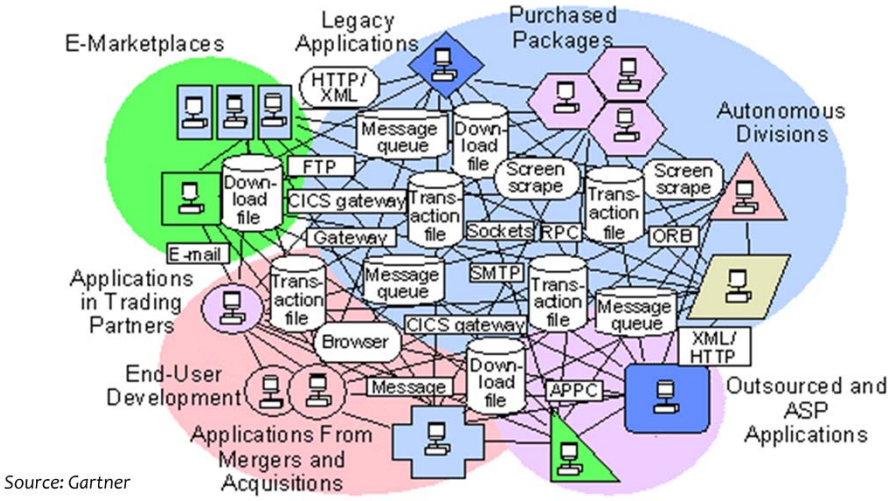

IGATE
Speed.Agility.Imagination


Service Oriented Architecture (SOA)
March 24, 2015
Proprietary and Confidential
- 2 -
IGATE
Speed.Agility.Imagination

Topics to be covered

- What is SOA?
- SOA Infrastructure
- Myths of SOA
- Main Concepts of SOA
- Basic SOA Reference Model
- Evolution of SOA
- Web Services
- Benefits of SOA

Real-world Enterprise Computing



Source: Gartner

IGATE
Speed.Agility.Imagination

What is SOA?

- **SOA is a blueprint that governs creating, deploying, executing, and managing *reusable business services***
  - Services are the fundamental unit of the architecture
  - Services have well-defined interfaces, have encapsulated implementations, and can be accessed by users, applications, and services in a consistent manner no matter where they reside or how they were developed
  - A SOA defines guiding principles, artifacts, people/roles, processes, tools, etc.
  - A SOA enables the *modeling of business problems* in terms of services with reusable interfaces

SOA Infrastructure

- **A SOA also defines the software infrastructure that allows different applications to interact**
  - Allows IT systems to implement business processes, exchange data and execute transactions regardless of the technical platforms used by the underlying applications and IT systems
- **Good SOA infrastructure does not require replacement of existing technologies**
  - Should be able to unite disparate systems and hide their differences

Myths of SOA

Myth – SOA means WebServices, .Net, J2EE, CORBA or ebXML

This not true. These are instead specialized SOA implementation that embody the core aspects of a Service-Oriented approach to Architecture.

Each of these implementations extends the basic SOA reference model

Main Concepts of SOA

The following main concepts are consistent in all SOA implementations:

- Services
- Service descriptions
- Advertising and discovery
- Specification of an associated data model
- Service Contract

IGATE
Speed.Agility.Imagination

Main Concepts of SOA

- **Services**
  - A service is a contractually defined behavior that can be implemented and provided by a component for use by another component
- **Service Descriptions**
  - It consists of the technical parameters, constraints and policies that define the terms to invoke the service. Each service should include a service definition in a standardized format. This enables applications and human actors to examine the service description and determine issues such as, what the service does, how they may bind to it, and what security protocol (if any) must be used with it.

Main Concepts of SOA

Advertising & Discovery of services

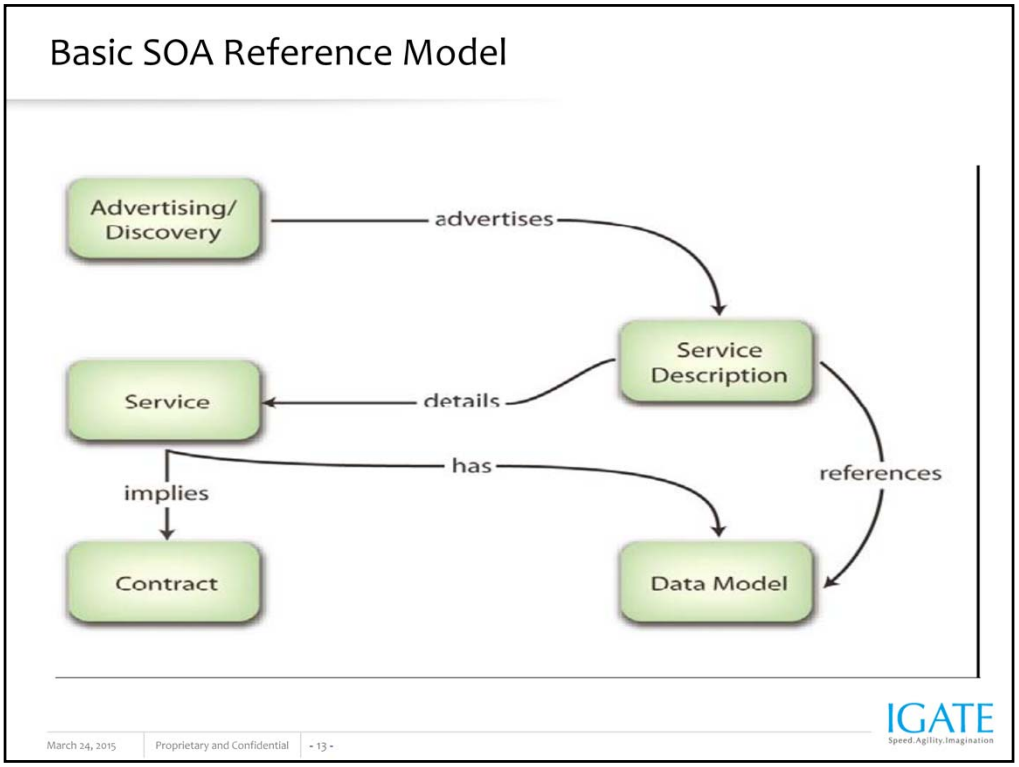
- **Advertising:**
  - A service must communicate its service description in an accessible manner to potential consumers. It does so by using one of the several advertising methodologies, such as Pull and Push
  - In Pull methodology, potential service consumers request the service provider to send them the service description
  - In Push methodology, the service provider, or its agent, sends the service description to potential service consumers
  - The push and pull methodologies may work together to facilitate advertising services through a third party in a publish-subscribe pattern

Main Concepts of SOA

- **Discovery**
  - Discovery occurs when a potential consumer obtains information about the existence of a service, its applicable parameters and terms. Discovery does not constitute authorization to execute against the service; although these details may be included in the discovery pattern
- **Implementing advertising and discovery**
  - It can be implemented using a registry/repository or a services directory. Although using these may make discovery easier, an SOA requires neither of them
  - Known Implementations: OASIS ebXML Registry-Repository and the OASIS Universal Description and Discovery Integration (UDDI)

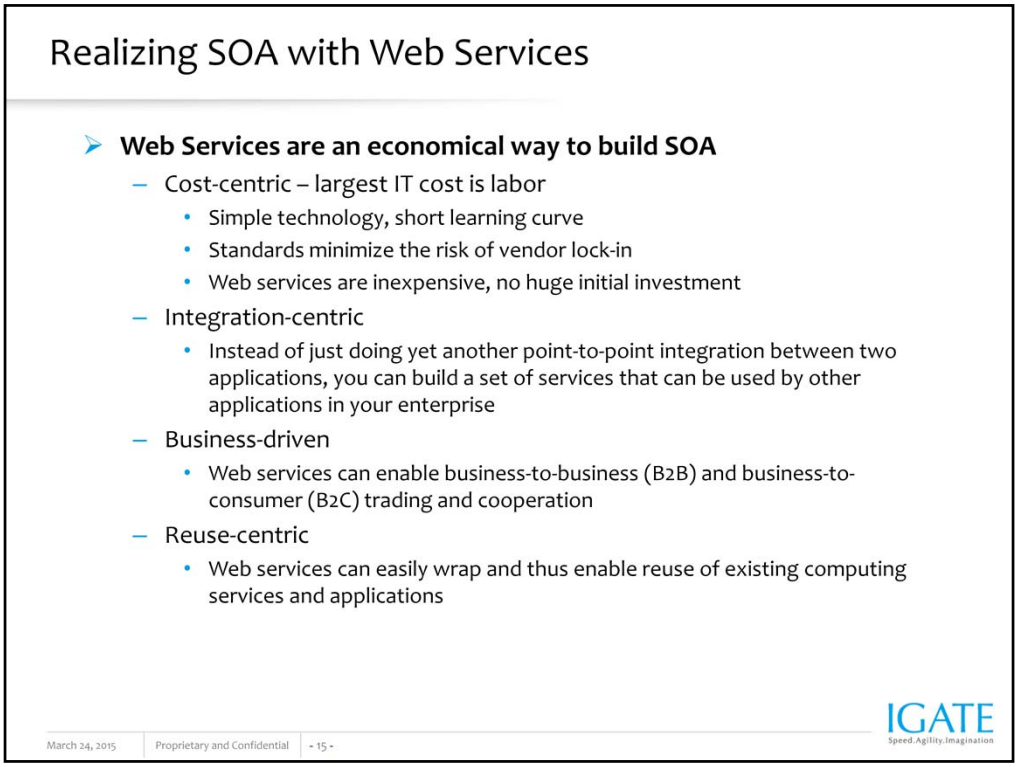
Main Concepts of SOA

- **Specification of associated data model**
  - When invoking a service, certain parameters may be required to help the service fulfill the service request, the service may also pass parameters back to the service consumer
  - Known Implementations: WSDL & ebXML Collaboration Protocol binding


Basic SOA Reference Model
Advertising/ Discovery
advertises
Service Description
Service
details
has
references
implies
Contract
Data Model
March 24, 2015
Proprietary and Confidential
- 13 -
IGATE
Speed.Agility.Imagination

Evolution of SOA

EVOLUTIONARY STEP	EXAMPLE
Monolithic	Large scale application using a procedural coding methodology
Structured or Object Oriented	Dividing applications into units of logic based on functionality. The first steps of SOA.
Clients and Servers	The logical progression of OO – bundling groups of functions on one computer (server) and invoking them from another (client).
3 Tier	Adding an extra layer to interaction. The primary driver was to create an interface that was agnostic to the specific environment of the server. For example, if you make an HTTP request to a server, the middle tier translates your HTTP get() request into the native format that is needed to get the resource requested.
N^{th} Tier	Layered request-response calls between applications. Portal development relied on this concept.
Distributed Objects	A ubiquitous and heterogeneous system of many distributed, orthogonal objects rather than a simple, 2 or 3 computer interaction.
Components	Aggregating objects into logical components that achieve specific functionality (often mapping to a component or the builder's requirements) and creating interfaces to those components. An example is a database server used as a persistent data-storage component for CRM software.
Service Oriented Components	A ubiquitous environment of orthogonal components interacting in a peer-based environment, often using service provider proxies (interfaces based on widely accepted standards) to offer services.


Realizing SOA with Web Services
Web Services are an economical way to build SOA
Cost-centric – largest IT cost is labor
Simple technology, short learning curve
Standards minimize the risk of vendor lock-in
Web services are inexpensive, no huge initial investment
Integration-centric
Instead of just doing yet another point-to-point integration between two applications, you can build a set of services that can be used by other applications in your enterprise
Business-driven
Web services can enable business-to-business (B2B) and business-to-consumer (B2C) trading and cooperation
Reuse-centric
Web services can easily wrap and thus enable reuse of existing computing services and applications
March 24, 2015
Proprietary and Confidential
- 15 -
IGATE
Speed.Agility.Imagination

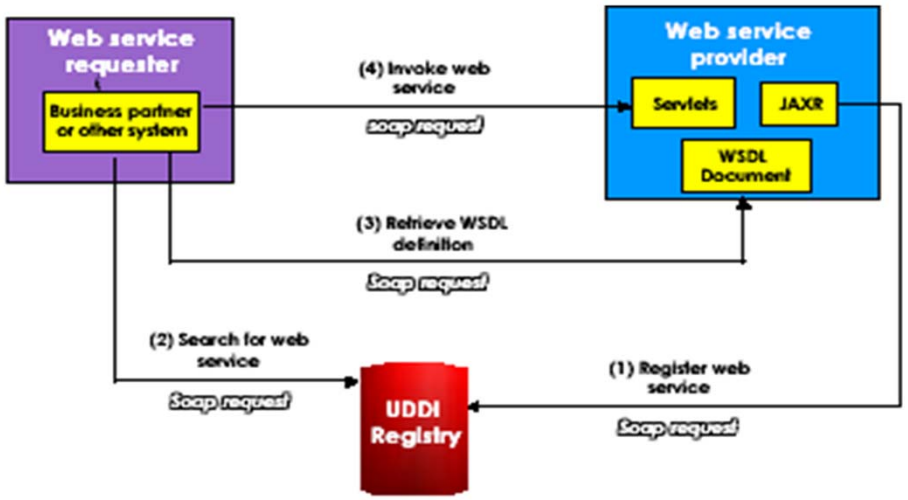
Web Services compared to the Web

- **Web Services are not the same as Web pages or Web applications**
  - No browsers, web servers, HTML, CGI, ASP pages, GUIs, etc.
- **Rather – exchange of XML documents between "co-operating" programs**
  - Web Services technologies intended for program-to-program communications and interaction
  - Not human-to-web page communications

Web Services Technologies Overview

- **XML Technologies**
  - Base XML for documents
  - XML Schema for describing XML documents
- **SOAP**
  - Formerly known as the "Simple Object Access Protocol"
  - How to format XML documents for transmission between applications
- **WSDL**
  - "Web Services Description Language"
  - Defines all details about a service
- **UDDI**
  - "Universal Definition, Discovery and Integration"
  - One way to advertise and discover services

Participants in a Web Services Environment



Benefits of service-oriented design

- **More flexibility ("business agility")**
  - Assumption: business process logic and business rules are no longer buried inside applications
  - Result:
    - Since they are now explicit, processes can be changed easier
    - Existing services can be used in different contexts
    - Shorter time-to market for changed processes
- **Reduced cost of operation through consolidation**
  - Assumption: Redundant functionality is eliminated
  - Result:
    - Fewer servers & Fewer licenses
    - Fewer assets to manage
    - Lower maintenance cost

Benefits of Service Oriented design

- **Higher quality**
  - Eliminating redundancy will reduce inconsistent data and inconsistent behavior
  - More transparency
  - Improved system architecture – easier to understand
- **Reduced risk, cost and complexity for development**
  - Clean architecture ⇒ reduced cost and risk
  - Increased developer productivity through reuse
    - Projects can leverage existing services
  - "Black box" reuse instead of copy & paste reuse

Benefits of Service Oriented design

- **Lessen the dependencies on vendors**
  - Service implementations can be replaced as long as interfaces stay the same
  - Services can be relocated from one platform to another
  - Services can even be outsourced to an external provider
- **Good service design (partitioning) will outlive your middleware or implementation technology**
  - All you have to do is to put a wrapper around it, if required
  - Many mainframe systems today provide many useful services that should be made available to applications elsewhere in the enterprise

Benefits of Service Oriented design

- **Commoditizing more and more parts of the IT infrastructure**
  - Off-the-shelf infrastructure components are moving up the layers and coming closer to the application
  - Due to existing industry standards and available products, developers stop building this stuff themselves:
    - 1990: DBMS, TP Monitors
    - 1992: Networking stacks
    - 1995: CORBA, RPC Middleware, Reliable Messaging
    - 1998: Naming Service, Publish and Subscribe, Event Notification
    - 2000: Various J2EE Services


Thank You
March 24, 2015
Proprietary and Confidential
- 23 -
IGATE
Speed.Agility.Imagination