Distributed Systems – Techs
5. Service-Oriented Architectures

SOA

- Term first coined by Yefim Natis in one of the research papers in 1994:

  SOA is a software architecture

  that starts with an interface definition and

  builds the entire application topology as a topology of interfaces, interface implementations, and interface calls.

- Despite being coined much earlier, SOA started to become a buzzword only in early 2000.
- With the advent of Web services and WSDL compliant business process, SOA started to become popular among technology enthusiasts.

SOA – an Architectural Style

- SOA is a *style of design that guides* all aspects of creating and using services throughout their lifecycle (from conception to retirement).
- SOA is a way to define and provision an IT infrastructure to allow *different applications to exchange data and participate in processes*,
  - regardless of the OSs or
  - Regardless the programming languages underlying those appls.
- An approach to building IT systems in which services are the key organizing principle used to align IT systems with the needs of the business.
  - In contrast, earlier approaches tended to directly use specific features and functions of a particular execution environment (e.g. OO)

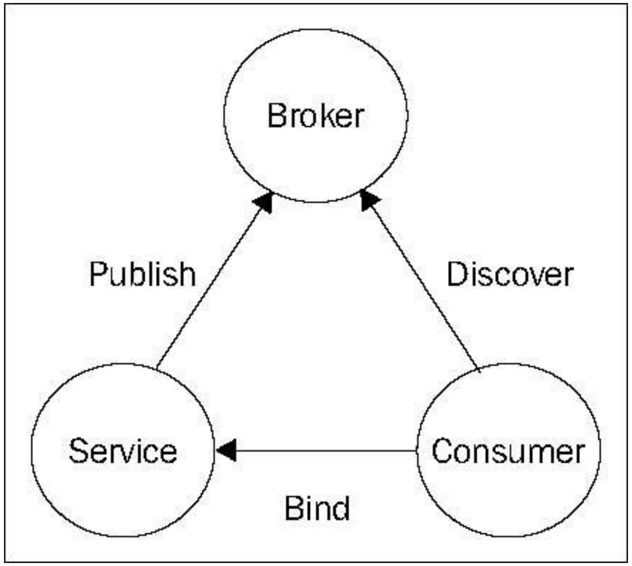
SOA promotes software reusability

- The concept is not new:
  - Traditional OO archs promote reusability by reusing classes or objects.
    - objects are often too fine grained for effective reuse.
  - Component-oriented architectures emerged that use software components as reusable entities.
    - These components consist of a set of related classes, their resources, and configuration information.
    - Do not address additional issues arising from current day enterprise envirs:
      - Today, enterprise environments are quite complex due to the use of a variety of software & hardware platforms, Internet-based distributed communication etc.
- SOA address these issues by using a service as a reusable entity.
  - The services are typically coarser grained than components
  - The services communicate with each other and with end-user clients through well-defined and well-known interfaces.

Fundamental of SOA

- Based upon: Service, Message and Dynamic discovery.
- In a SOA, we have:

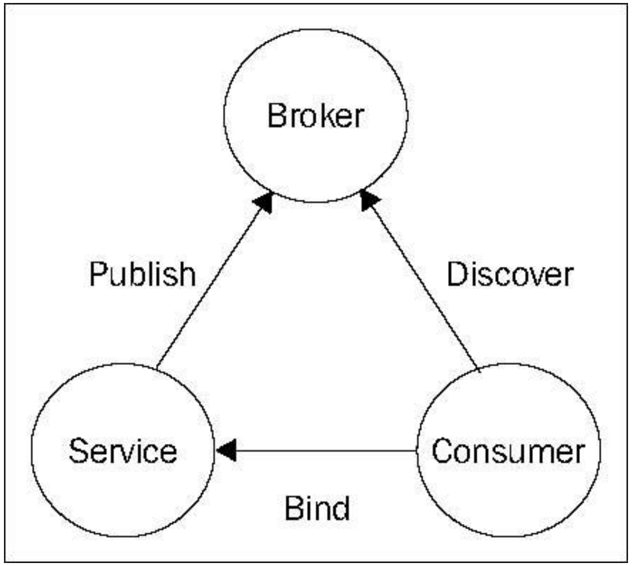
1. A **service** that implements the business logic and exposes this business logic through well-defined interfaces.
2. A **registry** where the service publishes its interfaces to enable clients to discover the service.
3. **Clients** (including clients that may be services themselves!) who discover the service using the registries and access the service directly through the exposed interfaces.



Core components

At a high level, SOA is formed out of three core components:

1. Service Provider (Service) - offers processes in the form of services
2. Service Consumer (Consumer) - services offered by the provider are called by the consumer
3. Directory Services (enabled by Broker) - lie between the provider and the consumer



-The service to be made available to the consumer is published to the directory services in the broker.

-The consumer will discover the service from the broker.

-If the service is found, it will bind to the service and execute the processing logic.

Service Abstraction

- The metadata specify:
  - The location on the network (network address for the service)
  - The a machine-readable description of the messages it receives and optionally returns.
  - Defines what message exchange patterns it supports.
  - A schema for the data contained in the message is used as the main part of the contract (i.e., description) established between a service requester and a service provider.
  - The operations it supports, and
  - Requirements for reliability, security, etc
- The service implementation can be any execution environment for which services support is available.

Executable agent and mapping layer

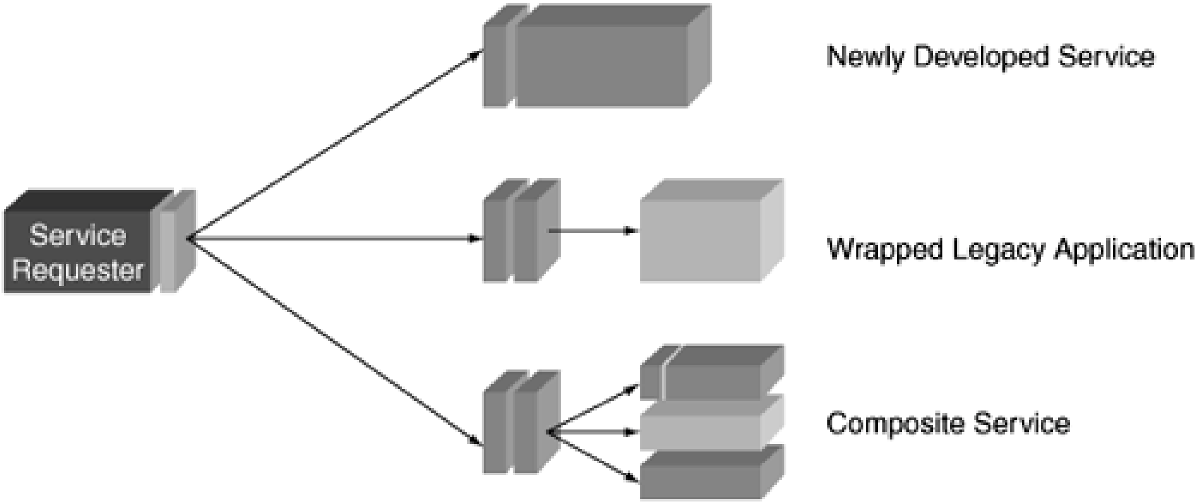
- The service implementation is also called the *executable agent*.
  - runs within the execution environment,
  - Service description is separated from its executable agent:
    - one description might have multiple different executable agents associated with it.
    - one agent might support multiple descriptions.
- A *mapping layer* (also called a transformation layer):
  - Is often implemented using proxies and stubs.
  - Is responsible for accepting the message,
  - Transforms the description data to the native format
  - Dispatches the data to the executable agent.

Service handler

- Services are published by the 'provider' and they bind to the 'consumer' through the service 'handler'.
- The service handler acts as a collaboration agent between the provider and the consumer.
- The handler contains the realization logic
- Once the service has been requested, it goes through various messaging paths and, at times, into multiple handlers
- The handler usually routes the messages to the target system or sometimes does some processing logic before forwarding the request to target system.

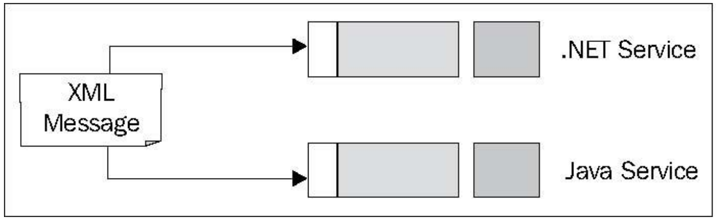
Requester and provider

- A requester (consumer) can be a provider & vice versa
  - an execution agent can play either or both roles
- One of the greatest benefits of service abstraction is its ability to easily access a variety of service types, including
  - newly developed services,
  - wrapped legacy applications, and
  - applications composed of other services (both new and legacy).



SOA Objectives

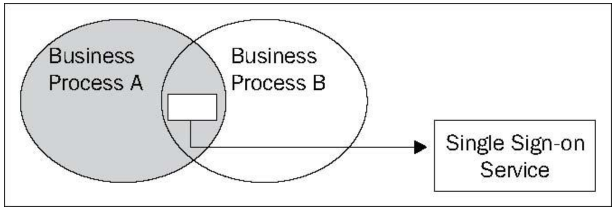
1. **Loose coupling**:
   - The decomposition into independent services will help in bringing down the dependencies on a single process.
2. **Platform-neutrality**:
   - XML-based message information flow enhances the capability to achieve platform neutrality.



3. **Standards**:
   - The message flow is in the form of globally accepted standards.
   - The service only has to depend on the service descriptions

SOA Objectives (2/2)

4. **Reusability**:
   - The application logic being divided into smaller logical units, the services can easily be re-used.



5. **Scalability**:
   - As the processes are decomposed into smaller units, adding new business logic is easy to accomplish.
   - The new logic could either be added as an extended unit of the current service, or it can also be constructed as a new service.

Advantages of SOA (1/2)

1. It enables **development of loosely-coupled applications** that can be distributed and are accessible across a network.
2. **Integration**:
   - SOA-based solution is usually based upon the principles of inter-operability.
   - Lower cost of integration development through a compounded sol.
3. **Business Agility**:
   - The benefit in terms of *software assets* can be derived from SOA's ability to re-use and simplify integrations.
     - development period get shortened.
     - easy to accommodate changes => solution evolves over a longer time period
   - In terms of *hardware benefits*, due to the abstract use of services being loosely coupled, they can be delegated across the domains
     - This helps in balancing the business processes load across the organization
4. **Assets Re-use**:
5. **Increased ROI (Return-of-Investment)**

Transition to SOA

- The biggest issue faced in SOA implementation is the complexity of the solutions.
  - the dismantling of the current business processes into smaller services is a huge challenge in itself.
- Approaches:

1. **Top-down**:
   - the business use cases are created, which gives the specifications for the creation of services.
   - the functional units are decomposed into smaller processes and then developed.
2. **Bottom-up**:
   - the current systems within the organization are studied, and
   - suitable business processes are identified for conversion to services.

SOA vs. OO and CBD

- SOA is a natural improvement over the **object-oriented (OO)** and the **component-based development (CBD)**.
  - it still retains some of the flavors from each of them.
  - the processes are powered by small pieces of software known as 'components'.
  - The logic inside the components is based on the principles of OO programming.

SOAs implemented using a variety of techs

- Distributed objects CORBA, J2EE, COM/DCOM.
- Message-oriented middleware (MOM) WebSphere MQ, Tibco Rendezvous.
- TP monitors CICS, IMS, Encinia, Tuxedo.
- B2B platforms ebXML, RosettaNet.
- Web services
- …

WebSphere MQ

- Many large organizations have created SOAs using WebSphere MQ
  - Case study: AXA Financial
    - insurance and financial services company,
    - uses WebSphere MQ as a messaging and integration layer to connect legacy systems with front-end applications.
    - AXA began developing the architecture in 1989.
    - The SOA integration architecture currently handles more than 600,000 transactions a day.
- Only a small fraction of WebSphere MQ systems are service-oriented.

CORBA (1/2)

- Why CORBA for SOA?:
  - Is an open standard.
  - Supports remote method invocation (i.e., RPC calls), asynchronous messaging, and publish/subscribe communications.
  - Provides integrated security, naming services, transaction management, and reliable messaging.
  - Supports multiple programming languages.
  - Provides CORBA IDL used as a service definition language.
  - Objects can be exposed as Web services because the OMG has defined a CORBA IDL to WSDL mapping.
- Some limitations for implementing an SOA:
  - is perceived as being complex.
  - requires both the requester and provider to be using CORBA.
  - does not provide explicit support for XML and
  - does not support asynchronous exchange of documents over Internet.

CORBA (2/2)

- Many large organizations have created SOAs using CORBA
  - Case study: Credit Suisse Group
    - is a leading global financial services company headquartered in Zurich, Switzerland.
    - In 1997, Credit Suisse started the implementation of an SOA called the Credit Suisse Information Bus (CSIB):
      - the goal of the CSIB was to enable reliable, secure, and scalable real-time request/reply interoperability between back-end systems and a variety of front-end applications based on different platforms (J2EE, C++, SmallTalk, HTML, COM, and Visual Basic).
      - it replaced an integration infrastructure based on IBM WebSphere MQ that was becoming expensive and difficult to maintain
    - Credit Suisse's SOA supports more than 100,000 users, including 600 business services in production.
- Only a small percentage of CORBA systs are service-oriented.

Java and J2EE technologies (1/2)

- Have many of the same advantages and disadvantages as CORBA when it comes to implementing an SOA.
- Similarities related to SOA with CORBA:
  - Both are open standards.
  - Both are distributed object technologies that provide excellent support for remote method invocation
  - Both require the service requester and the service provider to be using the same technology stack (i.e., J2EE and CORBA).
  - Both provide
    - integrated security,
    - naming services (JNDI and CORBA Naming Service),
    - transaction management (JTA/JTS and Object Transaction Service), and
    - reliable messaging (JMS and CORBA Notification).
  - Both J2EE EJBs and CORBA objects can be exposed as Web services.

Java and J2EE technologies (2/2)

- Here are some of the differences related to SOA:
  - CORBA supports multiple programming languages.
  - CORBA provides CORBA IDL as an explicit interface definition language.
  - J2EE Web services communicate natively using XML and SOAP, whereas the CORBA WSDL mapping still communicates using CDL and IIOP.
  - The Java Community Process has defined a series of APIs for manipulating XML (e.g., JAX-RPC, JAAS, JAX-B, and so on).
  - J2EE has a much larger and more robust developer community.
  - J2EE implementations are available from most of the major IT vendors.
- Not all J2EE systems are service-oriented,
- Most J2EE applications are tightly coupled

B2B platforms

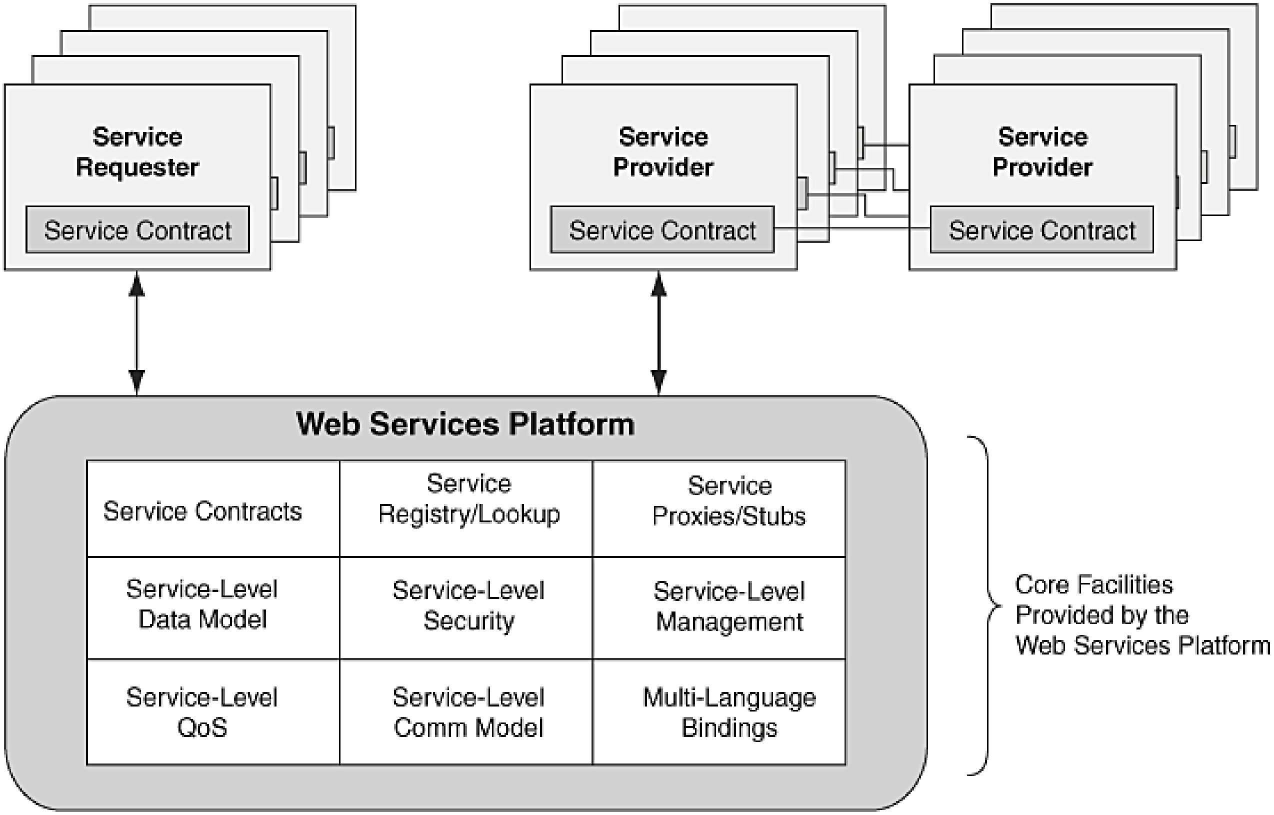
- Examples: ebXML and RosettaNet
- Ideal SOA platforms because:
  - Are open standards.
  - Are loosely coupled.
  - Are based on XML.
  - Are based on the asynchronous exchange of documents (i.e., XML documents).
  - Provide integrated mechanisms for
    - service registration,
    - service security,
    - service monitoring and management,
    - business process management,
    - compensating transactions, and
    - reliable messaging.

SOA using Web Services

- Major advantage of implementing an SOA using Web services:
  - WSs are pervasive, simple, and platform-neutral.
- Other advantages derived from the way in which the WWW achieved its success:
  - a simple document markup language approach such as HTML (or XML) can provide a powerful interoperability solution
  - a lightweight document transfer protocol such as HTTP can provide an effective, universal data transfer mechanism.
  - On the Web, it doesn't matter
    - whether the OS is Linux, Windows, OS390, HP NonStop, or Solaris.
    - whether the Web server is Apache or IIS.
    - whether the logic is coded in Java, C#, COBOL, Perl, or LISP.
    - whether the browser is Netscape, Internet Explorer, Mozilla, or the W3C's Amaya.
- WSs can understand and process an XML-formatted message received using a supported communications transport and return a reply if one is defined.

Web services platform

Capabilities of the complete Web services platform:



Service Contracts	Service Registry/Lookup	Service Proxies/Stubs
Service-Level Data Model	Service-Level Security	Service-Level Management
Service-Level QoS	Service-Level Comm Model	Multi-Language Bindings

Core Facilities Provided by the Web Services Platform