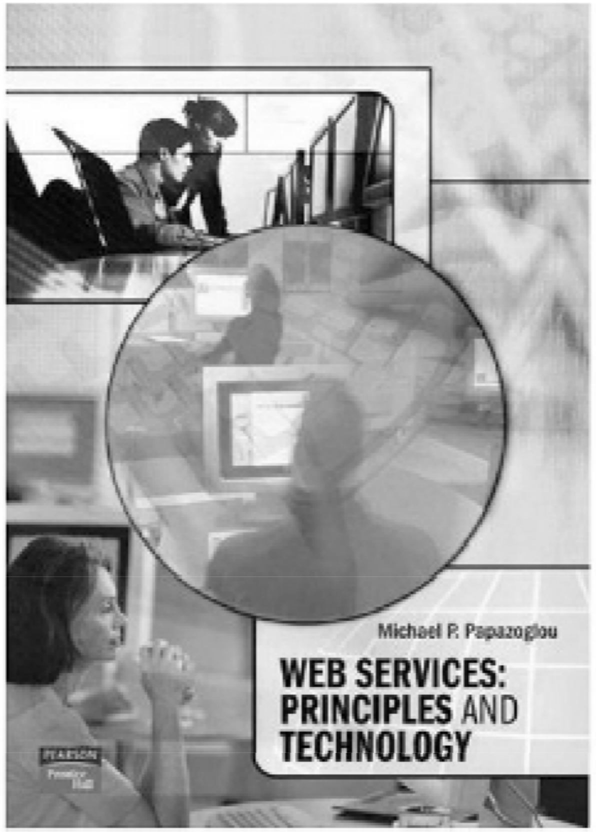
Chapter 1 Web Services Basics



Mike P. Papazoglou
mikep@uvt.nl

UNIVERSITEIT VAN TILBURG

Topics

- ***Introduction – definitions***
- *Software as a service*
- *Where can services be used?*
- *Characteristics of web services*
- *Web services interfaces and implementations*
- *Service-oriented architecture*
- *Web services technology stack*
- *Quality of service*
- *Impact and shortcomings of Web services*

Service-Oriented Paradigm

- The service-oriented paradigm to programming utilizes *services* as the constructs to support the development of rapid, low-cost and easy composition of distributed applications.
- A Web service is a ***programmatically available application logic exposed over the Internet.***
  - Any piece of code and any application component deployed on a system can be transformed into a network-available service.
- Services reflect a new ‘service-oriented’ approach to programming, based on the idea of composing applications by discovering and invoking network-available services rather than building new applications or by invoking available applications to accomplish some task.

What are Web Services?

A Web service: a self-describing, self-contained software module available via a network, which completes tasks, solves problems, or conducts transactions on behalf of a user or application

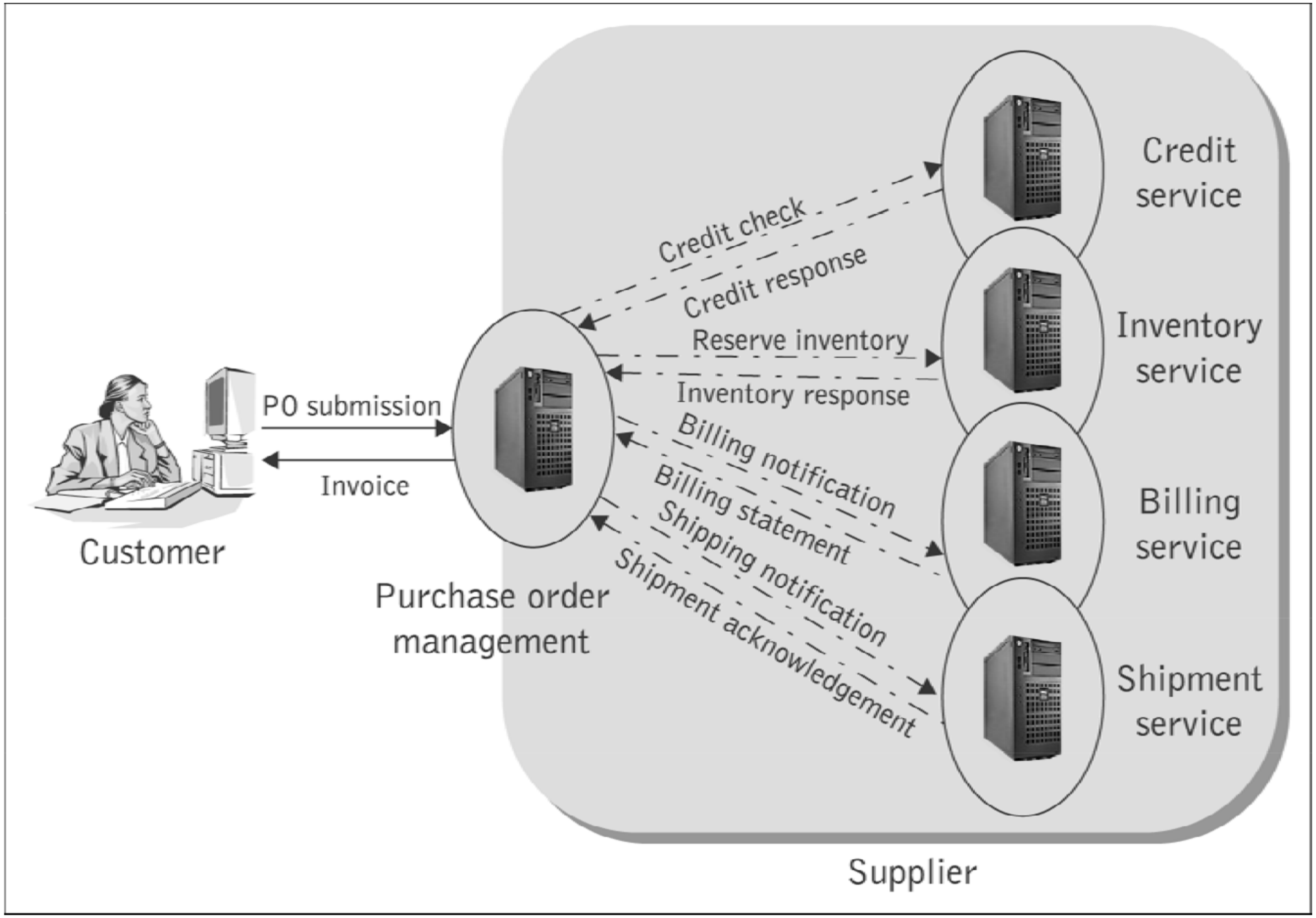
Web services perform encapsulated business functions such as:

1. a self-contained business task – a funds withdrawal or funds deposit service;
2. a full-fledged business process – the automated purchasing of office supplies;
3. an application – a life insurance application or demand forecasts and stock replenishment; or
4. a service-enabled resource – access to a particular back-end database containing patient medical records.

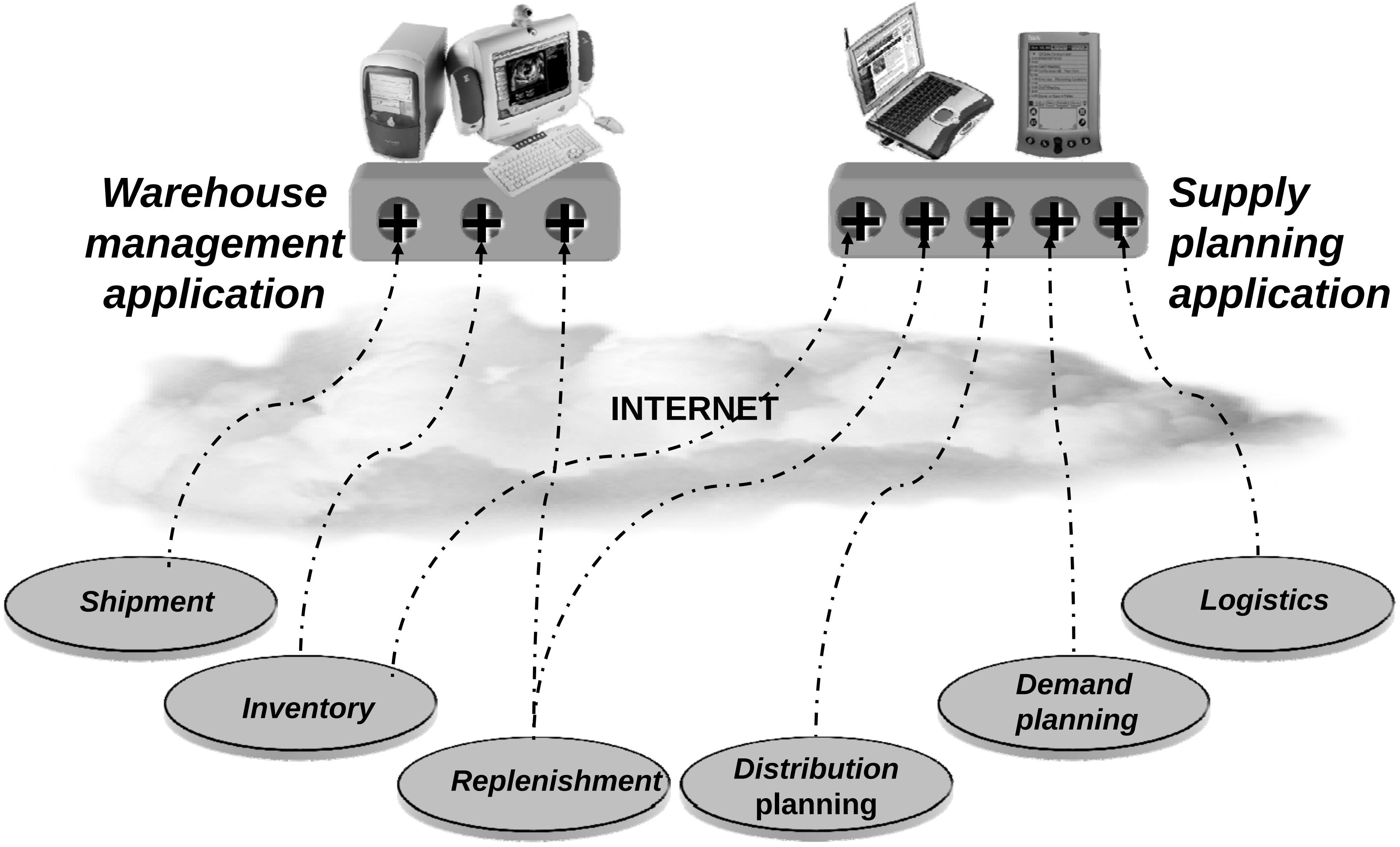
What are Web Services?

- Can be mixed and matched to create a complete process:
  - Enable integration with decreased human interaction, e.g., create complete enterprise processes, such as supply chain management, procurement, logistics, etc.
  - Both new and extensions to existing applications
- Available to a variety of clients (platform independent)
- Pricing (rating) model determines subscriber rates based on subscription and usage events, e.g., calculate charges for services based on the quality and precision of the service.
- Billing model:
  - “pay per use”, “lease it”, “pay for it” basis.

A purchase order application involving interacting web services



A Supply Chain application



Topics

- *Introduction – definitions*
- ***Software as a service***
- *Where can services be used?*
- *Characteristics of web services*
- *Web services interfaces and implementations*
- *Service oriented architecture*
- *Web services technology stack*
- *Quality of service*
- *Impact and shortcomings of web services*

Application Service Providers

- The concept of software-as-a-service is evolutionary and appeared first with the ASP (application service provider) software model:

  An ASP "rents" applications to subscribers.

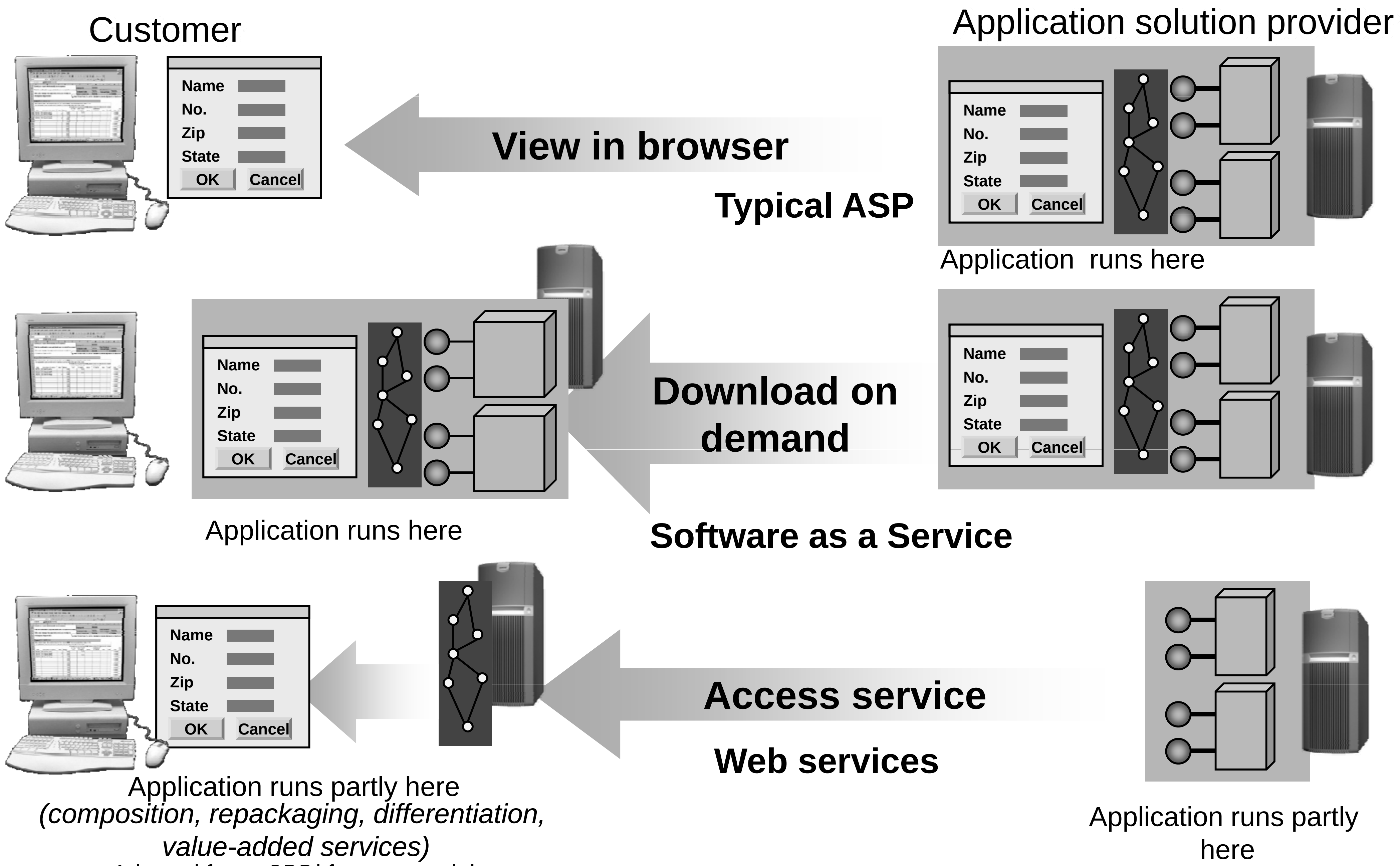
  - The whole application is developed in terms of the user interface, workflow, business, and data components that are all bound together to provide a working solution.
  - An ASP hosts the entire application and the customer has little opportunity to customize it beyond the appearance of the user interface, e.g., adding company logos.
  - An alternative of this is where the ASP is providing a software module that is downloaded to the customer's site on demand – this is for situations where the software does not work in a client/server fashion.

Source: Includes Everware-CBDI copyright materials

ASP vs. Web Services

- Although the ASP model introduced the concept of software-as-a-service first, it suffered from several inherent limitations such as
  - inability to develop highly interactive applications;
  - inability to provide complete customizable applications and;
  - inability to integrate applications.
- Today we are in the midst of another significant development in the evolution of software-as-a-service. The new architecture allows for:
  - loosely coupled asynchronous interactions
  - on the basis of eXtensible Markup Language (XML) standards
  - with the intention of making access to, and communications between, applications over the Internet easier, by means of appropriate standards.

Are ASP, Software-as-a-Service, and Web Service the same?



Adapted from: CBDi forum copyright Everware-CBDI

Topics

- *Introduction – definitions*
- *Software as a service*
- ***Where can services be used?***
- *Characteristics of web services*
- *Web services interfaces and implementations*
- *Service oriented architecture*
- *Web services technology stack*
- *Quality of service*
- *Impact and shortcomings of web services*

What problems do we solve?

Broader B2B

A mid-sized manufacturer needs to create online relationships with customers, each with their own set of standard and protocols.

Search capabilities

To meet its on-time delivery commitments, a high-tech manufacturer needs to place orders with the most advantageous parts manufacturer or assembler.

Easier aggregation

A high-tech manufacturer needs to easily integrate and synchronize third-party providers with its manufacturing and distribution requirements.

Describe services

Discover services

Integrate services together

Where are Services used?

- **Within an enterprise (enterprise application integration)**

➔Accelerate and reduce the cost of integration
➔Save on infrastructure deployment and management costs
➔Reduce skill requirements
➔Improve reuse

- **Between enterprises (e-Business integration)**

➔Providing service to a company's customers
  – **e.g., an Insurance company wishes to link its systems to the systems of a new institutional customer**

➔Accessing services from a company's partners and suppliers
  – **e.g., dynamically link to new partners and suppliers to offer their services to complement the value the company provides**

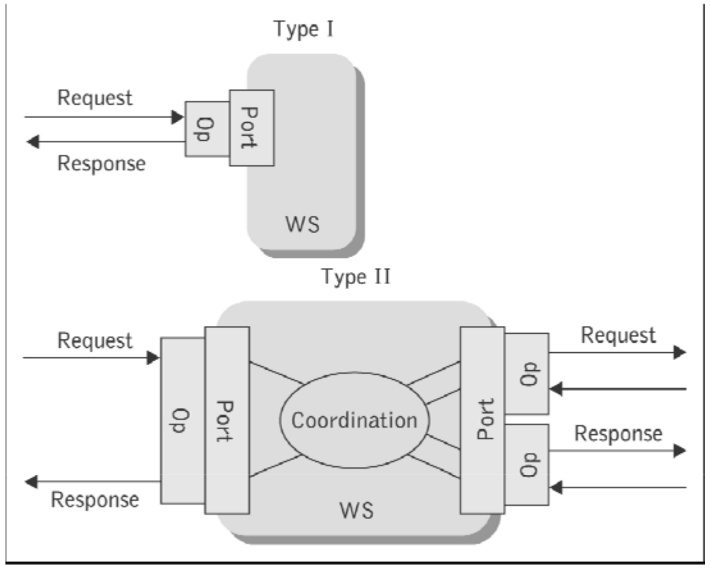
– **Standards and common infrastructure reduce the barriers**
– **Simplicity accelerates deployment**
– **Dynamics opens new business opportunities**

Topics

- *Introduction – definitions*
- *Software as a service*
- *Where can services be used?*
- ***Characteristics of web services***
- *Web services interfaces and implementations*
- *Service oriented architecture*
- *Web services technology stack*
- *Quality of service*
- *Impact and shortcomings of web services*

Types of Web Services

- Informational services are services of relatively simple nature. They either provide access to content interacting with an end-user by means of simple request/response sequences, or expose back-end business applications to other applications. Examples include:
  - content services such as weather report info., simple financial info., stock quote info., news items
  - simple trading services that can provide a seamless aggregation of information across disparate systems and data sources, e.g., logistic services
  - Information syndication services: value-added info Web services plugging into commerce sites, e.g., reservation services on travel site



- Complex services that involve the assembly and invocation of many pre-existing services possibly found in diverse enterprises to complete a multi-step business interaction:
  - a supply-chain application involving order taking, stocking orders, sourcing, inventory control, financials, and logistics.

Service Properties and State

- Functional and non-functional properties:
  - The functional service description details the operational characteristics that define the overall behavior of the service., e.g., how to invoke service, and location where it’s invoked
  - The non-functional description targets service quality attributes, e.g., service metering and cost, performance metrics (response time or accuracy), security, authorization, authentication, scalability, availability, etc.
- Stateless or stateful services:
  - Services that can be invoked repeatedly without having to maintain context or state they are called stateless.
    - Simple informational services are stateless.
  - Services that require their context to be preserved from one invocation to the next are called stateful.
    - Complex services (business processes) typically involve stateful interactions.

Loose Coupling and Granularity

- Loose coupling:
  - Coupling indicates the degree of dependency any two systems have on each other.
  - Web services are loosely coupled: they connect and interact more freely (across the Internet). They need not know how their partner applications behave or are implemented.
- Service granularity:
  - Simple services are discrete in nature, exhibit normally a request/reply mode of operation and are of fine granularity, i.e., they are atomic in nature.
  - Complex services are coarse-grained, e.g., a SubmitPurchaseOrder process. These involve interactions with other services and possibly end-users in a single or multiple sessions.
  - Coarse-grained communication implies larger and richer data structures (viz. those supported by XML schema), and enables looser coupling, which enables asynchronous communication

Synchronicity and Well-definedness

- Sychronicity: There are two programming styles for services:
  - Synchronous or remote procedure call (RPC)-style: Clients of synchronous services express their request as a method call with a set of arguments, which returns a response containing a return value.
    - Simple informational services, e.g., returning the current price for a given stock; providing the current weather conditions in a particular location; or checking the credit rating of a potential trading partner.
  - Asynchronous or message (document)-style: When a client invokes a message-style service, it typically sends it an entire document, e.g., a purchase order, rather than a discrete set of parameters.
    - Business processes, e.g., processing a purchase order; responding to a request for quote order from a customer; or responding to an order placement by a particular customer.
- Well-definedness:
  - The service interaction must be well defined. The web services description language (WSDL) allows applications to describe the rules for interfacing and interacting to other applications.

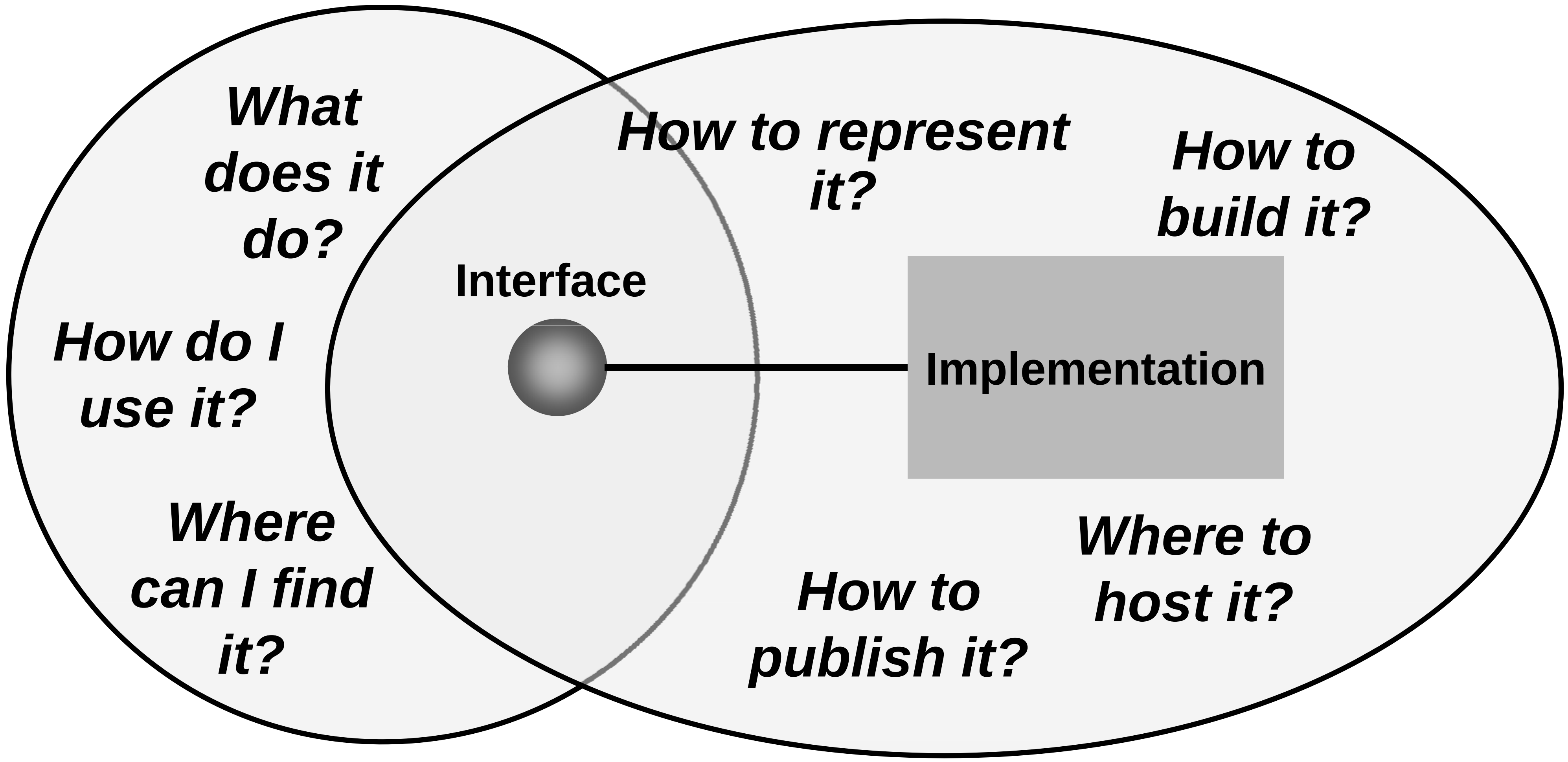
Topics

- *Introduction – definitions*
- *Software as a service*
- *Where can services be used?*
- *Characteristics of web services*
- ***Web services interfaces and implementations***
- *Service oriented architecture*
- *Web services technology stack*
- *Quality of service*
- *Impact and shortcomings of web services*

Service Interface and Implementation

- The service interface defines service functionality visible to the external world and provides the means to access this functionality.
  - The service describes its own interface characteristics, i.e., the operations available, the parameters, data-typing and the access protocols, in a way that other software modules can determine what it does, how to invoke its functionality, and what result to expect in return.
- The service implementation realizes a specific service interface whose implementation details are hidden from its users.
  - Different service providers using any programming language of their choice may implement the same interface.
  - One service implementation might provide functionality itself directly, while another implementation might use combination of other services to provide the same functionality

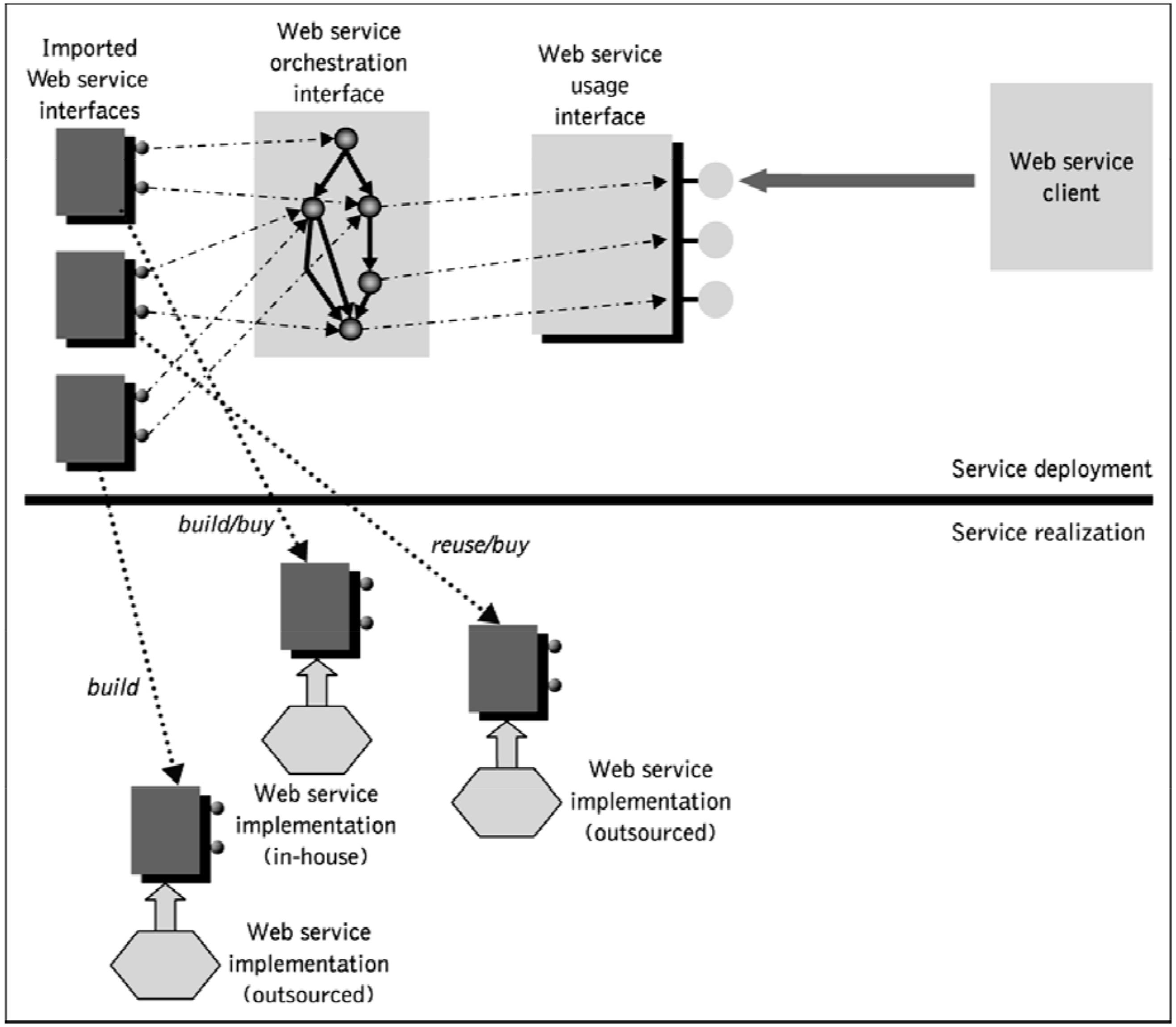
Client vs. provider perspective



Client perspective

Provider perspective

Service deployment vs. service realization


Imported Web service interfaces
Web service orchestration interface
Web service usage interface
Web service client
Service deployment
Service realization
build/buy
reuse/buy
build
Web service implementation (in-house)
Web service implementation (outsourced)
Web service implementation (outsourced)

Topics

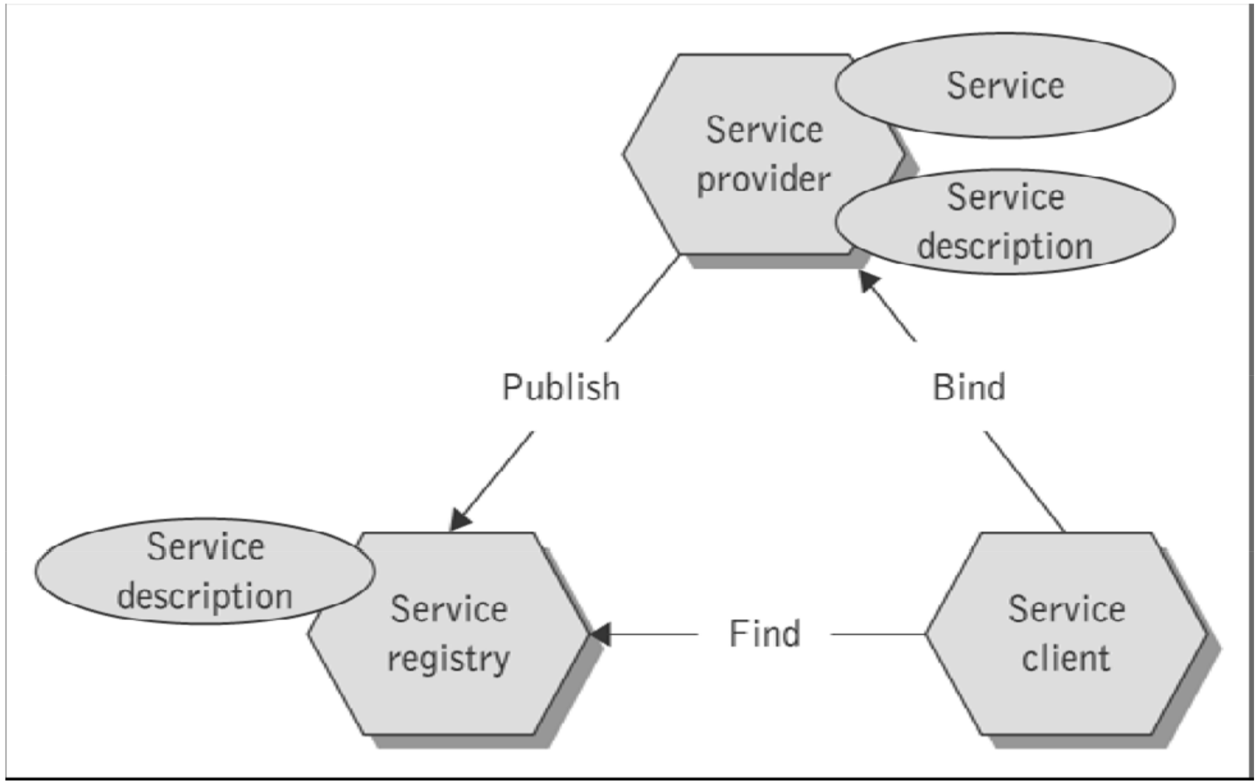
- *Introduction – definitions*
- *Software as a service*
- *Where can services be used?*
- *Characteristics of web services*
- *Web services interfaces and implementations*
- ***Service oriented architecture***
- *Web services technology stack*
- *Quality of service*
- *Impact and shortcomings of web services*

Roles

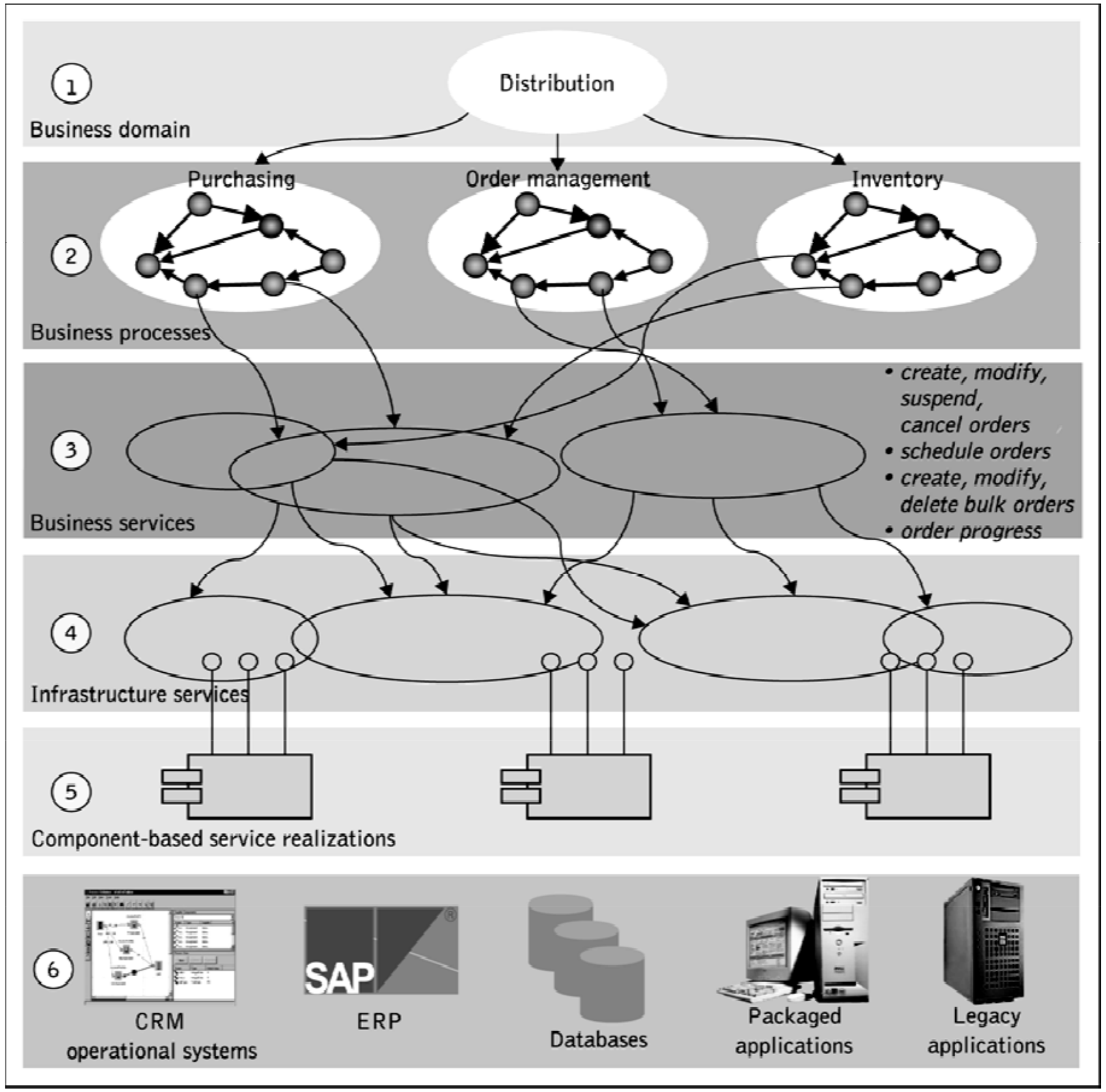
- The service model allows for a clear distinction to be made between
  - ***service providers*** (organizations that provide the service implementations, supply their service descriptions, and provide related technical and business support);
  - ***service clients*** (end-user organizations that use some service);
  - ***service registry*** (a searchable directory where service descriptions can be published and searched).
    - Service requestors find service descriptions in the registry and obtain binding information for services.
    - This information is sufficient for the service requestor to contact, or bind to, the service provider and thus make use of the services it provides.

Service-Oriented Architecture

- SOA is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable interfaces.



Layers in an SOA

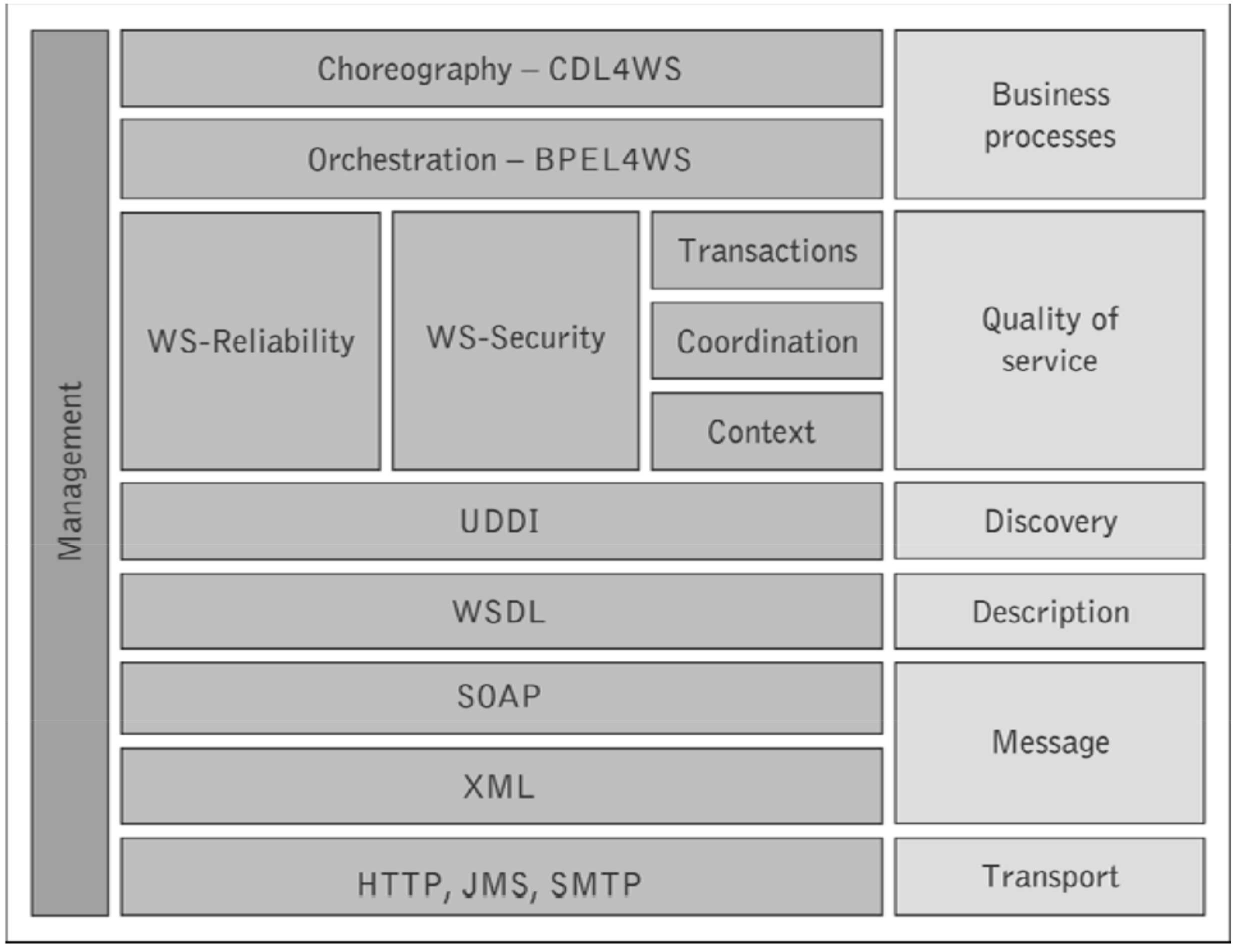


Topics

- *Introduction – definitions*
- *Software as a service*
- *Where can services be used?*
- *Characteristics of web services*
- *Web services interfaces and implementations*
- *Service oriented architecture*
- ***Web services technology stack***
- *Quality of service*
- *Impact and shortcomings of web services*

Web Services Technology Stack

- Web services are implemented by a collection of several related technologies and standards.



Topics

- *Introduction – definitions*
- *Software as a service*
- *Where can services be used?*
- *Characteristics of web services*
- *Web services interfaces and implementations*
- *Service oriented architecture*
- *Web Services technology stack*
- ***Quality of service***
- *Impact and shortcomings of web services*

Quality of Service (QoS)

- QoS refers to the ability of a Web service to respond to expected invocations and perform them at the level commensurate to the mutual expectations of both its provider and its customers.
- The key QoS in a Web services environment are:
  - Availability: absence of service downtimes
  - Accessibility: degree with which Web service request is served, e.g., success rate
  - Conformance to standards: compliance of Web service with standards, e.g., WSDL 2.0
  - Integrity: degree with which Web service performs its tasks according to its WSDL and conformance with SLA
  - Performance: measured in terms of throughput and latency
  - Reliability: ability to function correctly and consistently and provide same service quality despite system or network failures

Quality of Service (QoS)

- Scalability: ability to consistenly serve requests despite variations in volume of requests
- Security: involves authentication, authorization, message integrity, and confidentiality
- Transactionality: requirement of transactional behavior and context propagation

Service Level Agreements (SLAs)

- An SLA is a formal agreement (contract) between a provider and client, formalizing the details of use of a Web service (contents, price, delivery process, acceptance and quality criteria, penalties, etc in measurable terms) in a way that meets the mutual understandings and expectations of both providers and clients.
- An SLA may contain the following parts:
  - Purpose: describes reasons behind the creation of the SLA
  - Parties: describes parties involved in the SLA and their respective roles, e.g., service provider and service consumer (client)
  - Validity period: defines period of time that SLA will cover
  - Scope: defines services covered in the agreement
  - Restrictions: defines necessary steps to be taken in order for the requested service levels to be provided
  - Service-level objectives: defines levels of service both consumers and providers agree on, and include a set of service level indicators, e.g., availability, performance, and reliability

Service Level Agreements (SLAs)

- Penalties: defines what sanctions should apply in case service provider underperforms and is unable to meet objectives specified in the SLA
- Exclusion terms: specify what’s not covered in the SLA
- Administration authority: describes the processes and measurable objectives in an SLA and defines organizational authority for overseeing them

Topics

- *Introduction – definitions*
- *Software as a service*
- *Where can services be used?*
- *Characteristics of web services*
- *Web services interfaces and implementations*
- *Service oriented architecture*
- *Web services technology stack*
- *Quality of service*
- ***Impact and shortcomings of web services***

Impact of Web Services

- The most appealing characteristic of Web services is that they are evolving to embrace the convergence of e-business, EAI, traditional middleware, and the Web. Web services offer:
  - a standard way to expose legacy application functionality as a set of reusable self-contained, self-describing services;
  - a standard, easy, and flexible way to help overcome application integration issues, leading to rapid application assembly;
  - a standard way to develop and/or assemble Internet-native applications for both the internal and the extended enterprise by using internally or externally created services as building blocks;
  - a common facade for cross-enterprise specific systems, making it easier to create the service-level agreements needed for business-to-business integration.

Pitfalls of Web Services

- As the business requirements that drive Web services become even more complex, Web services technologies require additional capabilities to handle demanding situations that severely test the most obvious current shortcomings of Web services. These include:
  - performance issues,
  - lack of appropriate support for sophisticated transaction management,
  - lack of expressing business semantics and, especially,
  - achieving a widespread agreement and harmonization on a wide range of existing and emerging standards.
    - There is an overwhelming number of existing and emerging standards.
    - Unless these standards mature enough to the degree that they are harmonized and can act together, the long-term viability of Web services applications being used in mission-critical, production environments will be severely tested.