Web Services

- An Introduction -
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Distributed Information Systems</td>
</tr>
<tr>
<td>2</td>
<td>Middleware</td>
</tr>
<tr>
<td>3</td>
<td>Web Technologies</td>
</tr>
<tr>
<td>4</td>
<td><strong>Web Services</strong></td>
</tr>
<tr>
<td>5</td>
<td>Basic Web Service Technologies</td>
</tr>
<tr>
<td>6</td>
<td>Web 2.0 Services</td>
</tr>
<tr>
<td>7</td>
<td>Web Service Security</td>
</tr>
</tbody>
</table>
Outline

• Motivation
• Technical solution
  – Web Services and their Approach to Distributed Computing
  – Web Service Technologies
  – Web Services Architectures
• Possible extensions
• Summary
• Resources
Motivation
Motivation

• Previously we talked about:
  – Architectures of information systems (Lecture 1)
  – Middleware (Lecture 2)
  – Basic Web technologies (Lecture 3)

• The topics have shown chronological evolution of application integration and distributed application technologies
  – These technologies have been successful in addressing some integration problems but restricted to certain settings (homogeneous middleware, local networks, etc).

• True application integration requires more than what conventional middleware can provide.

• Web services and associated technologies are a solution to such problem
  – The use of standard technologies reduces heterogeneity, thus facilitating application integration.
Technical Solution
Web Services and their Approach to Distributed Computing
Defining Web Services

• What is a Web service?
  – Application accessible to other applications over the Web. (*Sun Microsystems*)
  
  – Self-contained, modular business application which has open, Internet-oriented, standards-based interface. (*UDDI consortium*)
  
  – A software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols. (*W3C*)
  
  – A standardized way of integrating Web-based applications using XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available. (*Webopedia*)

• Regardless of standards and technologies used to enable Web services, the underlying problems are the same.
• Today’s business processes across the companies are carried out manually.
  - The need for automation is driven by various goals: lower costs, streamlined and more efficient processes, monitoring and profiling of process executions, managing exceptions.
Limitations of Conventional Middleware in B2B Integration

There are several reasons why conventional middleware is not suitable for B2B integration:

1. No obvious place where to put middleware in cross-organizational interactions.
   - All parties must agree on using and cooperatively managing a certain middleware platform and implement a global workflow that drives the whole business process.
   - This approach is not really feasible: lack of trust, autonomy of company, confidentiality issues.
   - Companies may try to address the problem in point-to-point fashion: every partner is responsible for its own role in integration with another partner.
   - This leads to wide integration problems for each of the partners.

2. Difference in EAI and conventional middleware assumptions
   - Short lived EAI interactions vs. long-lived cross-organizational interactions.
   - Cross-organizational interactions are mostly implemented as asynchronous exchanges.

3. Inside (EAI interactions) vs. across trust domain (cross-organizational interactions) interactions
   - Required authentication, authorization, encryption, restrictions of functionality.
B2B Integration before Web Services

• Broker companies which facilitate integration by performing functions of centralized EAI middleware
  – Binding and routing messages among services provided by different companies.
  – Lack of support for formats and protocols offered by brokers, trust-related issues.

• EDIFACT-based B2B integration
  – Systems supporting EDIFACT lack of standards and appropriate infrastructure (e.g., middleware).
  – Underlying hardware and communication support was heavy-handed (e.g., leased lines).
  – Systems were expensive to develop, difficult to maintain, not adaptive to new technologies.
• The Internet has alleviated some of the problems
  – Lack of standardization at the system and communication protocol level is still present.
  – Standardization attempts: homogeneous middleware platforms to communicate to each other (Inter-ORB via GIOP/IIOP).

• The Web as final step towards facilitating application integration
  – The Web brought standards in interaction protocols (HTTP) and data formats (XML) which became the base for a common middleware.
  – HTTP and XML are not enough – lack of IDLs, name and directory services, transaction support, and other abstractions needed for application integration.

• Web services are filling the gap between Web technologies (HTTP, XML) and what application integration requires.
  – They resolve the limitations of conventional middleware through three main aspects: SOA, redesign of middleware protocols, and standardization.
B2B Integration with Web Services
Service-oriented Paradigm

- A company exposes functionality as a service
  - Service invocation is made by a program (inter program communication).

- Web services are used similarly as middleware services
  - They are invokable across the Web and across companies.

- Web services are loosely-coupled.
  - Defined, developed, managed by different companies.
  - Web services are facilitating adoption of SOA.
    - Everything is a service.
    - Services are autonomous and independent.

- They provide foundation for building modular and flexible applications.
• Redesign of the middleware protocols to work in a P2P fashion and across companies
  – 2PC is designed for intra organizations interactions
    • Central transaction coordinator, locking of resources.

• 2PC must be redesigned to work in a fully distributed fashion
  – Transaction coordinator must be distributed.
  – More flexibility is needed in terms of resource locking, and trust handling.
• In conventional middleware the presence of standards helped to address many issues:
  – CORBA and Java enabled development of portable applications, low-cost middleware tools, reduced learning curves due to adoption of common models and abstractions.

• For Web services standardization is not only beneficial but necessary
  – SOA languages and protocols must be standardized and widely adopted.

• Alongside OASIS and W3C, major software vendors are committed to standardization.

• Web services are just following the same successful path taken by the Web regarding Web technologies in terms of standardization.
B2B Integration with Web Services

- **Customer**
  - Web service
  - Internal procurement requests
    - Internal infrastructure
  - Interactions based on protocols redesigned for peer-to-peer and B2B settings
  - Internal functionality made available as a service

- **Supplier**
  - Web service
  - Internal infrastructure
    - Languages and protocols standardized, eliminating need for many different middleware infrastructures (need only the Web services middleware)

- **Warehouse**
  - Web service
  - Internal infrastructure

Copyright Springer Verlag Berlin Heidelberg 2004
• Web Services are entry points to the local information systems
  – Exposing functionality performed by internal systems and making it discoverable and accessible through the Web in a controlled manner.
  – They constitute the base on which middleware supporting application integration on the Web is constructed by avoiding the standardization issues.
Technical Solution
Web Service Technologies
Service Description

- What exactly is a service and how can it be described?

- In conventional middleware it is based on interfaces and IDLs
  - All the information is known in advance by the programmer developing the clients.
  - Some information on service description and binding process is implicitly assumed.

- In Web services such an implicit context is not available
  - Thus service descriptions must be richer and more detailed, going beyond the mere service interface.
Service Description

• Common base language
  – Common meta-language used as a basis to specify all the languages necessary to describe the different aspects of a service.
  – eXtensible Markup Language (XML) – standardized and flexible.

• Interfaces
  – Interface definitions resemble CORBA (IDL).
  – The difference is in availability of different interaction modes and XML Schema-driven data types.
  – Context-related information must also be present: service endpoint (URI), transport protocol (e.g., HTTP)
  – The dominant proposal for Web services IDL is WSDL.

• Business protocols
  – A number of operations that a client invokes in a certain order to achieve the goal is a conversation.
  – Rules that govern conversation are specified as part of business protocol supported by the service.
  – Languages for defining business protocols:
    • Web Services Conversation Language (WSCL)
    • Business Process Execution Language for Web Services (BPEL4WS)
• **Properties and semantics**
  – Additional layers of information are needed to facilitate binding in autonomous and loosely-coupled settings.
  – Non-functional properties (cost, reputation, quality) and textual descriptions may be crucial in deciding about the service usage but are not part of a service interface.
  – Universal Description, Discovery and Integration (UDDI) for building repositories and organizing information about Web services.

• **Verticals**
  – Aforementioned layers are generic – they standardize neither contents nor semantics of a service.
  – Vertical standards are using the layers to specify semantics and contents of Web service-based interactions in a specific application domain.
  – RosettaNet describes commercial exchanges in the IT world.
Service Discovery

• Service descriptions are stored in a service directory
  – Service designers may register new services.
  – Service users may search and locate services.

• Discovery can be done at
  – Design-time – browsing and identifying the most relevant services,
  – Run-time – using dynamic binding techniques.

• Directory can be managed and hosted by a trusted third party or as internal company solution.

• Clients are interacting with the directory service by relying on APIs and protocols.

• UDDI specification defines standard APIs for publishing and discovering info in service directories.
Service Interactions

- A set of abstractions and tools enabling interaction among Web services
  - A set of standards that address different aspects of the interactions at different levels

![Service interaction stack diagram](image)

- The protocols are useful to any Web service and are therefore implemented by the Web services middleware.
  - They are transparent to the developers.
Service Interactions

- **Transport**
  - Communication network is hidden behind a transport protocol
  - Wide range of protocols can be used, such as HTTP.

- **Messaging**
  - A standard way to format and package the information to be exchanged
  - Simple Object Access Protocol (SOAP) specifies generic message templates to add on top of the application data.

- **Protocol infrastructure (meta-protocols)**
  - Much of the software required to support such protocols can be implemented as generic infrastructure components.
    - Maintenance of state of the conversation, association of messages to the appropriate conversation, verification of message exchanges.
  - Meta-protocols facilitate and coordinate the execution of business protocols
    - Agreement between clients and services on which protocol should be executed, who is coordinating, how the protocol execution identifiers are embedded in messages.
  - WS-Coordination standardizes these meta protocols.
• Middleware (horizontal) protocols
  – Since Web services and their supporting infrastructure are distributed in nature, middleware properties that go beyond the basic communication are achieved by means of standardized P2P protocols.
  – They are horizontal since they are applicable to many Web services.
  – Reliability and transactions require the execution of protocols (e.g., 2PC) among the interacting entities.
  – Horizontal protocols can be supported by meta-protocols.
  – They are hidden from the Web service developers and managed by infrastructure.
  – WS-Transaction – builds upon WS-Coordination to define how to implement transactional properties when dealing with Web services.
Technical Solution
Web Service Architectures
The Two Faces of Web Service Architectures

- Web services as a way to expose internal operations of a company
  - System receives requests through the Web and passes them to the underlying IT system.
  - The problems are analogous to those encountered in conventional middleware.
  - This is internal middleware for Web services (term internal architecture is used to refer to organization and structure of the internal middleware).

- Web services as a way to integrate systems across the Internet.
  - Middleware infrastructure is needed to integrate different Web services.
  - This is external middleware for Web services (term external architecture is used to refer to organization and structure of the external middleware).
  - External architecture has three components
    - Centralized brokers – message routing and providing support for interactions (logging, transactional guaranties, name and directory services, etc).
    - Protocol infrastructure – coordinating interactions between Web services in distributed settings.
    - Service composition infrastructure – definition and execution of composite services.
The Two Faces of Web Service Architectures
Internal Architecture of a Web Service

- Web services as yet another tier on top of the other tiers of the enterprise architecture.
  - The result is a multi-tier system in which services are implemented on top of other services and basic programs.
• Multiple middleware instances don’t need to be the same.
  – Compatible service abstractions or compatibility wrappers are needed.
  – Middleware is reconciling interaction heterogeneities.

• Web services are playing the same role
  – Implementing a Web service essentially requires an extra tier on top of the others to enable access using standard Web service protocols.

Much of the internal middleware for Web Services targets packing and unpacking of messages exchanged between Web services and converting them into the format supported by the underlying middleware.
• Case for the external middleware is not clear
  – Who owns the middleware?
  – Where to locate it?
  – How to trust to the provided middleware services?

• Two solutions to solve the problem:
  1. Implement middleware as P2P system
     • All participants cooperate to provide the services
     • Reliability and trustworthiness is questionable (e.g., name and directory services)
  2. Introduce intermediaries or brokers acting as necessary middleware
     • Part of the middleware can reside at different locations

• Currently only name and directory services (UDDI) is standardized and “used” in practice.
External Architecture of a Web Service

Company A (service requester)

Web service client

Web services middleware (internal)

other tiers

Company B (service provider)

Web service

Web services middleware (internal)

other tiers

1. publish the service description

2. find

3. interact

the abstraction and infrastructure provided by the registry are part of the external middleware

service descriptions

Company C (directory service provider)

Copyright Springer Verlag Berlin Heidelberg 2004
• But where are the other middleware features (transaction management supported by TP monitors, services provided by CORBA)?

• Centralized transaction broker is theoretically possible and technically feasible but runs into various problems
  – Standard way of running transactions accepted by everyone so that transactional semantics is not violated.
  – All participants trust the broker (highly improbable).

• Alternative is to implement the transaction broker as a P2P system
  – Each service requester has its own transaction manager.
  – Functionality provided by this solution is a subset of the functionality offered by conventional middleware systems.
External Architecture of a Web Service
Possible Extensions
Possible Extensions

• RESTful-based services are dominating the Service Web
  – 68% RESTful\(^1\) services vs. 19% SOAP services\(^2\).
  – It is expected that the dominance of RESTful services will grow up in future.

• RESTful-approach represents a natural way to offer Web Services as opposed to the SOAP-based Web Services
  – It builds on top of the architectural style which pervades the Web
  – It relies on the proven Web protocol (HTTP) and data formats (XML, JSON).
  – It integrates easily with the dominant visualization tool (a.k.a. Web browser) through JavaScript and AJAX.

• Ongoing efforts are trying to address the presented issues (like WADL service description for RESTful services)

\(^1\) Statistics retrieved from the Service Finder demo on Dec 17\(^{th}\), 2009 @ http://demo.service-finder.eu/statistics
\(^2\) Statistics retrieved from the Programmable Web on Dec 17\(^{th}\), 2009 @ http://www.programmableweb.com/apis
Possible Extensions

• Semantic Web Services
  – Automating the Web service lifecycle by relying on semantic technologies
  – Most prominent approaches are:
    • WSMO, OWL-S, SWSF, METEOR-S (for traditional Web services)
    • Micro-WSMO, SA-REST (for RESTful Web services)
Summary
Summary

- Web services reduce heterogeneity and facilitate application integration.

- Web services are filling the gap between Web technologies (HTTP, XML) and application integration requirements.

- Web services are built on top of the standardization efforts to define common based languages (XML), interfaces (WSDL), business protocols (WSCL, BPEL4WS), properties and semantics (UDDI), and vertical standards (e.g. RosettaNet).

- Web service interactions are supported through transportation protocols (HTTP), messaging (SOAP), meta-protocols (WS-Coordination), and middleware-protocols (WS-Transaction)

- Web service architectures can be characterized as internal and external.
References
• Mandatory reading

• Wiki and Web reference
  – W3C http://en.wikipedia.org/wiki/W3C
  – WSDL http://en.wikipedia.org/wiki/Web_Services_Description_Language
References

• Wiki and Web reference (cont’d)
  – P2P http://en.wikipedia.org/wiki/Peer-to-peer
  – 2PC http://en.wikipedia.org/wiki/2PC
Table of Contents:

<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Distributed Information Systems</td>
</tr>
<tr>
<td>2</td>
<td>Middleware</td>
</tr>
<tr>
<td>3</td>
<td>Web Technologies</td>
</tr>
<tr>
<td>4</td>
<td>Web Services</td>
</tr>
<tr>
<td>5</td>
<td><strong>Basic Web Service Technologies</strong></td>
</tr>
<tr>
<td>6</td>
<td>Web 2.0 Services</td>
</tr>
<tr>
<td>7</td>
<td>Web Service Security</td>
</tr>
</tbody>
</table>
Questions?