

Seminar on Web Services and Web 2.0

Kenneth M. Anderson
University of Colorado, Boulder
CSCI 7818 — Lecture 1 — 08/27/2008

Credit Where Credit is Due

- Portions of this lecture are derived from material in “Web Services: Principles and Technology” by Michael P. Papazoglou and its accompanying instructors materials. As such, they are Copyright 2008 by Pearson/Prentice Hall

A bit about me...

- Associate Professor
- At CU since July 1998
- Ph.D. at UC Irvine (1997)
- Research Interests
 - Software Engineering
 - Hypermedia
 - Web Engineering



A little bit more...

- 21st Semester at CU
- 3rd time teaching a seminar on these topics
- Software Development Experience
 - Approximately 16 systems, 30K-100K LOC each
 - Some industry experience with IBM & Unisys
 - Experience with academic/industry collaboration
 - NCAR, NREL, Northrop Grumman, ioSemantics
- Have been working on hypermedia since 1991, first demo at Hypertext'93
- Watched the “birth” of the Web occur while sitting across from Roy Fielding and Jim Whitehead at UCI

Office Hours

- ... by appointment
- When meeting with me, we'll use either
 - ECCS 127 (across from the CSEL)
 - Faculty Lounge (across from ECOT 717)

Class Website


CSCI 7818 — Fall 2008

Web Services and Web 2.0


[HOME](#) [WHAT'S NEW](#) [LECTURES](#) [TEXTBOOKS](#) [ASSIGNMENTS](#) [SYLLABUS STATEMENTS](#)

Textbooks

This seminar will draw on two textbooks this semester.



[Web Services: Principles and Technology](#) by Michael P. Papazoglou. Published by Pearson/Prentice Hall. ISBN: 978-0-321-15555-9. This book provides a comprehensive overview of SOAP-based web services and the field of service-oriented architecture (SOA). It will provide us with the background that we need to examine the work going on in "big web services" and serve as a "jumping off point" for looking at the latest work on the Web that is being applied to the WS-* specifications, vendor toolsets, and the like.



[RESTful Web Services](#) by Leonard Richardson & Sam Ruby. Published by O'Reilly. ISBN-13: 978-0-596-52926-0. This book takes a deep look at the "other" approach to web services, known as RESTful web services. REST stands for Representation State Transfer and refers to a concept that appears in Roy Fielding's dissertation. Roy Fielding developed REST while working as a graduate student at U.C. Irvine under the direction of Richard N. Taylor. The RESTful approach to Web services tries to model Web-based services on the techniques that made the Web successful in the first place. This book will also provide us with the background knowledge we need to understand the work being applied to this approach to web services and how REST is being applied to well-known Web application frameworks, such as Ruby On Rails.

With this background, the seminar will endeavor to compare/contrast the two approaches and discuss when/where each technology should be used.

The seminar will also examine Web 2.0 design concepts but no textbook will be used for that portion of the seminar; instead we will draw on materials straight from the Web, including social networking sites, the writings of Tim O'Reilly, blog discussions, and the like.

© Kenneth M. Anderson, 2008

Class Info

Time: Wed. 4:00 PM – 6:30 PM
Location: ECST 1B21

Useful Links

[Professor's Home Page](#)
[Department of Computer Science](#)
[Class Moodle](#)

Created with Sandvox

<<http://www.cs.colorado.edu/~kena/classes/7818/f08/>>

About the Class Website

- Check the website *every day!*
 - An RSS feed of the What's New page is available
- The website is your source for
 - class schedule
 - homework assignments
 - announcements
 - etc.

Textbooks



Web Services: Principles and Technology



RESTful Web Services

Seminar on Web Services

- Present Historical Perspective of Distributed Information Systems (from textbook)
- Explore Web Services Landscape (WS-*)
- Investigate REST architectural style (often seen as a competitor to WS-*)
- Examine “Web 2.0” (and its relationship to the above)

Desirable Characteristics of Seminar-Style Class

- Everyone has read the Assigned Readings
- Lots of Discussion
- Student Presentations
- Technology Demonstrations
- Fun!

Course Evaluation

- (At least three) Student Presentations
 - Roughly 15 minutes each
- Group Project (team size: 2—4)
 - Build a Web Service of Non-Trivial Complexity
 - Write a Report about your Service
 - Give Demo and Answer Questions (during last two weeks of class)

Structure of Course (Proposed; Will Change!)

- First Five Weeks: SOAP-Based Web Services / SOA
- Second Five Weeks: RESTful Web Services
- Last Five Weeks: Web 2.0

- Topics will include
 - Historical context of distributed middleware systems
 - Core web technologies
 - XML, SOAP, WSDL, UDDI
 - REST and ROA (Research-Oriented Architecture)
 - and more!

Example Topics for Student Presentations

- SOAP
- WSDL
- UDDI
- XMLHTTPObject
- AJAX
- Google (Yahoo, Amazon, etc.) API
- RSS/Atom/AtomPub
- The latest and greatest social networking site, Web 2.0 frameworks, etc.
- ...

Other ways to participate

- Send in pointers to relevant materials/articles on the Web
 - Then volunteer to discuss them in class!
- Locate speakers from industry to come speak about a related topic
 - Find someone from IBM/Microsoft/Sun to come and talk about what they are doing in the WS-* space
 - Find someone from a Web 2.0 startup to cover their business model, the technologies they use, etc.
- Present counter arguments to the material discussed in lecture
- Summarize seminar-relevant discussions that are occurring in the blogosphere

Questions?



Chapter 1: The Basics of Web Services

- Definitions
- Software as a Service
- Where can services be used?
- Web Service Characteristics
- Web Service Interfaces and Implementations
- Service-Oriented Architecture
- Technology Stack
- Quality of Service
- Impact/Shortcomings of Web Services

Service Oriented Paradigm (I)

- Web Services exist within a paradigm that centers around services
 - We'd like to use services to support the rapid development of low-cost, easily composable distributed applications
- A Web service is a piece of programmatically-available application logic exposed over the Internet
- Services reflect a “service-oriented” approach to development based on the idea of creating applications by composing them from smaller services
 - Its the “component-based programming” paradigm moved to the Net

Service Oriented Paradigm (II)

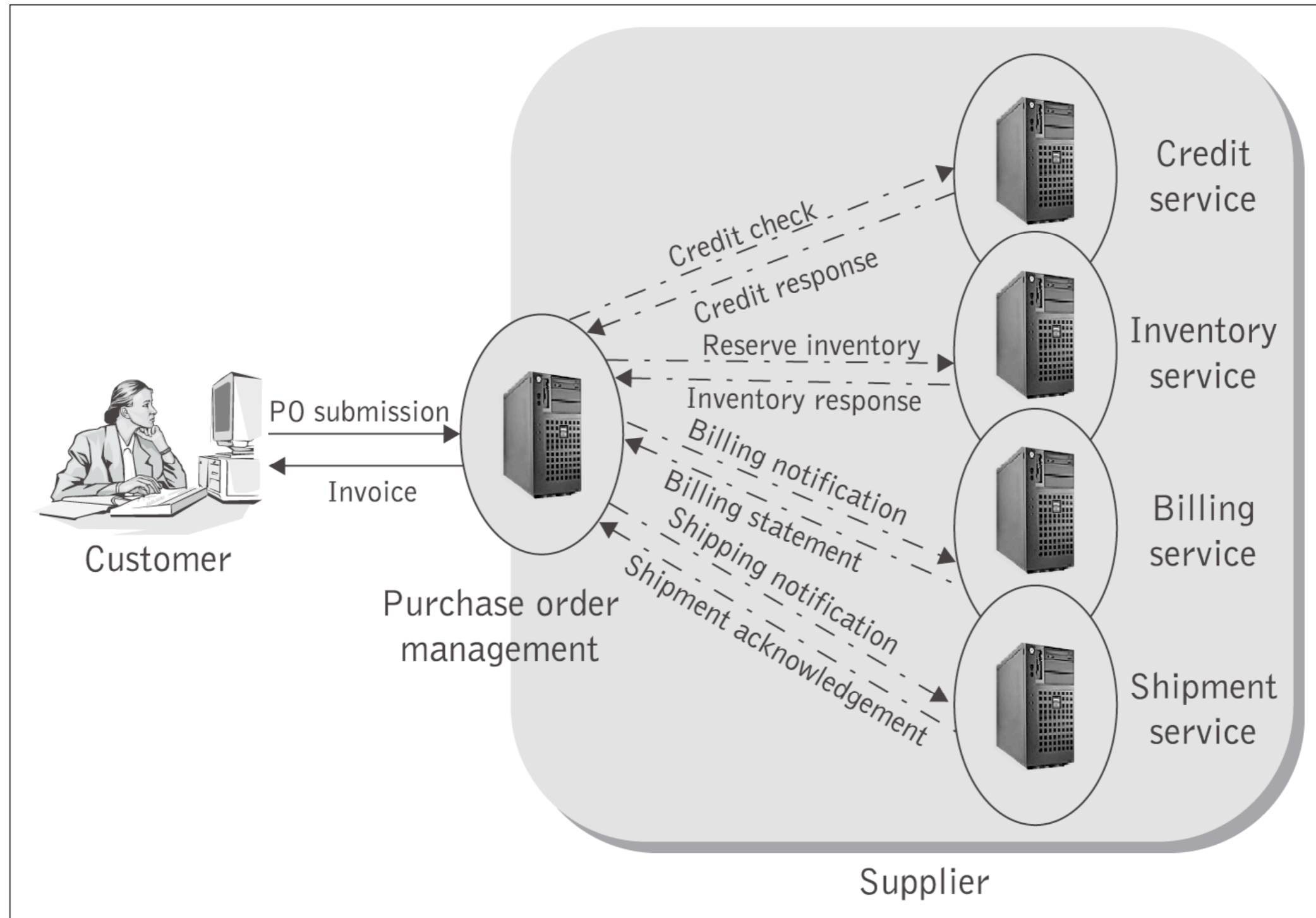
- As a result, services are viewed as independent entities that
 - can be mixed/matched to create a complete business process/application
 - are available to a variety of clients
 - (platform independent from client perspective)
 - able to support a business by charging clients for use of the service
 - pay per use
 - lease a certain number of transactions
 - subscription model
 - etc.

Example Web Services

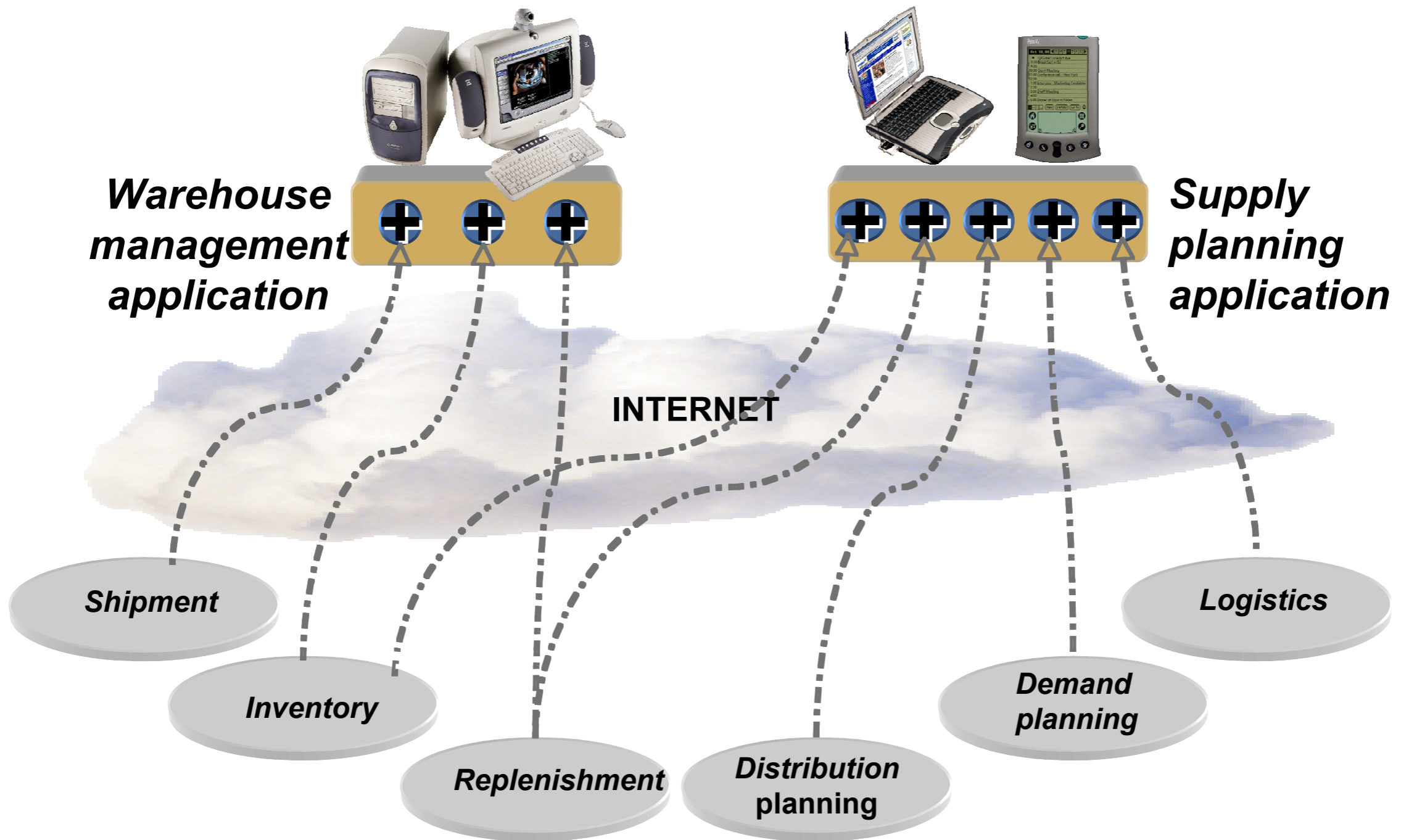
- **a self-contained business task:** withdrawing or depositing funds in an account
- **a full-fledged business process:** automating the purchase of office supplies
- **an application:** forecasting application, stock replenishment, claims, etc.
- **a service-enabled resource:** providing access to a particular back-end database within an organization

- “Big Web Services” are strongly oriented towards a business perspective
 - Also prevalent in certain scientific disciplines, such as Grid computing

Purchase Order Application via Web Services



Supply Chain Application via Web Services



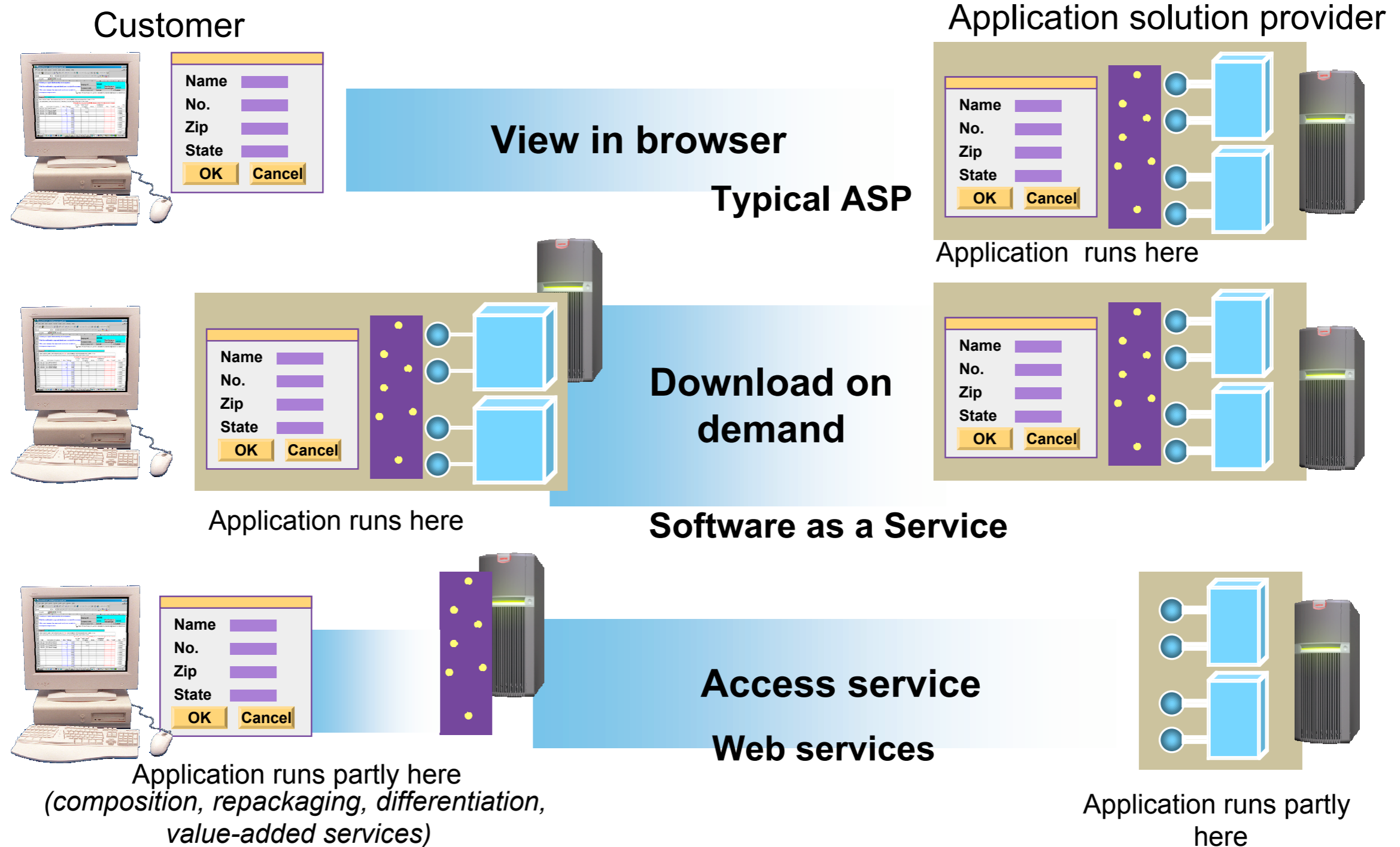
Software as a Service

- Software as a Service is an evolution on top of ideas from the past
 - Most notably component-based programming
- But also appeared in the late 90s early 00s via a model known as the **application service provider**
 - An ASP rents applications to subscribers
 - ASP creates a configurable application that can be customized in various ways for multiple companies
 - Clients pay ASP to manage the infrastructure required to run the application and to store their data; clients access app via browser
 - Some ASP applications are downloadable (fat client model) but still access services/data managed by the ASP on a remote server

ASP vs. Web Services

- Problems with ASP approach
 - inability to develop highly interactive applications
 - Applications were mainly Web-based forms with workflow added on
 - inability to provide complete customization
 - inability to integrate multiple applications together
- Web Services takes a different approach based on
 - loosely coupled, asynchronous interactions
 - sharing data stored in XML files
 - in which the interactions and the data formats have been standardized

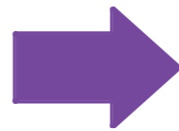
Comparison



Adapted from: CBDi forum copyright
Everware-CBDi

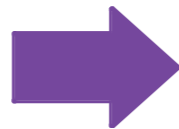
Problems addressed by Web Services

**Broader
B2B**



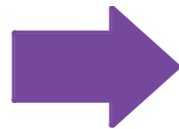
A mid-sized manufacturer needs to create online relationships with customers, each with their own set of standard and protocols.

**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?

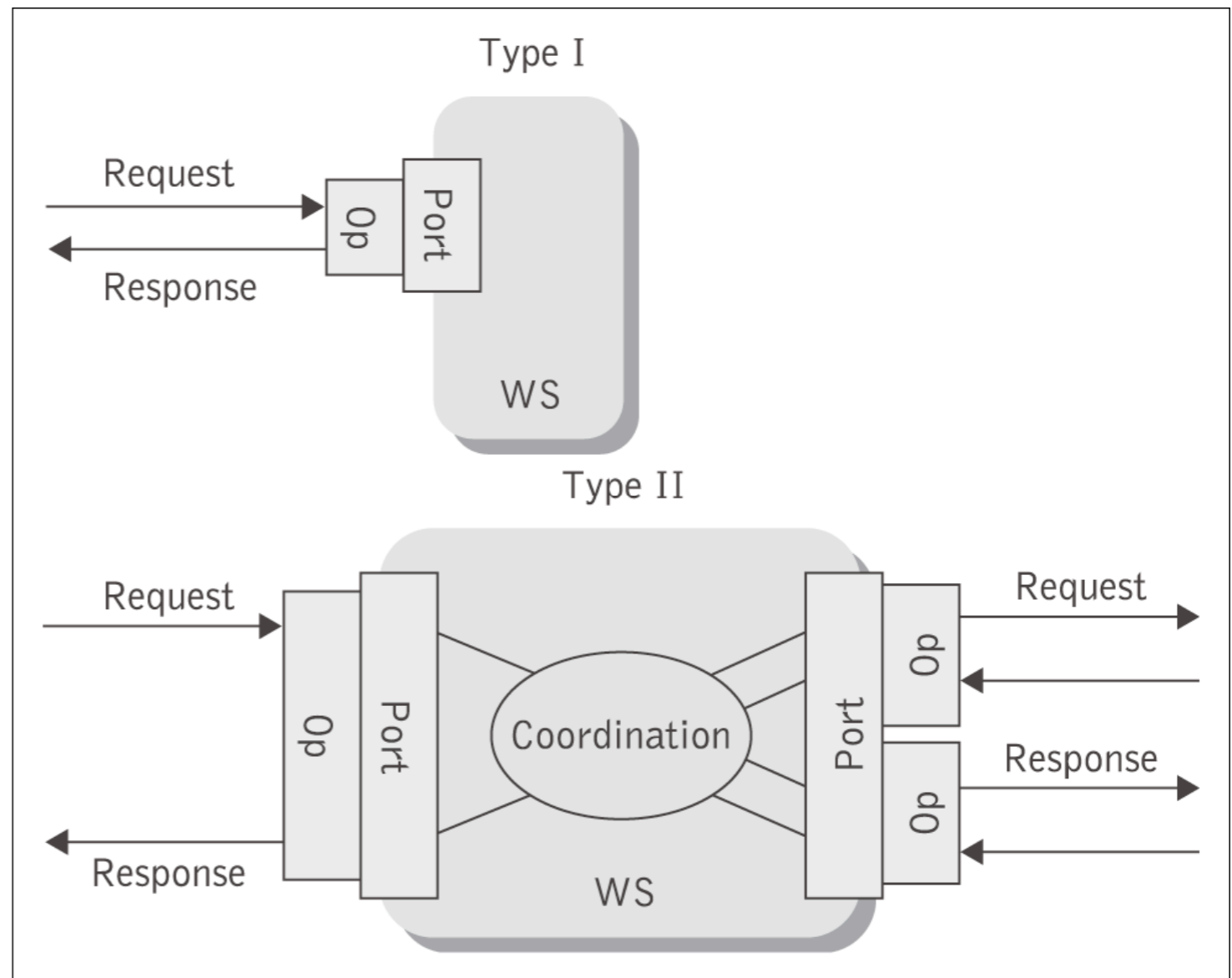
- Within an organization (enterprise application integration)
 - Accelerate integration efforts and reduce costs
 - Save on deployment/maintenance costs
 - Reduce skill requirements
 - Improve re-use
- Between organizations (e-Business integration)
 - Providing services to a company's customers
 - Accessing services from a company's partners and suppliers

Web Service Characteristics

- Types
- Properties and State
- Loose Coupling and Granularity
- Message Style
- Well definedness

Types of Web Services

- Simple Services
 - Provide information via request/response interaction style
 - Example: stock quotes
- Complex Services
 - Unified service consisting of multiple sub-services
 - Example: supply-chain application: ordering, inventory control, etc.



Service Properties and State

- Functional and Non-Functional (Like all software)
 - Functional: what does it do?
 - Non-Functional: how well does it do it?
 - security, robustness, scalability, availability, etc.
- Does the service track state?
 - Stateless: service does not keep track of state
 - Think: Web Server, HTTP is a stateless protocol
 - Stateful: service keeps track of context between calls of a client
 - much more complete interaction style, useful for business processes

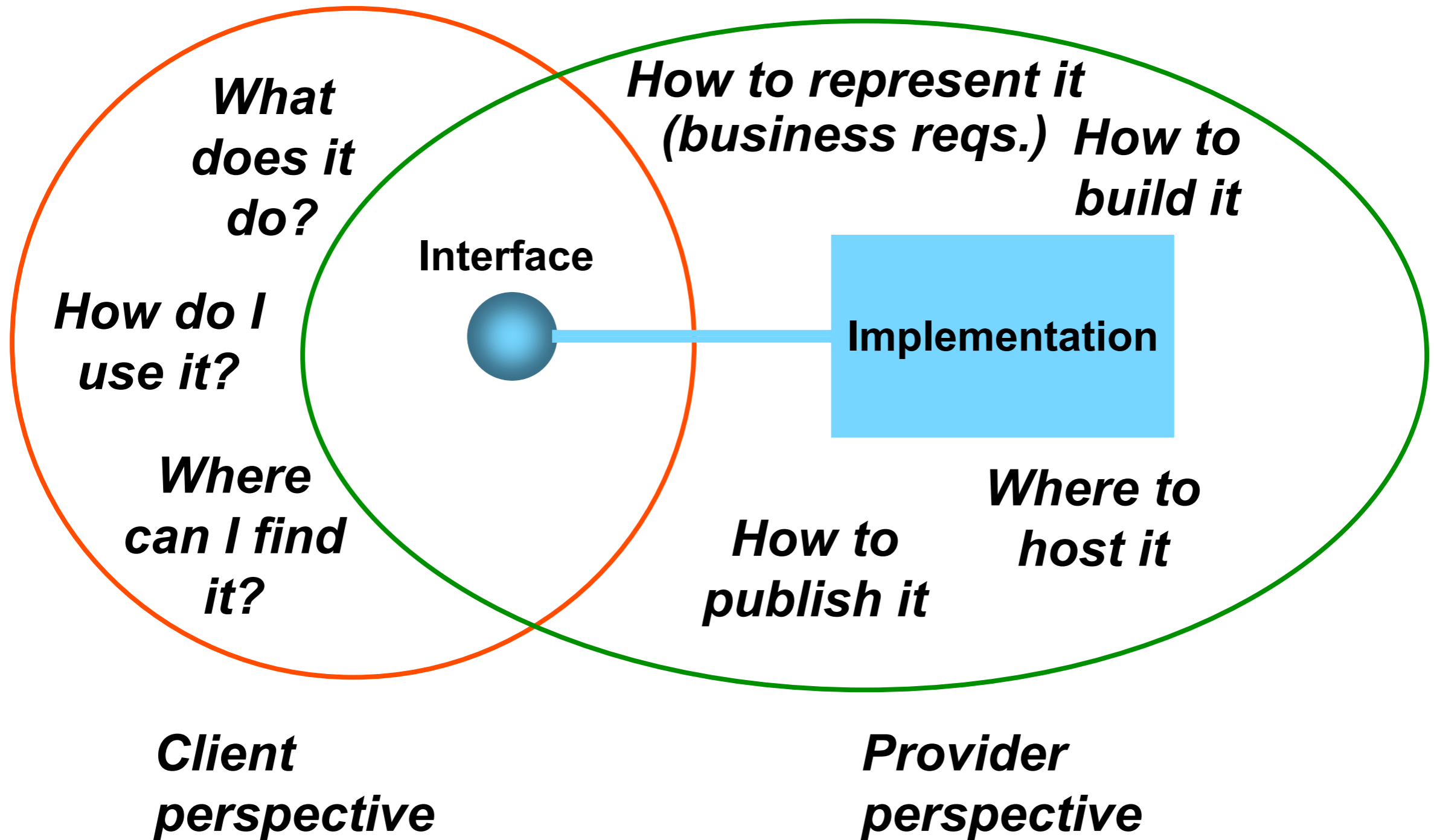
Loose Coupling and Granularity

- Services are considered to be loosely coupled
 - Services typically do not depend on the implementation of client/peer services
 - Interactions are specified via interfaces
 - many different services might implement a single interface
 - can then be swapped in/out based on quality of service requirements
- Service granularity can vary according to complexity
 - Often small XML documents containing reply to single query
 - Complex services might process more course-grained document that, e.g., represent an entire purchase order

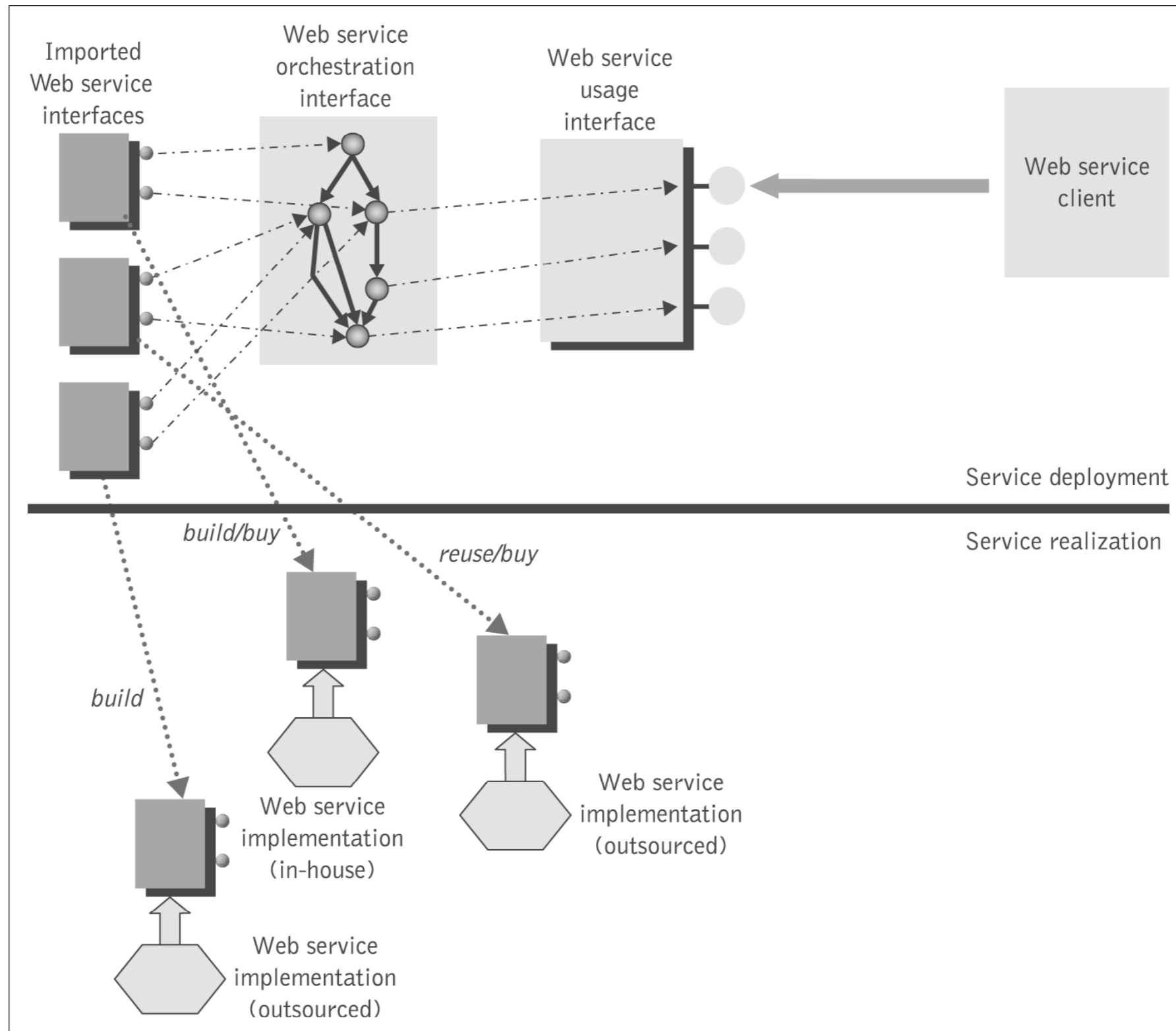
Message Style and Well definedness

- Messages can be passed between services either
 - synchronously: e.g. remote procedure call; client blocks waiting for answer
 - asynchronously: aka “document style”
 - entire document is sent and processed by service
 - at a later point, service gets back in touch with status updates and/or results
- Well definedness
 - Service interactions must be well-defined. The Web services description language is one piece of the puzzle in ensuring that services have well-defined interfaces and interaction styles

Interface vs. Implementation



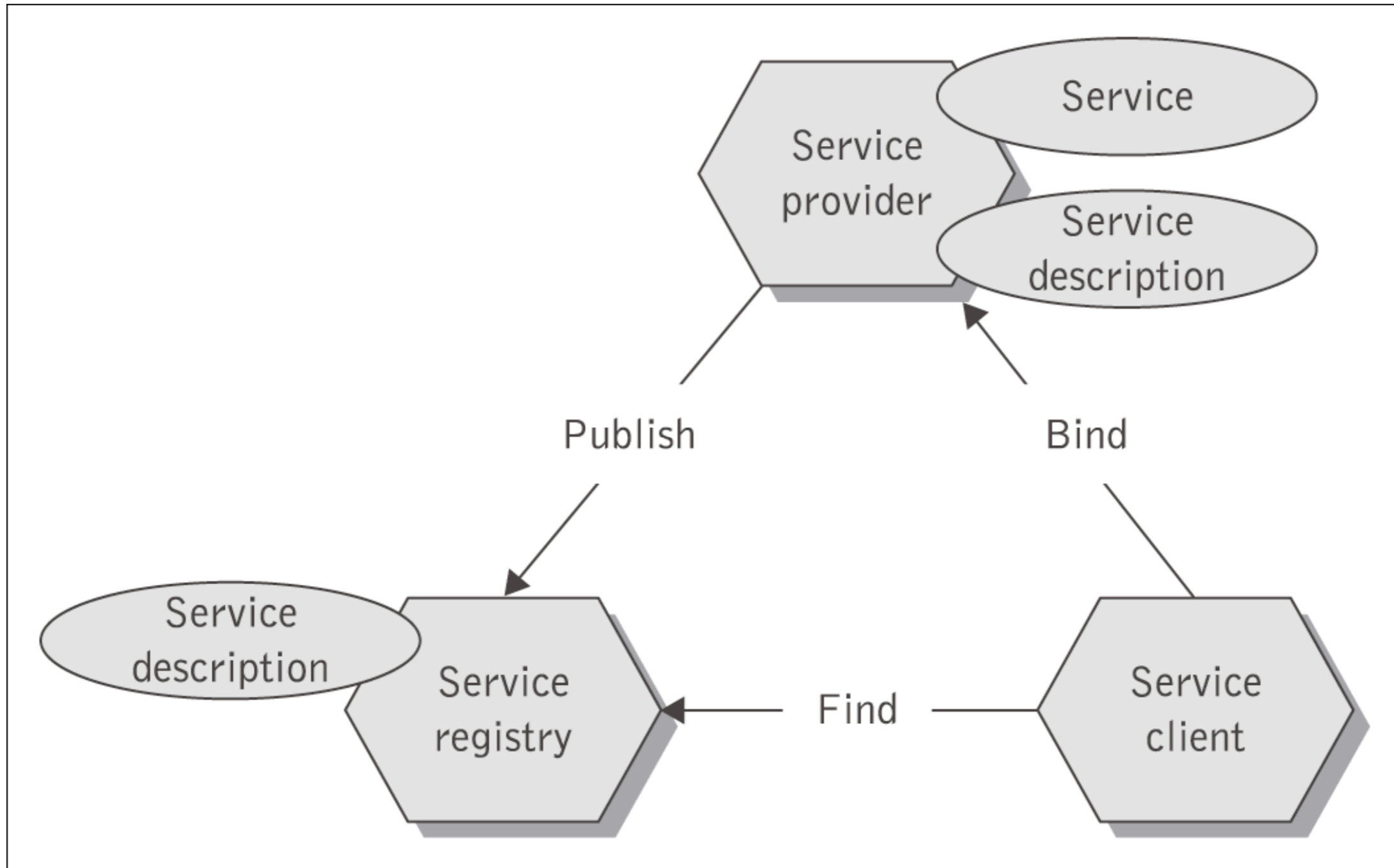
Deployment vs. Realization (Implementation)



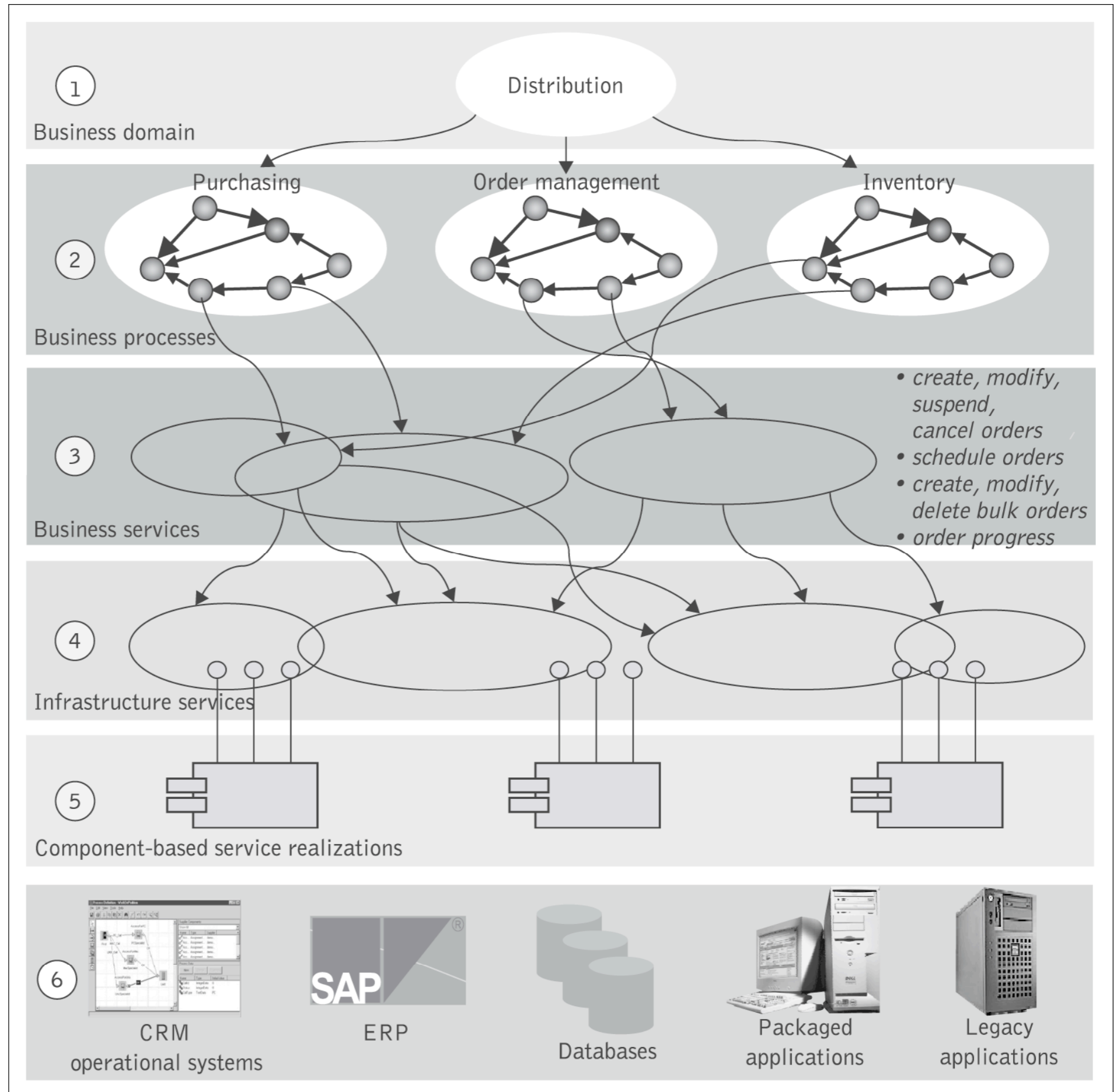
SOA Roles

- Service Providers
 - Create web services and web service descriptions
- Service Clients
 - Users of a Web Service
- Service Registry
 - a service to connect the former
 - providers register services with registry
 - clients search registry for services they need
 - Note: I don't think this model has really taken off;
 - most orgs. play all three roles!

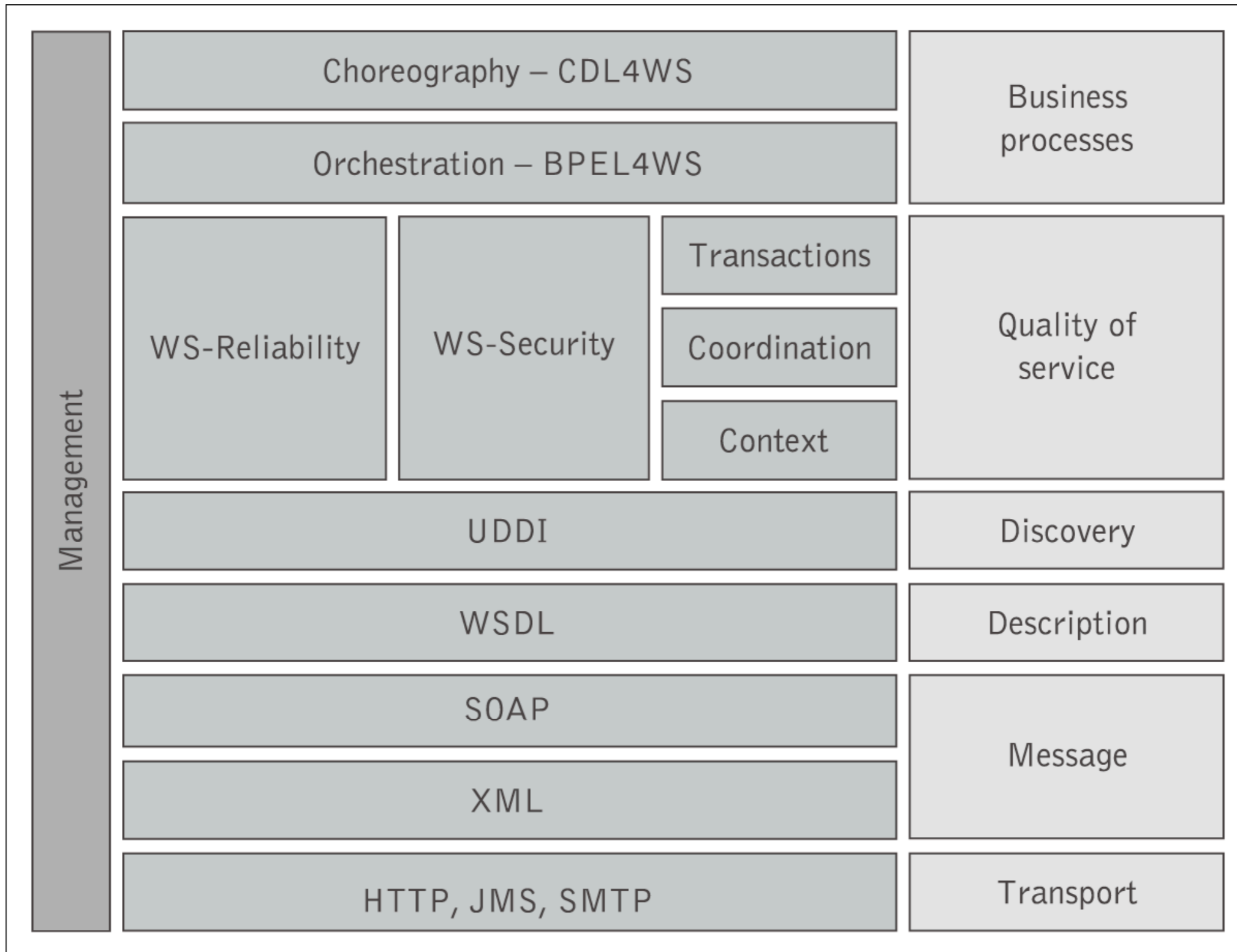
SOA: Roles in Action



Layers of SOA



Web Services Technology Stack



Quality of Service (QoS)

- QoS refers to a Web Service's ability to respond to customer invocations and provide a specified level of quality
 - Key characteristics include
 - availability
 - accessibility,
 - performance
 - security
 - scalability
 - ability to support transactions
- These contracts can be written up in SLAs (service level agreements)

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 services;
 - a standard, easy, and flexible way to help overcome application integration issues;
 - a standard way to develop and/or assemble Internet-native applications for both the internal and the extended enterprise;
 - 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.

Chapter 2: Distributed Computing Infrastructure

- Will switch to a separate set of slides for this part of the presentation

Need volunteers to present next week!

- What is XML
 - Examples of XML Tools
- What is SOAP
 - Example SOAP transactions
 - SOAP-based tools
- Take a look at chapters 3 and 4 and send me mail by Friday if you'd like to present next week
 - I will put volunteers in touch with each other