, conditional and loop. Another kind of construct, called discriminator or fault tolerant was introduced. Resulting with the following 5 patterns:

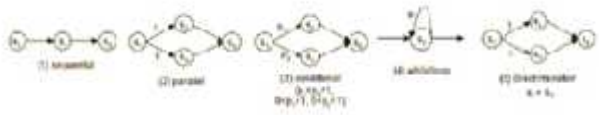

Fig 10 web service executing patterns

1)**Sequential**: a web service is executed after
the execution of a predecessor one
2)**Parallel**: two or more web services are
executed simultaneously.
3)**Conditional**:There's a condition which establishes
what web service will be called.
4)**While/loop**: The same web service is called
several times.
5) **Discriminator**: Two or more web services performing the same functionality are executed at the same time. Only the

response of the faster one is taken.

5.XML - Extensible Markup Language

XML is a method for putting structured data in a text file. XML looks a bit like HTML but isn't HTML. Like HTML, XML makes use of tags (words bracketed by '<' and '>') and attributes (of the form name="value"), but while HTML specifies what each tag & attribute means. XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it

5.1 Difference between HTML and XML

• HTML designed to display data whereas XML was designed to transport & store data.
• XML is not a replacement for HTML, rather it is a complement for HTML
• XML was designed with the focus on what data is whereas HTML was designed with the focus on how data looks.

5.2 Syntax for XML

• XML documents are based on a tree structure. All the contents of the document are contained within a single root tag which can be anything defined by the user.
• Within the root tag there are child elements, which further can have sub child elements as well as siblings (elements at the same level).
• XML documents have to follow a strict syntax & formatting, which will ultimately result —well formed|| XML documents.
• All XML tags must have a closing tag. Omitting the closing tag for a XML element is illegal& it would identify as an error XML tags are case sensitive.

Therefore, start & end tags must write with the same case

5.3 Algorithm for mapping XML Document Structure to Database Schema

Figure 2.16 shows possible mappings from an XML document structure to an Object Oriented or Object Related schema. The arrows denote possible mappings between the two. As shown in the arrows numbered 1 through 4, each XML element or attribute can be mapped to either a database class or a column. However, if XML attributes are mapped to classes, join operations are required unnecessarily when processing queries and, therefore.

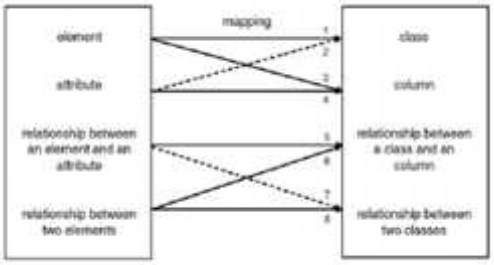

Figure11 mapping from XML structure to database schema

A relationship between an element and an attribute can be mapped to either a relationship between a class and a column (i.e., arrow 5) or a relationship between a class and another class (i.e., arrow 7). However, only the latter option applies because XML attributes are mapped to only the database columns. The relationship between an element and another element is mapped to either the relationship between a class and a column (i.e., arrow 6) or the relationship between two classes (i.e., arrow 8).

Algorithm for mapping XML to database
Input: D XML document
Output: object O with records E

1. {
2. for (element E in the XML document D)
3. {
4. if (E is mapped to a class)
5. {
6. create an object O in the class to which E is mapped;
7. if (there exists an object Op that stores the parent element and the relationship R)
8. {
9. if (R is a one-to-one relationship)
10. store the OID of O in the column of Op that is a reference to O;

11. else if (R is a one-to-many relationship)
12. store the OID of O in the column of Op that is a collection of references to O;
13. if (R is a bi-directional relationship)
14. store the OID of Op in the column of O that is a reference to Op;
15. }}
16. else if (E is mapped to a column C)
17. {
18. if (the type of C is simple)
19. store the value of E in the column of the object O to which E is mapped;
20. else if (the type of C is a simple collection)
21. store the value of E in the column of the object O to which E is mapped as a collection
22. element; }
23. for (each attribute A that belongs to E)
24. store the value of A in the column of the object O to which E is mapped;
25. }

5.4 Advantage of XML

• The main use of XML is that it can be used as a way to structure & describe information. Highly structured nature of XML makes it possible to apply it for all kinds of information.
• XML is used as a way of transporting & storing data. XML is intended to use with the internet to deliver information.
• XML is used to transport data through internet, so that data can be accessed from anywhere.
• XML can be used as a way to interchange data between systems, which were originally not designed to do so.
• Bridge the gap between different systems which are using data of incompatible formats XML is also used as a complement for HTML.
• When displaying dynamic data in HTML documents, the

data can be stored in separate XML files, so that no need to change HTML code when data change.

• It is text-based. It supports Unicode, allowing almost any information in any written human language to be communicated.

• It represent the most general computer science data structures: records, lists and trees.

• The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent.

• XML is heavily used as a format for document storage and processing, both online and offline. It is based on international standards. It can be updated incrementally.

• It allows validation using schema languages such as XSD and Schematron, which makes effective unit-testing, firewalls, acceptance testing, contractual specification and software construction easier.

• The hierarchical structure is suitable for most (but not all) types of documents.

• It manifests as plain text files, which are less restrictive than other proprietary document formats.

• Forward and backward compatibility are relatively easy to maintain despite changes in DTD or Schema

5.5 XML in SOA and Web services

Advantages that XML offers in SOA and Web services include

oa low barrier to entry (it is simple and quite easy to read and debug)

oflexibility (its self describing nature is more forgiving than rigid data formats)

orich structure (it easily models hierarchical, variant, and even graph-oriented data)

oLoose coupling (its textual nature eliminates some of the problems that arise when applications exchange binary data such as serialized language objects).

5.6 Weblogic Integration using XML

Business processes send and receive messages and interact with their environment through a controls paradigm. Messages can contain either XML or non-XML (binary or textual) data. XML Schema types are used to describe all message contents; in the case of non-XML message types, WebLogic Integration (tool FormatBuilder) that allows developers to (virtually) define their data content in XML terms.

6. SOA (Service Oriented Architecture)

Integration Technologies before SOA was performed using 2 stages as described below:

• **Data integration:** The goal of data integration systems is to build applications by integrating heterogeneous data sources. Data integration systems have three elements

o Source schema: refers to the data model of data sources to be integrated.

oMediated (target) schema: schema is the view of the integrated system from the existing data sources.

Mapping provides mechanisms for transforming queries and data from the integrated systems to those of data sources.

The drawback of this approach is that it requires a significant effort to understand the data models and to maintain the mediated schema in the wake of changes in the data sources.

• **Business logic integration:** The integration of applications at the business logic level has been thoroughly studied giving rise to technologies such as remote procedure calls (RPCs), object brokers (such as DCOM and CORBA), message brokers, electronic data interchange (EDI) and also standard specifications such as RosettaNet.

6.1 Business Perspective of SOA

Out of a business perspective, SOA is said to improve business agility and to maintain services being directly applicable to the existing business logic of the business: A service-oriented architecture provides the flexibility to treat elements of business processes and the underlying IT infrastructure as secure, standardized components (Services) that can be reused and combined to address changing business priorities.

6.2 Technical perspective of SOA

The technical perspective emphasizes the importance of the actual structure of the architecture, i.e. of what SOA is made of and how it works:

SOA is an enterprise-wide IT architecture that promotes loose coupling, reuse, and interoperability between systems. An application architecture in which all functions or services are defined using a description language and have callable interfaces that are called to perform business processes.

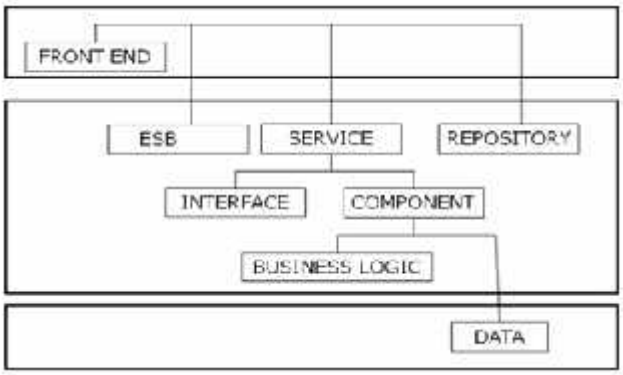

Figure 12 components of SOA, only considering service providing aspect

• **Client/server architecture, front-end:** SOA is best described as a client/server architecture. SOAs are usually built upon two or more tiers. The tier maintaining the graphical user interface (GUI) and representing the so called Front-End is called Presentation-tier, while the one presenting the components with the business logic is called Business logic-tier.

• **Enterprise service bus (ESB): N**ot only one server acts as provider, but normally several, where each and every one of them might be a service that is available to all possible clients

or requestors.

• **Service repository:** Together with the middleware action, such as service interaction, the Business logic-tier also maintains service repositories or directories that store metadata, i.e. the metadata of published services. Repository is similar to a place where construction goods are stored and can be instantly retrieved for use when needed.

• **Service repository:** Together with the middleware action, such as service interaction, the Business logic-tier also maintains service repositories or directories that store metadata, i.e. the metadata of published services. Repository is similar to a place where construction goods are stored and can be instantly retrieved for use when needed.

• **Service:** A service is a reusable function which can be used according to the requirement. Services are exchanged between requestors and providers over the ESB through interfaces. Service is an application component deployed on network-accessible platforms hosted by the service provider.

• **Components:** A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard. Components, in contrast to objects, have no actual need of only containing classes or even at all. Instead components might contain traditional procedures and even have global (static) variables, or it may be realized in its entirety using a functional programming approach, or using assembly language, or any other approach

• **Interface:** The interfaces are the —contracts creating transparency, maintaining all the
information needed (information hiding) to symbolize one specific service, as well as gathering all component end-points for system independency:

1. A shared boundary across which information is passed.
2. A hardware or software component that connects two or more other components for the purpose of passing information from one to the other.
3. To connect two or more components for the purpose of passing information from one to the other.

6.3 SOA resume

After having presented some introducing general definitions of SOA, as well as some detailed information about different attributes of the architecture, it might seem difficult to put forward a general definition. SOAs consist of services that are defined by explicit, implementation independent interfaces. They are loosely bound and invoked through communication protocols that stress location transparency and interoperability.

• **Service enablement:** Each discrete application needs to be exposed as a service.

• **Service orchestration:** Distributed services need to be configured and orchestrated in a unified and clearly defined distributed process.

• **Deployment:** Emphasis should be shifted from test to the production environment, addressing security, reliability, and scalability concerns.

• **Management:** Services must be audited, maintained and reconfigured. The latter requirements require that corresponding changes in processes must be made without rewriting the services or underlying application.

6.4 Service oriented modeling

The process of service-oriented modeling and architecture consists of three general steps:

- **Identification**
- **Specification and**
- **Realization of Services**

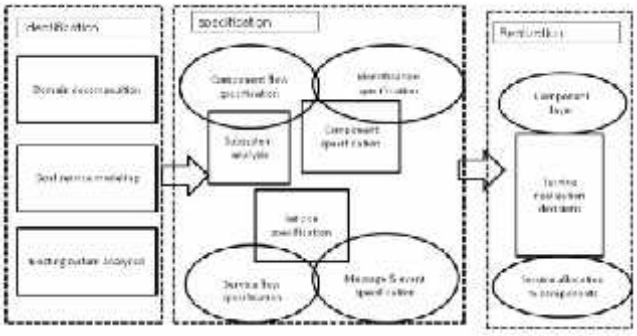

Figure 13 Service-Oriented Modeling And Architecture Method

Service Identification: in this process SOA combination of top-down, bottom-up, and Middle-out techniques of domain decomposition, existing asset analysis, and goal-service modeling are used. Based on the types of the business models used respective domain decomposition is done.In the top-down view, use cases provide the specification for business services. This process is often referred to as domain decomposition, of the decomposition of the business domain into its functional areas and subsystems, including its flow or process decomposition into processes, sub-processes, and high-level business use cases.In the bottom-up view, existing systems are analyzed and selected as viable candidates for providing lower cost solutions to the implementation of underlying service functionality that supports the business process. In this re-modularization of the existing is done to achieve the new technique. The middle-out view, validate and unearth other services not captured by either top-down or bottom-up service identification approaches. It ties services to goals and sub-goals, key performance indicators, and metrics. Service request schema works as shown in the figure 13.

Service Classification or Categorization

This activity is started when services have been identified. This is to classify the service into service hierarchy, which reflects the composite or fractal nature of services. Here services can and should be composed of finer- grained components and services. Classification along with determining composition and layering of the services. It also coordinates building of interdependent services based on the hierarchy. Also, it helps to define, design, and deploy fine grained services with very little governance, which will result in major performance, scalability, and management issues.

Subsystem Analysis: after the domain decomposition, this activity takes the subsystems and specifies the interdependencies and flow between them. The analysis of the subsystem consists of creating object models to represent the

internal workings and designs of the containing subsystems that will expose the services and realize them. The design subsystem will be realized as an implementation construct of a large-grained component.

Component Specification: in this details of the component that implement the services are specified as Data, Rules, Services, Configurable profile, Variations and Messaging and events specifications and management definition occur at this step.

• **Service Allocation:** this process is of assigning services to the subsystems that have
been identified so far. These subsystems realize their published functionality to the layers in the SOA. Always this subsystem has a one-to-one correspondence with the enterprise components. Structuring of these subsystems is to construct enterprise components with a combination of Mediators, Façade, Rule objects, Configurable profiles, Factories. Allocation of components and services to layers in the SOA requires the documentation and resolution of key architectural decisions. This relates to the application architecture and also to the technical operational architecture designed and used to support the SOA realization at runtime.

• **Service Realization:** here the decision is taken to which legacy system module will be
used to realize a given service and which services will be built from the ground-up. Other realization decisions like security, management and monitoring of services. Top-down domain is conducted in parallel with a bottom-up analysis of existing legacy assets that are candidates for componentization and service exposure

III. CONCLUSION

A Web services are refocusing organizations on the concepts of service-oriented architecture. Although highly reusable, loosely coupled architectures have been a goal for many organizations. Web services are fostering interest in and providing the technology to implement service-oriented architectures that enable them to realize their vision. SOA has a lot of emphasis on interface. Starting from the messages which are the parts of the interface, the contract which is the collection of the messages, the endpoint where the contract is delivered and the policy which governs the behavior of the endpoint. The focus on interfaces is what gives SOA the ability to create loose coupling, composable components, reuse and achieve the various design goals. Another nice attribute of this definition is that we can use as a base for both the technical and the business perspectives of SOA as the common elements of both perspective are used in this definition.

REFERENCES

[1] A Dynamic Data Integration Model Based on SOA Jun Wang 2009 ISECS International Colloquium on Computing, Communication, Control, and Management

[2] Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. 7, 39–59 (1994)

[3] Lingjuan Li,Wenyu Tang. Research of the Applications of CBR in Business. Journal of Nanjing University of Posts and Telecommunications,No.10,2006 pp:17-21

[4] Loh, Y., Yau, W., Wong, C., Ho, W.: Design and Implementation of an XML Firewall. Computational Intelligence and Security 2, 1147–1150 (2006)

[5] Yee, G., Shin, H., Rao, G.S.V.R.K.: An Adaptive Intrusion Detection and Prevention (ID/IP) Framework for Web Services. In: International Conference on Convergence Information Technology, pp. 528–534. IEEE Computer Society, Washington (2007)

[6] Research on Intelligent Learning Strategy for Knowledge-enabled Customer Relationship Management Based on SOA and CBR.

[7] Yan Qin, Zhang Guoliang, Wang Keyi Knowledge-enabled Human Resource Development Based on e-HR. Computer Applications and Software, No.11,2008

[8] BPEL 2.0 specification - OASIS. (2007). [Online]. Available: http://docs.oasis-open.org/wsbpel/

[9] I. Matsumura, T. Ishida, Y. Murakami, and Y. Fujishiro. Situated web service: Context-aware approach to high speed web service communication. In IEEE International Conference on Web Services (ICWS-06), pages 673–680, 2006.

[10] Utilizing WS-BPEL business processes through ebXML BPSS. Bahareh heravi

[11] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Y aron Goland, Satish Thatte, "Business Process Execution Language for Web Services Version 1.1", 5 May 2003

[12] An Optimized Design of Service Orchestration Haoping Bai', Meina Song', Huiyang Xu', Qian Wang', Lingyun Fu' Choreographing Web Services Adam Barker, Christopher D. Walton, and David Robertson

[13] The OASIS Committee, Web Services Business Process Execution Language (WS-BPEL) Version 2.0. 2007. Choreography Design Using WS-BPEL Oliver Kopp Frank Leymann

[14] How BPEL and SOA Are Changing Web Services Development James Pasley Cape Clear Software Tier Caching for SOA Performance by Kiran Dattani, Milind Pandit, and Markus Zirn

[15] Enterprise Architecture and Web Services Dinarle Ortega, Elluz Uzcátegui, María M. Guevara

[16] Service oriented architectures: approaches, technologies and research issues Mike P. Papazoglou · Willem-Jan van den Heuvel

[17] Research of Business Transaction Process in SOA Environment Ling Yun，Lin Guangyan

[18] Service-Oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned Olaf Zimmermann, Vadim Doubrovski

Mr. D.R. Ingle (ISTE LM'2004) is Professor of Computer Engineering Department at Bharati Vidyapeeth College of Engineering, Navi Mumbai, Maharashtra state; India received bachelor degree, and Master degree in computer engineering. He has participated in more than 10 refresher courses to meet the needs of current technology. He has contributed more than 25 research papers at national, International Journals. He is life member of Indian Society of Technical Education. His area of interest is in Databases, intelligent Systems, and Web Engineering.

Dr. B.B.Meshram (CSI LM'95, IE '95) is Professor and head of Computer Technology department at VJTI, Matunga, Mumbai, Mahar ashtra state, India. He received bachelor degree, Master degree and doctoral degree in computer engineering. He has participated in more than 30 refresher courses to meet the needs of current technology. He has chair more than 15 AICTE STTP Programs and conferences. He has received the appreciation for lecture at Manches ter and Cardip University, UK. He has contributed more than 250 research papers at national, International Journals. He is life member of computer society of India and Institute of Engineers. His current research interests are in Databases, data warehousing, data mining, intelligent Systems, Web Engineering and Network security.

