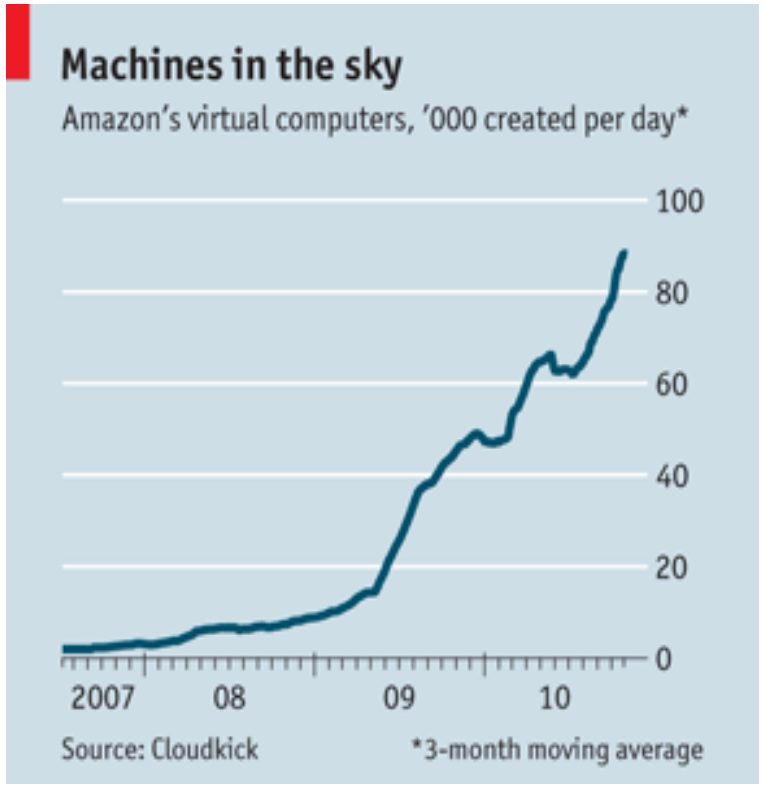
Cloud Computing :
Broker for Cloud Marketplace

Introduction

- Rapid growth of available cloud services
- Huge number of providers with varying QoS
- Different types of customer use cases – each with different requirements

- *Need for a "middle man" (Intelligent Broker!) to*
  - Suggest the best cloud provider to the customer
  - Safeguard the interests of the customer



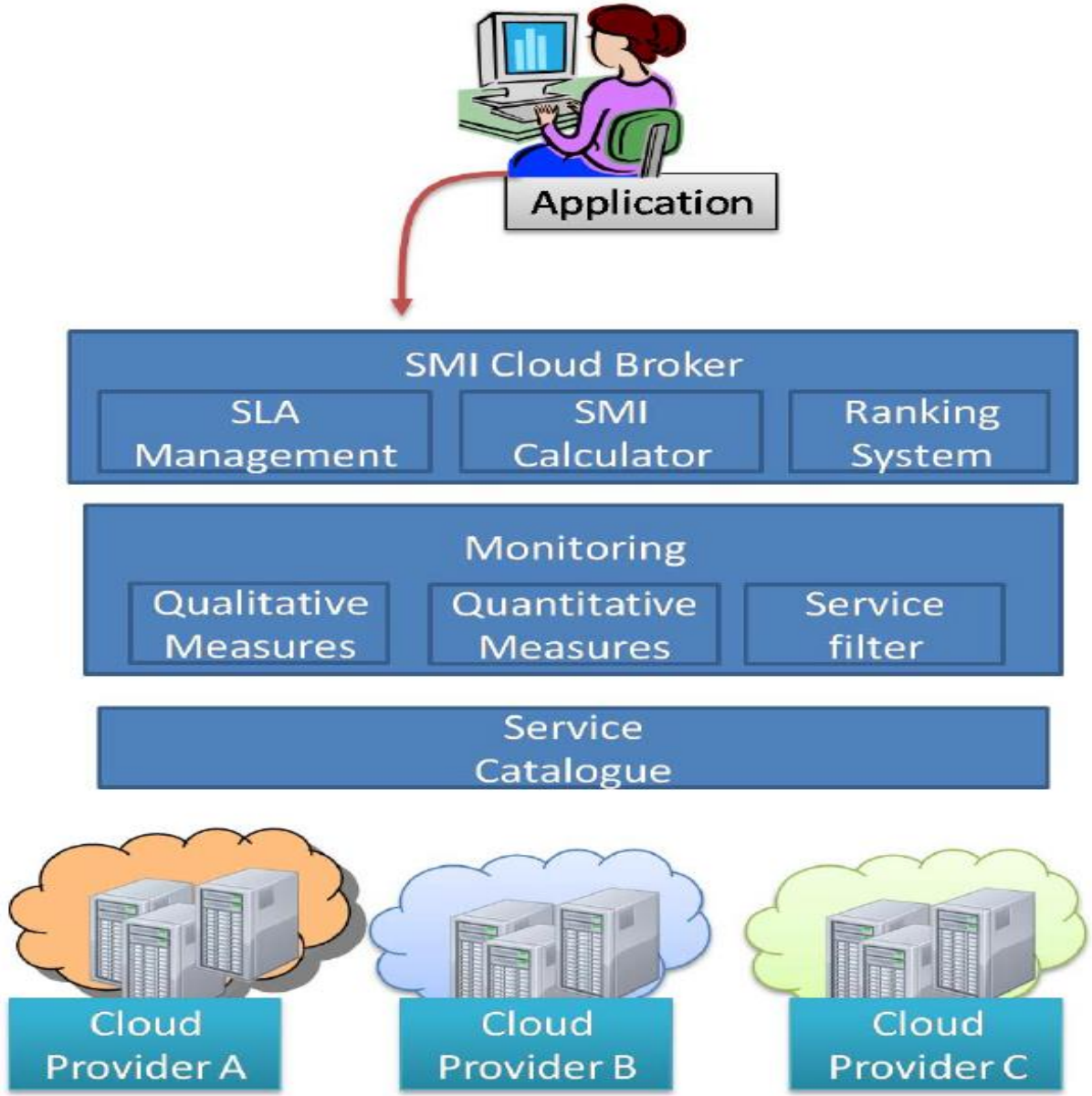
Objectives

- Selection of the most suitable provider satisfying customer's QoS requirements
- Calculation of the degree of SLA satisfaction and trustworthiness of a provider
- Decision making system for dynamic service migration based on experienced QoS

SMICloud

- SMICloud: A Framework for Comparing and Ranking Cloud Services
- High-Level Attributes
  - Accountability
  - Agility
  - Cost
  - Performance
  - Assurance
  - Security and Privacy
  - Usability

SMICloud Framework



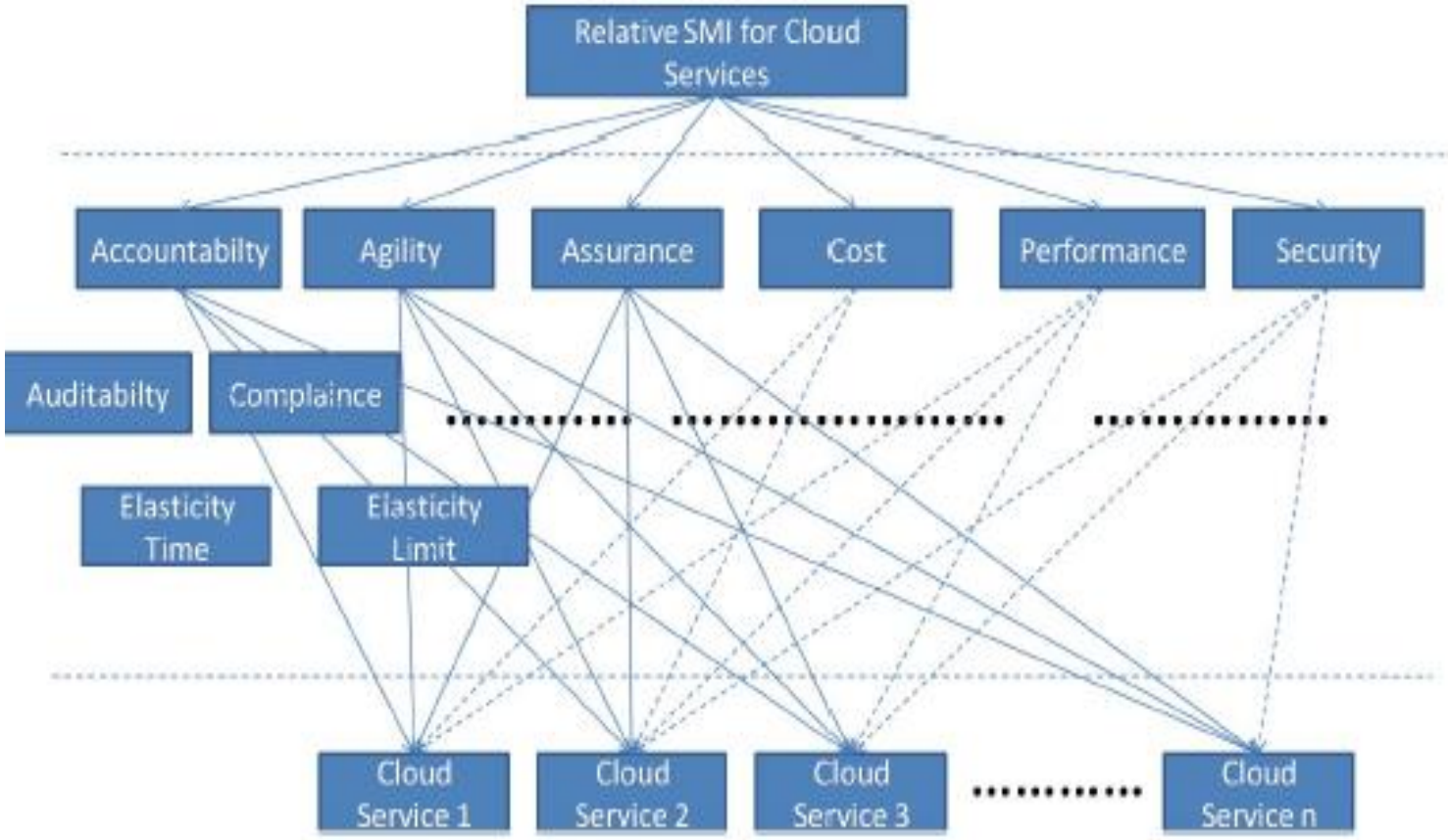
Quality Model For Iaas Provider

- Service Response Time
- Sustainability
- Suitability
- Accuracy
- Transparency
- Interoperability
- Availability
- Reliability
- Stability
- Cost
- Adaptability
- Elasticity
- Usability

Service Ranking Using AHP

- Phase1: Hierarchy structure for Cloud Services based on SMI KPIs
- Phase2: Computation of relative weights of each QoS and service
- Phase 3: Relative value-based weights for ranking Cloud services
- Phase 4: Aggregation of relative ranking for each SMI attribute

Phase1: Hierarchy structure for Cloud Services based on SMI KPIs



Phase2: Computation of relative weights of each QoS and service

- User Assigned weight

RELATIVE IMPORTANCE VALUE

Equal importance/quality	1
Somewhat more important/better	3
Definitely more important/better	5
Much more important/better	7
Extremely more important/better	9

- Arbitrary user assigned weights

Phase 3: Relative value-based weights for ranking Cloud services

- To compare both values for each Cloud service,
- Firstly, dimensional units of both the values are same.
- Secondly, compare the two values based on their types since the attributes values can vary
- from Boolean to an unordered set.
- Thirdly, users can specify essential and non-essential attributes.
- Ranking Model

Boolean:

$$
\begin{aligned}
s_i / s_j &= 1 \ \textit{if} \ v_i \equiv v_j \\
&= w_q \ \textit{if} \ v_j = 1 \ \textit{and} \ v_i = 0 \\
&= 1/w_q \ \textit{if} \ v_j = 0 \ \textit{and} \ v_i = 1
\end{aligned}
$$

Phase 3: Relative value-based weights for ranking Cloud services

- **Numeric:**
  - It can be of two types, higher is better or lower is better.
  - If higher is better than it is v_i / v_j is the value of s_i / s_j .
  - If lower value is better, v_i / v_j is the value of s_i / s_j .
  - If q is a non-essential requirement, then it may be possible that one service may not have a value. In that case, s_i / s_j is equal to w_q if v_j is given otherwise it will be $1/w_q$.

Phase 3: Relative value-based weights for ranking Cloud services

- **Unordered Set:**
  - if q is essential:

$$s_i/s_j = \frac{size(v_i)}{size(v_j)}$$

  - if q is non-essential and if v_r is specified:

$$s_i/s_j = \frac{size(v_i \cap v_r)}{size(v_j \cap v_r)} \; if \; v_j \cap v_r \neq \phi \wedge v_i \cap v_r \neq$$

$$= 1 \; if \; v_j \cap v_r \neq \phi \wedge v_i \cap v_r \equiv \phi$$
$$= w_q \; if \; v_j \cap v_r \neq \phi \vee v_i \cap v_r \equiv \phi$$
$$= 1/w_q \; if \; v_j \cap v_r \equiv \phi \vee v_i \cap v_r \neq \phi$$

Phase 3: Relative value-based weights for ranking Cloud services

- **Range Type:**
  - if q is essential:

$$s_i/s_j = \frac{len(v_i \cap v_r)}{len(v_j \cap v_r)}$$

  - if q is non-essential and if v_r is specified:

$$s_i/s_j = \frac{len(v_i \cap v_r)}{len(v_j \cap v_r)} \; if \; v_j \cap v_r \neq \phi \wedge v_i \cap v_r \neq$$

$$= 1 \; if \; v_j \cap v_r \equiv \phi \wedge v_i \cap v_r \equiv \phi$$
$$= w_q \; if \; v_j \cap v_r \neq \phi \wedge v_i \cap v_r \equiv \phi$$
$$= \frac{1}{w_q} \; if \; v_j \cap v_r \equiv \phi \wedge v_i \cap v_r \neq \phi$$

Phase 4: Aggregation of relative ranking for each SMI attribute

- Relative ranking vectors of each attribute are aggregated with their relative weights assigned in Phase2
- This aggregation process is repeated up for all the attributes in the SMI hierarchy which results in the final ranking of all the Cloud Services.

Case Study: Ranking Compute Cloud Services Based On User Qos Requirements

- Amazon EC2
- Windows Azure
- Rackspace
- The unavailable data such as security level is randomly assigned to each Cloud service
- User weights are also randomly assigned to each QoS service attribute.
- The top level QoS groups are
  - Accountability
  - Agility
  - Assurance
  - Cost
  - Performance
  - Security.

QUALITY MODEL FOR IAAS PROVIDER

Top level QoS Groups (Weights)	First level Attributes (Weights)	Second Level Attributes (Weights)		Service 1 (S1)	Service 2 (S2)	Service 3 (S3)	Value Type	User Required Value
Accountability (.05)	level:0-10 (1)			4	8	4	Numeric	4
Agility (0.1)	Capacity (0.6)	CPU (0.5)	0.5	9.6	12.8	8.8	Numeric	4x1.6 GHZ
		Memory (0.3)	0.3	15	14	15	Numeric	10 GB
		Disk (0.2)	0.2	1690	2,040	630	Numeric	500 GB
	Elasticity (.4)	Time (1)	0.4	80-120	520-780	20-200	Range	60-120 sec
Assurance (0.2)	Availability (0.7)	0.7		99.95%	99.99%	100%	Numeric	99.9%
	Service Stability (0.2)	Upload Time (0.3)	0.3	13.6	15	21	Numeric	
		CPU (0.4)	0.4	17.9	16	23	Numeric	
		Memory (0.3)	0.3	7	12	5	Numeric	
	Serviceabilty (0.1)	Free Support (0.7)	0.7	0	1	1	Boolean	
		Type of Support (0.3)	0.3	24/7,Diagnostic Tools, Phone, Urgent Response	24/7,Diagnostic Tools, Phone, Urgent Response	24/7, Phone, Urgent Response	Unordered set	24/7, phone
Cost (0.3)	On-Going Cost (1)	VM Cost (0.6)	0.6	0.68	$0.96	0.96	Numeric	< 1 dollar/hour
		Data (0.2)	inbound	10	10	8	Numeric	100 GB/month
			outboun d	11	15	18		200 GB/month
		Storage (0.2)	0.2	12	15	15	Numeric	1000 GB
Performance (0.3)	Service Response Time (1)	Range (0.5)	0.5	80-120	520-780	20-200	Range	60-120 sec
		Average Value (0.5)	0.5	100	600	30	Numeric	
Security (0.05)	level: 0-10 (1)			4	8	4	Numeric	4

- Security

$$RSRM_{security} = \begin{array}{c|ccc} & S1 & S2 & S3 \\ S1 & 1 & 4/8 & 4/4 \\ S2 & 8/4 & 1 & 8/4 \\ S3 & 4/4 & 4/8 & 1 \end{array}$$

Computing the Relative Service Ranking Vector (RSRV) for security from the matrix $RSRM_{security}$, we have

$RSRV_{security}$=[0.25 0.5 0.25]

- **Accountability:** $RSRV_{Accountability}$=[0.25 0.5 0.25]

Agility

- ***Elasticity***

$RSRV_{Elasticity}$=[0.3470 0.1991 0.4538]

- **Capacity**

$$RSRV_{capacity} = \begin{pmatrix} 0.30769 & 0.34090 & 0.36234 \\ 0.41025 & 0.31818 & 0.43738 \\ 0.28205 & 0.34090 & 0.20026 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0.3 \\ 0.2 \end{pmatrix}$$

Therefore,

$$RSRV_{capacity} = \begin{pmatrix} 0.3286 & 0.3881 & 0.2834 \end{pmatrix}$$

Similarly, the relative service ranking vector for Agility is given by:

$$RSRV_{agility} = \begin{pmatrix} 0.3286 & 0.34701 \\ 0.3881 & 0.19914 \\ 0.2834 & 0.45384 \end{pmatrix} \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix}$$

$$RSRV_{agility} = \begin{pmatrix} 0.336 & 0.3125 & 0.3516 \end{pmatrix}$$

- ***Assurance***

$$RSRV_{assurance} = \begin{pmatrix} 0.3812 & 0.2671 & 0.3517 \end{pmatrix}$$

- **Cost**

$$RSRV_{Cost} = \begin{pmatrix} 0.4073 & 0.3338 & 0.2589 \end{pmatrix}$$

- **Performance**

$$RSRV_{performance} = \begin{pmatrix} 0.2846 & 0.1181 & 0.5973 \end{pmatrix}$$

Relative Service Matrix for Three Providers

$$RSRM = \begin{pmatrix} 0.25 & 0.336 & 0.3812 & 0.1619 & 0.2846 & 0.25 \\ 0.5 & 0.3125 & 0.2671 & 0.1308 & 0.1181 & 0.5 \\ 0.25 & 0.3516 & 0.3517 & 0.7073 & 0.5973 & 0.25 \end{pmatrix}$$

Relative ranking of all the Cloud service

$$RSRV = \begin{pmatrix} 0.25 & 0.336 & 0.3812 & 0.4073 & 0.2846 & 0.25 \\ 0.5 & 0.3125 & 0.2671 & 0.3338 & 0.1181 & 0.5 \\ 0.25 & 0.3516 & 0.3517 & 0.2589 & 0.5973 & 0.25 \end{pmatrix} \begin{pmatrix} 0.05 \\ 0.1 \\ 0.2 \\ 0.3 \\ 0.3 \\ 0.05 \end{pmatrix}$$

Resultant RSRV (0.3424, 0.2702, 0.3874)

- The Cloud services are ranked as *S*3 >*S*1> *S*2.

Web Services, Service Oriented Architecture

What are “ Web Services”?

- “Software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts” – W3C Web Services Architecture Requirements, Oct. 2002
- “Programmable application logic accessible using Standard Internet Protocols…” – Microsoft
- “An interface that describes a collection of operations that are network accessible through standardized XML messaging …” – IBM
- “Software components that can be spontaneously discovered, combined, and recombined to provide a solution to the user’s problem/request … ” - SUN

History!

- Structured programming
- Object-oriented programming
- Distributed computing
- Electronic Data Interchange (EDI)
- World Wide Web
- Web Services

Distributed Computing

- When developers create substantial applications, often it is more efficient, or even necessary, for different task to be performed on different computers, called N-tier applications:
- A 3-tier application might have a user interface on one computer, business-logic processing on a second and a database on a third – all interacting as the application runs.
- For distributed applications to function correctly, application components, e.g. programming objects, executing on different computers throughout a network must be able to communicate. E.g.: DCE, CORBA, DCOM, RMI etc.
- Interoperability:
  - Ability to communicate and share data with software from different vendors and platforms
  - Limited among conventional proprietary distributed computing technologies

Electronic Data Interchange (EDI)

- Computer-to-computer exchange of business data and documents between companies using standard formats recognized both nationally and internationally.
- The information used in EDI is organized according to a specified format set by both companies participating in the data exchange.
- Advantages:
  - Lower operating costs
    - Saves time and money
  - Less Errors => More Accuracy
    - No data entry, so less human error
  - Increased Productivity
    - More efficient personnel and faster throughput
  - Faster trading cycle
    - Streamlined processes for improved trading relationships

Web Services

- Take advantage of OOP by enabling developers to build applications from existing software components in a modular approach:
    - Transform a network (e.g. the Internet) into one library of programmatic components available to developers to have significant productivity gains.
- Improve distributed computing interoperability by using open (non-proprietary) standards that can enable (theoretically) any two software components to communicate:
    - Also they are easier to debug because they are text-based, rather than binary, communication protocols

Web Services …Contd.

- Provide capabilities similar to those of EDI (Electronic Data Interchange), but are simpler and less expensive to implement.
- Configured to work with EDI systems, allowing organizations to use the two technologies together or to phase out EDI while adopting Web services.
- Unlike WWW
  - Separates visual from non-visual components
  - Interactions may be either through the browser or through a desktop client (Java Swing, Python, Windows, etc.)

Web Services …Contd.

- Intended to solve *three* problems:
- **Interoperability**:
  - Lack of interoperability standards in distributed object messaging
  - DCOM apps strictly bound to Windows Operating system
  - RMI bound to Java programming language
- **Firewall traversal**:
  - CORBA and DCOM used non-standard ports
  - Web Services use HTTP; most firewalls allow access though port 80 (HTTP), leading to easier and dynamic collaboration
- **Complexity**:
  - Web Services: developer-friendly service system
  - Use open, text-based standards, which allow components written in different languages and for different platforms to communicate
  - Implemented incrementally, rather than all at once which lessens the cost and reduces the organizational disruption from an abrupt switch in technologies

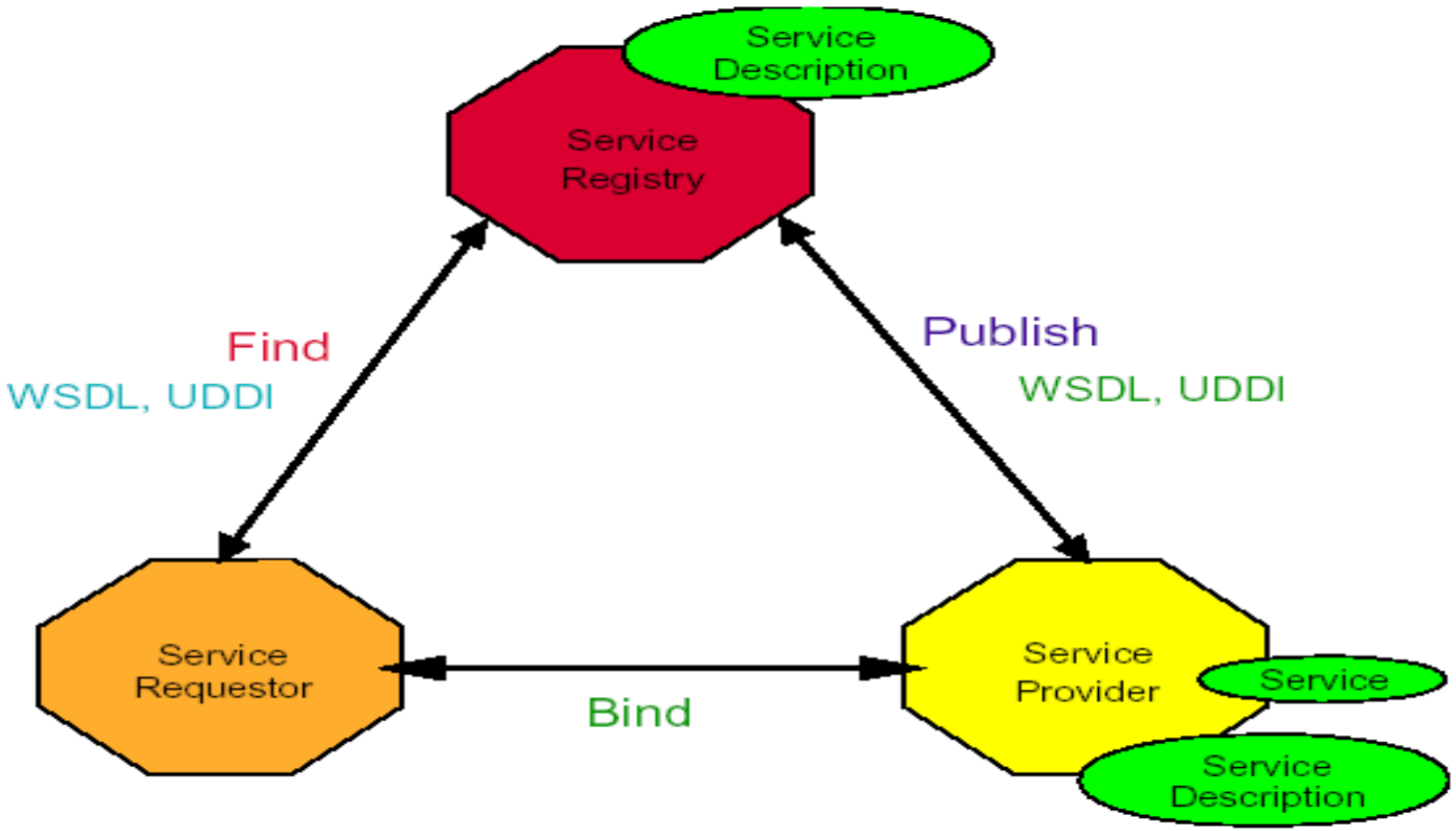
Web Service: Definition Revisited

- An application component that:
- Communicates via open protocols (HTTP, SMTP, etc.)
- Processes XML messages framed using SOAP
- Describes its messages using XML Schema
- Provides an endpoint description using WSDL
- Can be discovered using UDDI

Service Oriented Architecture (SOA)

- IBM has created a model to show Web services interactions which is referred to as a Service-Oriented Architecture (SOA) consisting of relationships between three entities:
  - A service provider;
  - A service requestor;
  - A service broker
- IBM's SOA is a generic model describing service collaboration, not just specific to Web services.
  - See: http://www-106.ibm.com/developerworks/webservices/

Web Service Model



Web Service Model *(contd…)*

- Roles in Web Service architecture
- Service provider
  - Owner of the service
  - Platform that hosts access to the service
- Service requestor
  - Business that requires certain functions to be satisfied
  - Application looking for and invoking an interaction with a service
- Service registry
  - Searchable registry of service descriptions where service providers publish their service descriptions

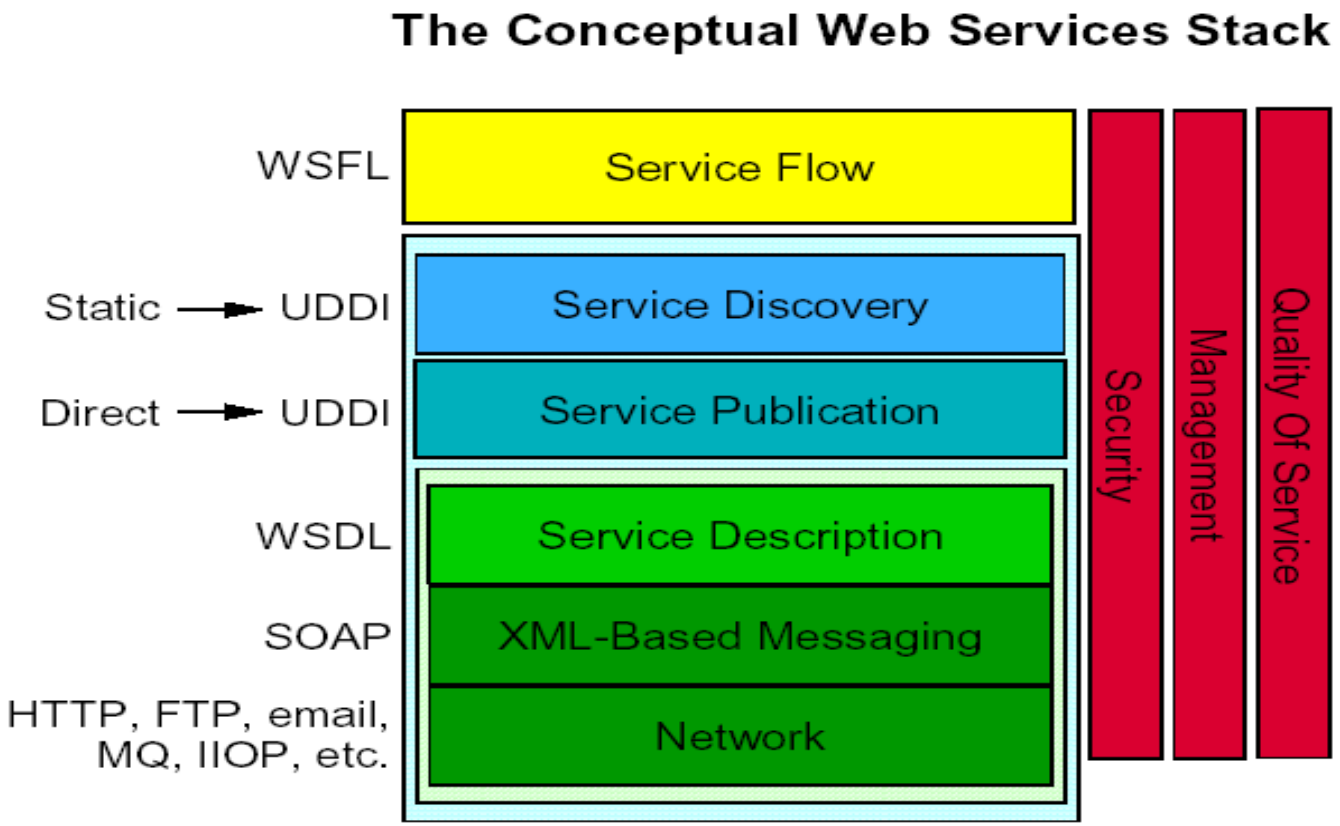
Web Service Model *(contd...)*

- Operations in a Web Service Architecture
- Publish
  - Service descriptions need to be published in order for service requestor to find them
- Find
  - Service requestor retrieves a service description directly or queries the service registry for the service required
- Bind
  - Service requestor invokes or initiates an interaction with the service at runtime

Web Service Components

- **XML** – eXtensible Markup Language
  - A uniform data representation and exchange mechanism.
- **SOAP** – Simple Object Access Protocol
  - A standard way for communication.
- **WSDL** – Web Services Description Language
  - A standard meta language to described the services offered.
- **UDDI** – Universal Description, Discovery and Integration specification
  - A mechanism to register and locate WS based application.

Web Service Stack



XML

- Developed from Standard Generalized Markup Method (SGML)
- Widely supported by W3C
- Essential characteristic is the separation of content from presentation
- Designed to describe **data**
- XML document can optionally reference a *Document Type Definition (DTD),* also called a *Schema*
  - XML parser checks syntax
  - If an XML document adheres to the structure of the schema it is *valid*
- XML tags are not predefined
  - You must **define your own tags.**
- Enables cross-platform data communication in Web Services

SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- Format for sending messages over Internet between programs
- XML-based
- Platform and language independent
- Simple and extensible
- Uses mainly HTTP as a transport protocol
  - HTTP message contains a SOAP message as its payload section
- Stateless, one-way
  - But applications can create more complex interaction patterns

SOAP Message Structure

- Request and Response messages
  - Request invokes a method on a remote object
  - Response returns result of running the method
- •SOAP specification defines an “envelop”
  - “envelop” wraps the message itself
  - Message is a different vocabulary
  - Namespace prefix is used to distinguish the two parts

Why SOAP?

- Other distributed technologies failed on the Internet
  - Unix RPC – requires binary-compatible Unix implementations at each endpoint
  - CORBA – requires compatible ORBs
  - RMI – requires Java at each endpoint
  - DCOM – requires Windows at each endpoint
- SOAP is the platform-neutral choice
  - Simply an XML wire format
  - Places no restrictions on the endpoint implementation technology choices

SOAP Characteristics

- SOAP has three major characteristics:
- Extensibility – security and WS-routing are among the extensions under development.
- Neutrality - SOAP can be used over any transport protocol such as HTTP, SMTP or even TCP.
- Independent - SOAP allows for any programming model.

SOAP Usage Models

- RPC-like message exchange
  - Request message bundles up method name and parameters
  - Response message contains method return values
  - However, it isn’t required by SOAP
- SOAP specification allows any kind of body content
  - Can be XML documents of any type
  - Example:
    - Send a purchase order document to the inbox of B2B partner
    - Expect to receive shipping and exceptions report as response

SOAP Security

- SOAP uses HTTP as a transport protocol and hence can use HTTP security mainly HTTP over SSL.
- But, since SOAP can run over a number of application protocols (such as SMTP) security had to be considered.
- The *WS-Security specification* defines a complete encryption system.

WSDL - Web Service Definition Language

- WSDL : XML vocabulary standard for describing Web services and their capabilities
- Contract between the XML Web service and the client
- Specifies what a request message must contain and what the response message will look like in unambiguous notation
- Defines where the service is available and what communications protocol is used to talk to the service.

WSDL Document Structure

- A WSDL document is just a simple XML document.
- It defines a web service using these major elements:
  - **port type** - The operations performed by the web service.
  - **message -** The messages used by the web service.
  - **types -** The data types used by the web service.
  - **binding -** The communication protocols used by the web service.

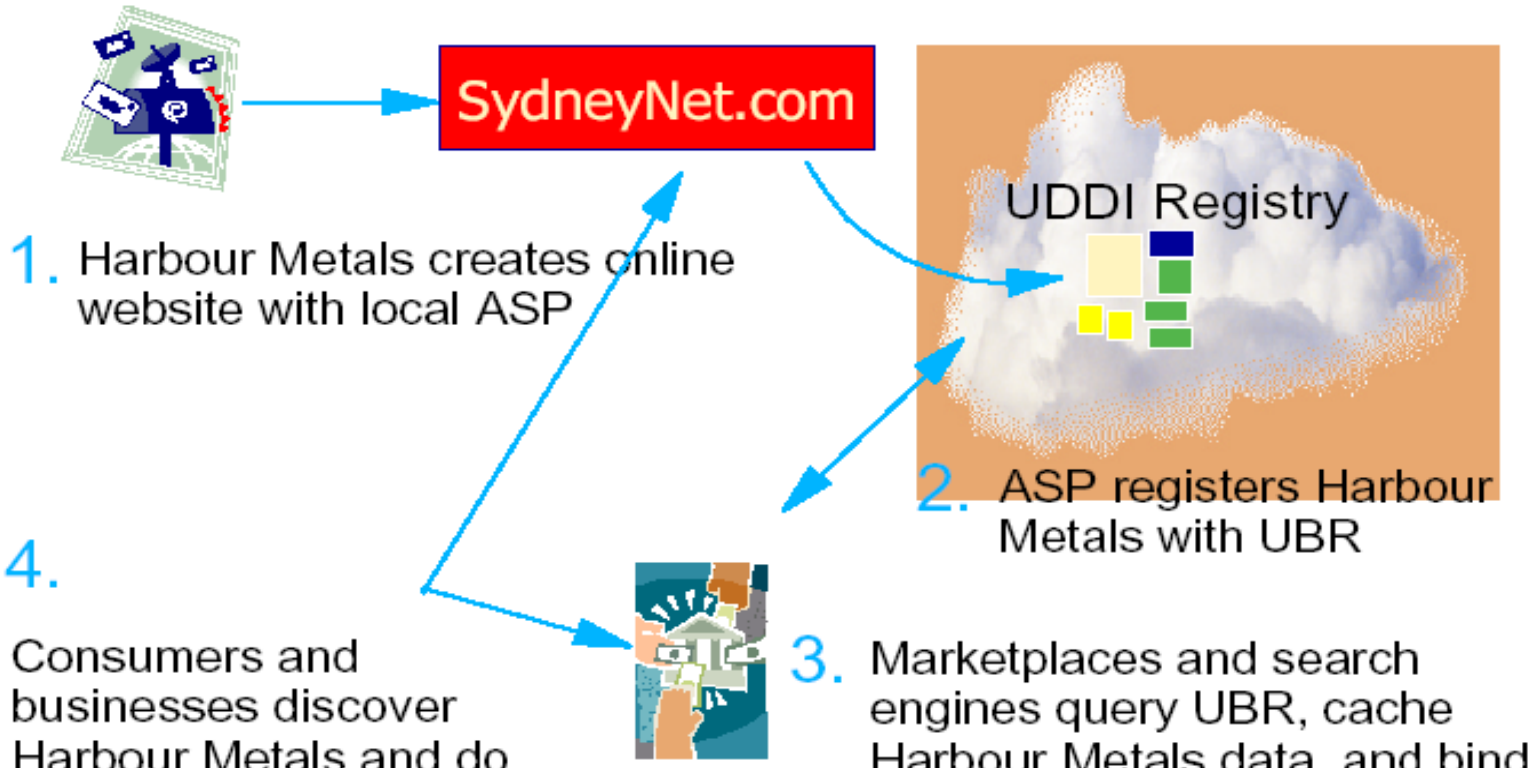
UDDI - Universal Description, Discovery, and Integration

- A framework to define XML-based registries
- Registries are repositories that contain documents that describe business data and also provide search capabilities and programmatic access to remote applications
- Businesses can publish information about themselves and the services they offer
- Can be interrogated by SOAP messages and provides access to WSDL documents describing web services in its directory

UDDI Roles and Operations

- Service Registry
  - Provides support for publishing and locating services
  - Like telephone yellow pages
- Service Provider
  - Provides e-business services
  - Publishes these services through a registry
- Service requestor
  - Finds required services via the Service Broker
  - Binds to services via Service Provider

How can UDDI be Used?



UDDI Benefits

- Making it possible to discover the right business from the millions currently online
- Defining how to enable commerce once the preferred business is discovered
- Reaching new customers and increasing access to current customers
- Expanding offerings and extending market reach