

Chapter 8 – Web Services Foundations



Outline

- Service-oriented computing
 - Motivation & Architecture
 - Implementing SOAs
- Web Services
 - System Architecture
 - Overview of Technologies and Standards
- SOAP for Web Service Invocation
 - Message Format
 - Processing Model
- WSDL for Web Service Description
 - Main Ingredients & Document Structure
 - Web Service Policies
- UDDI for Web Service Discovery
- Application Development

Service-Oriented Computing (SOC)

- Service-oriented architecture (SOA)
 - models an application as a composition of reusable services
 - focus is on functions, not things (in contrast to OO-design)
 - services are characterized by
 - the messages they exchange
 - the interface contracts defined between service requester and provider
- TP system based on SOA may include
 - multiple reusable services offered by a single TA-program, or multiple distributed services
 - both synchronous and asynchronous communication mechanisms
 - service is invoked by sending a message to the service
 - service can implement a TA or a step within a TA (request controller or TA server)
- Increased popularity of SOC
 - service-oriented access to functions of large-scale web sites (search, social networking, e-commerce)
 - advent of standard web service protocols

Types of E-Business

Business To Consumer (B2C)	Business To Business (B2B)	Intra Business
<ul style="list-style-type: none">• Relation between enterprise and customers• Sales-related aspects are predominant, like product presentation, advertising, service advisory, shopping	<ul style="list-style-type: none">• Relation between processes of different enterprises• Predominant are relation to suppliers, and customer relations to other enterprises like industrial consumers, retailers, banks	<ul style="list-style-type: none">• Electronic organization of internal business processes, like realization within workflow systems

B2B Integration – Conventional Middleware

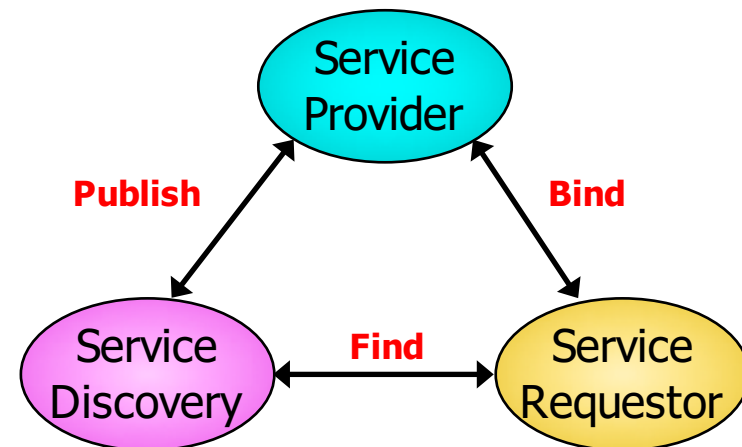
- Middleware itself is (logically) centralized
 - usually controlled by a single company
 - now requires agreement on using, managing specific middleware platform across companies ("third party")
 - need to implement a "global workflow"
 - problems
 - lack of trust
 - autonomy needs to be preserved
 - business transactions are confidential
- Point-to-point solutions
 - lack of standardization
 - many partners involved -> heterogeneity of middleware platforms
- Focus on LAN
 - insufficient support for internet protocols
 - problems with firewalls
 - cannot work with multiple trust domains

Service-Oriented Architecture (SOA)

- **Service Requestor**
 - Finds required services via Service Discovery
 - Binds to services via Service Provider
- **Service Provider**
 - Provides e-business services
 - Publishes availability of these services
- **Service Discovery**
 - Service Registry
 - Provides support for publishing and locating services
 - Like telephone yellow pages
 - Service Index
 - Publication is "passive": service descriptions are made available and gathered by index service
 - Peer-to-peer Discovery
 - Dynamic discovery: requestor send queries to peers in a network

Definition (given by OASIS SOA Reference Model):

"A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains"



Implementing SOAs

- Implementation based on Web Services
 - application function is mapped to a specific service interface (e.g., *AddCustomer*)
 - standards for SOA: invocation (SOAP), interface description (WSDL), registry (UDDI), all based on XML
 - interoperability: interfaces are available for appl. servers, ORBs, MOM, DBMS, ...
 - includes transaction interoperability
 - service assembly & composition: tools and techniques available
- Implementation based on Representational State Transfer (REST)
 - architectural style for building large-scale distributed hypermedia systems
 - application function is mapped to a specific resource providing a generic interface
 - resource identification through URI; uniform interface: HTTP GET, PUT, POST, DELETE
 - example: *HTTP POST* on *www.company-xyz.com/customers* to add customer
 - resources encode session state, function identifiers and parameters
 - *representational state transfer* using self-contained messages
 - self-descriptive messages using HTTP header (*content-type*, *accept* fields)
 - resources are decoupled from their representation
 - content can be accessed in a variety of formats, such as JSON (content negotiation)

RESTful Web Services

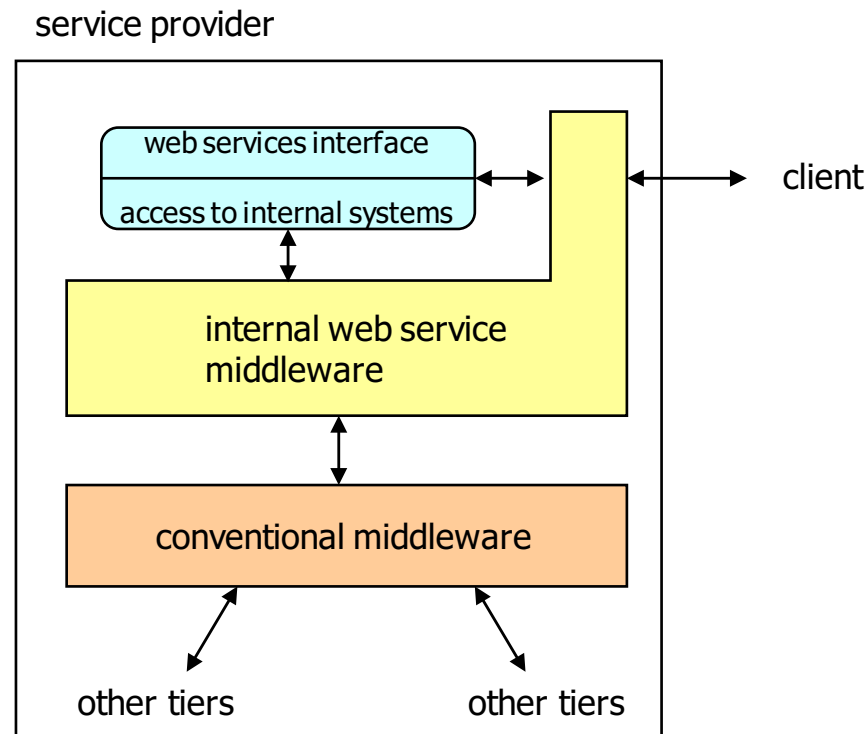
- Web services provided purely based on the above principles
 - alternative to WSDL/SOAP-based “big” web services
- Rationale
 - perceived to be simple
 - leverages existing, well-known standards (HTTP, XML, URI, MIME)
 - light-weight infrastructure that requires only minimal tooling, is inexpensive
 - similar to building dynamic web site
 - REST and Ajax (see later chapter) complement each other nicely
 - suitable for hosting applications on the web
- Drawbacks and limitations
 - (still) needs careful design and enumeration of resources to be exposed, mapped to generic interface, and of data representations used
 - more flexible, but more format variations to account for
 - only RPC-style interactions, HTTP-only
 - requires manual implementation of reliability, transactions

Web Services

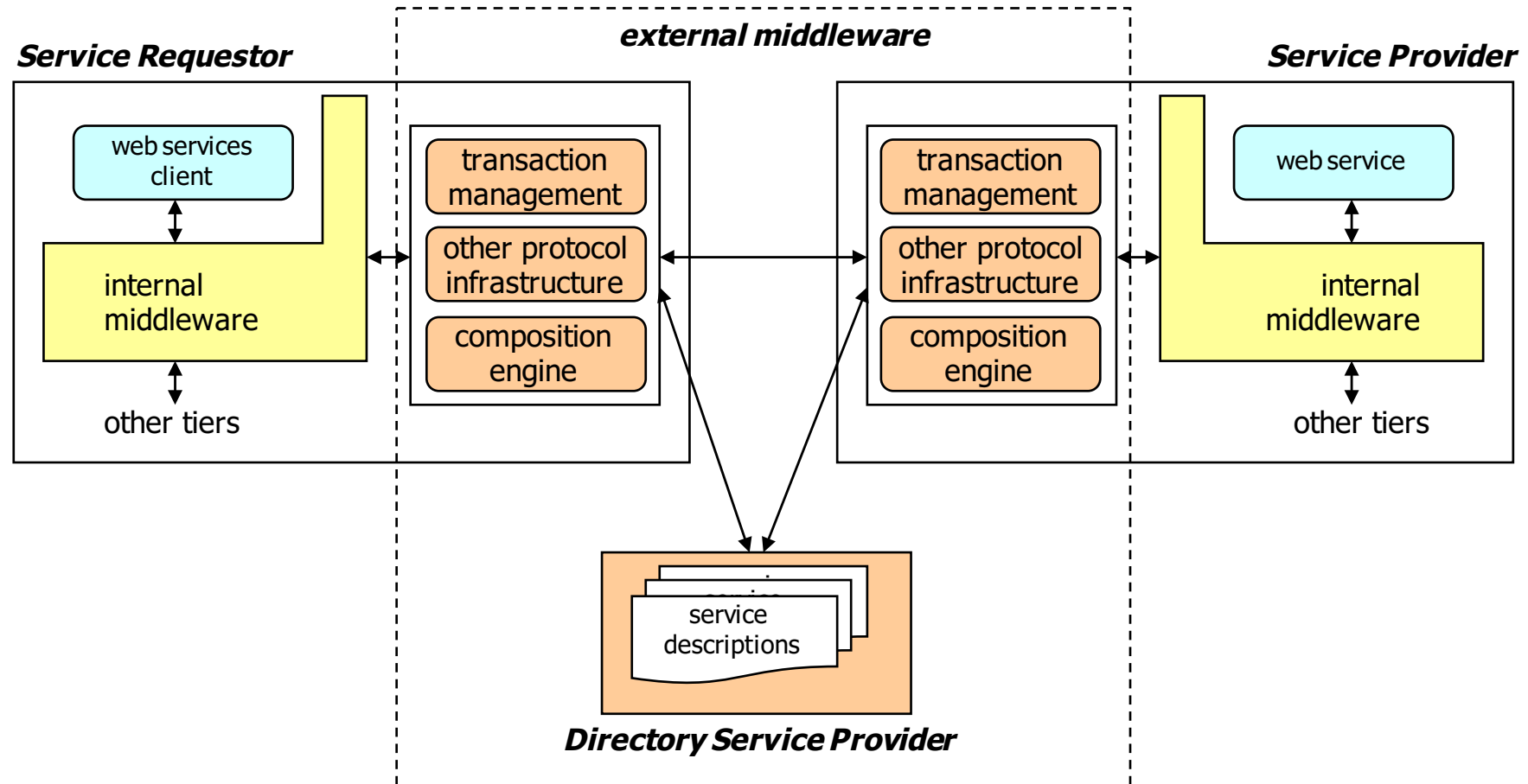
- New distributed computing platform built on existing infrastructure including XML & HTTP
 - Web services are for B2B what browsers are for B2C
 - Self-contained, self describing, modular service that can be published, located and invoked across the web
 - Refer to open standards and specifications:
 - component model (WSDL)
 - inter-component communication (SOAP)
 - discovery (UDDI)
 - Platform- and implementation-independent access
 - Described, searched, and executed based on XML
 - Enable component-oriented applications
 - Loose coupling from client to service
 - Enable to integrate legacy systems into the web
 - Useful for other distributed computing frameworks such as CORBA, DCOM, EJBs
- ➔ Web services as wrappers for existing IS-functionality

Web Service System Architecture

- Common internal architecture leveraging conventional middleware



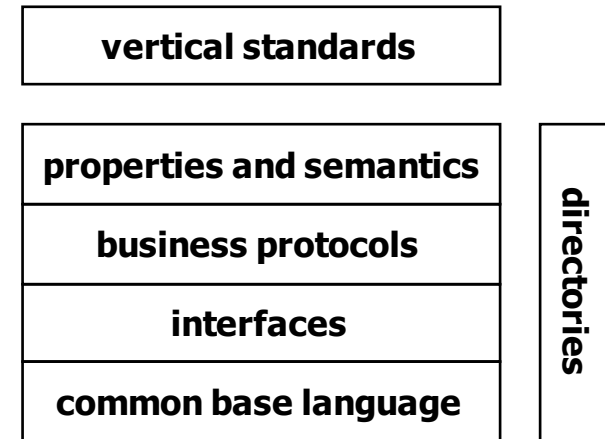
External Web Services Architecture



Technologies: Service Description & Discovery

■ Service **Description**

- Common Base Language (→XML)
- Interfaces (→WSDL)
 - extend "traditional" IDLs
 - interaction mode
 - address/transport protocol info
- Business Protocols (→WSCL, BPEL)
 - describe possible *conversations*
 - order of interactions
- Properties and Semantics (→UDDI, WS-Policy)
 - descriptions to facilitate binding in a loosely-coupled, autonomous setting
 - e.g., non-functional properties (cost, transactional & security support)
 - textual descriptions
 - organize this information
- Vertical Standards
 - interfaces, protocols, etc. specific to application domains



Service Description and Discovery Stack

■ Service **Discovery**

- Directory/Repository for WS descriptions
- APIs and protocols for directory interaction
 - at design-time or run-time

Technologies: Service Interaction & Composition

■ Service **Interaction**

- Transport
 - lots of possibilities
 - HTTP most common
- Basic and Secure Messaging
 - standardize how to format/package information to be exchanged (→ SOAP)
 - define how to extend basic mechanism to achieve additional capabilities (→ WS-Security)
- Protocol Infrastructure (meta-protocols)
 - general infrastructure for business interactions
 - maintain state of conversation
 - meta-protocols
 - which protocols do we use?
 - who is coordinating?
- Middleware Properties (horizontal protocols)
 - properties similar to those of conventional middleware
 - reliability, transactions, ...

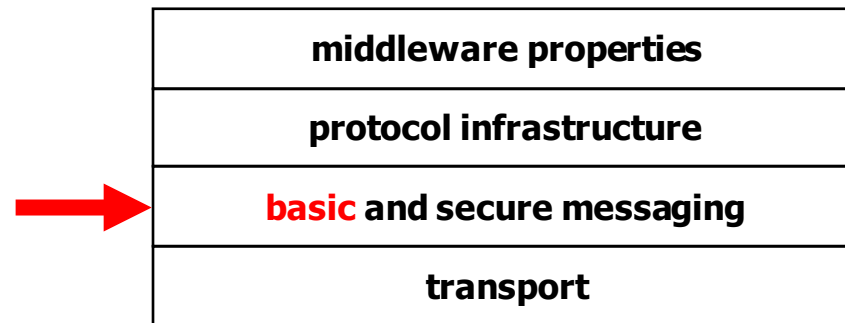
middleware properties
protocol infrastructure
basic and secure messaging
transport

Service Interaction Stack

■ Service **Composition**

- Implement web service by invoking other web services
- Similar to workflow management, only for web services

SOAP – Simple Object Access Protocol



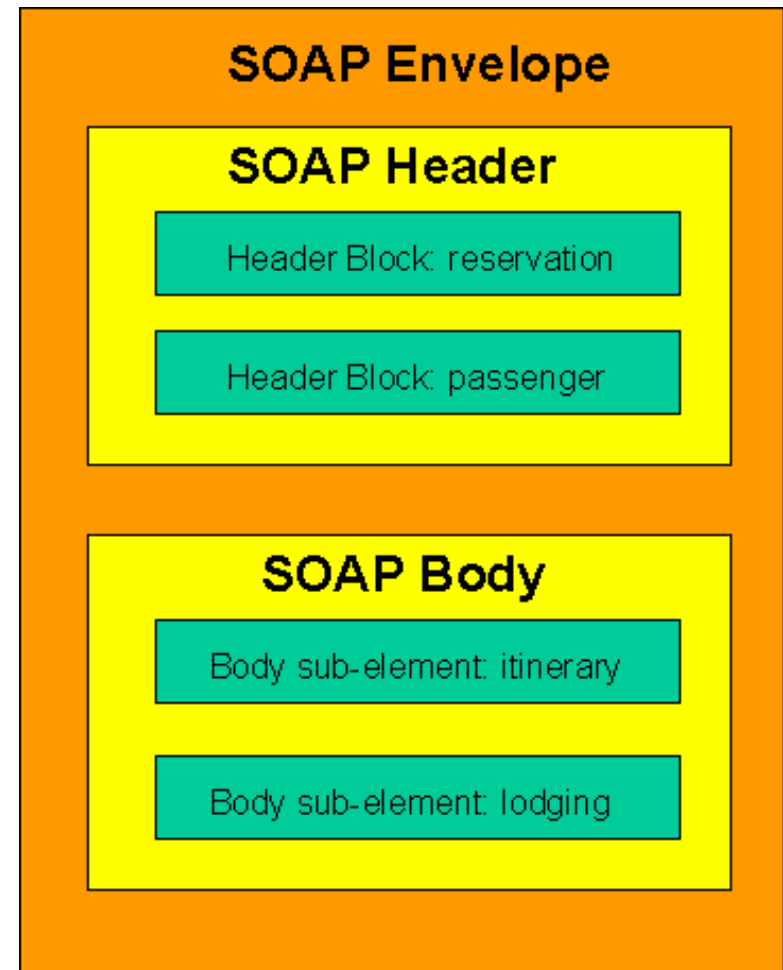
Service Interaction Stack

- Defines how to format information in XML so that it can be exchanged between peers
 - message format for **stateless, one-way communication**
 - support loosely-coupled applications
 - conventions for interaction patterns (RPC)
 - implement "on top of" one-way messaging
 - first message encodes the call, second (reply) message the result
 - processing rules for SOAP messages
 - how to transport SOAP messages on top of HTTP, SMTP

SOAP Envelope Framework

- Defines mechanism for identifying
 - What information is in the message
 - Who should deal with the information
 - Whether this is optional or mandatory
- **Envelope** element is the root element of the SOAP message, contains
 - Optional **header** element
 - Mandatory **body** element
- Body element
 - Contains arbitrary XML
 - application-specific
 - Child elements are called body entries (or bodies)

Example:

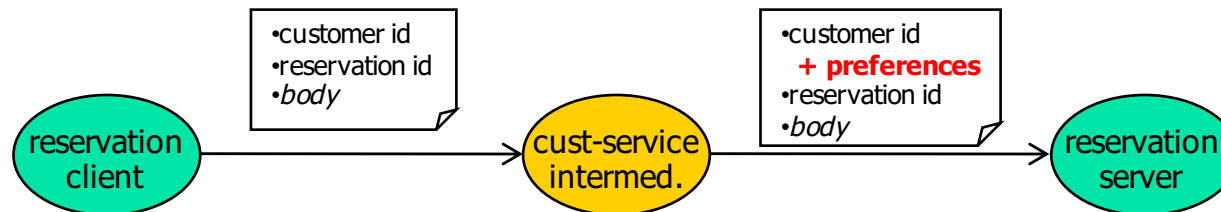


SOAP Headers

- Primary extensibility mechanism in SOAP
 - Additional facets can be added to SOAP-based protocols
 - Mechanism to
 - provide additional "control" information (e.g., directives, context information)
 - pass information that is orthogonal to the specific information to execute the request
 - Any number of headers can appear in a SOAP envelope
- Usage areas
 - Application-specific extensions (see previous example)
 - e.g., reservation identification, customer identification and information, ...
 - Generic service extensions
 - authentication, authorization, transaction management, payment processing, tracing, auditing
- Header content
 - Arbitrary XML
 - Determined by the schema of the header element

SOAP Intermediaries

- SOAP intermediaries provide "value-added services"
 - SOAP message can travel through multiple SOAP nodes
 - Sender [-> Intermediary ...] -> ultimate Receiver
 - Intermediaries process one or more SOAP headers
 - Header is removed from the message after processing (default behavior)
 - can be reinserted by the intermediary, possibly with modified values
 - Intermediary does not need to understand message body



SOAP-based RPCs

- SOAP is fundamentally a stateless, one-way message exchange paradigm
 - ...but applications can create more complex interaction patterns
 - Request/response, request/multiple responses
- SOAP-based RPC
 - Employs request/response message exchange pattern (MEP)
 - MEPs define "templates" for more complex message exchanges
 - Invocation is modeled as a struct of in/inout parameters
 - `<doCheck>`
 - `<product> ... </product>`
 - `<quantity> ... </quantity>`
 - `</doCheck>`
 - Response is modeled as a struct as well
 - `<doCheckResponse> ... </doCheckResponse>`
 - All data is passed by-value
 - Endpoint (address of target node) to be provided in a protocol binding-specific manner

More SOAP

- SOAP protocol bindings

- SOAP is transport-independent, can be bound to any protocol type
 - E.g., SMTP, JMS, UDP, ...
 - SOAP standard defines a binding to HTTP
- Binding to HTTP (synchronous protocol) makes RPC-style “natural”
 - One-way exchange will use simple acknowledgement as HTTP response

- SOAP with Attachments

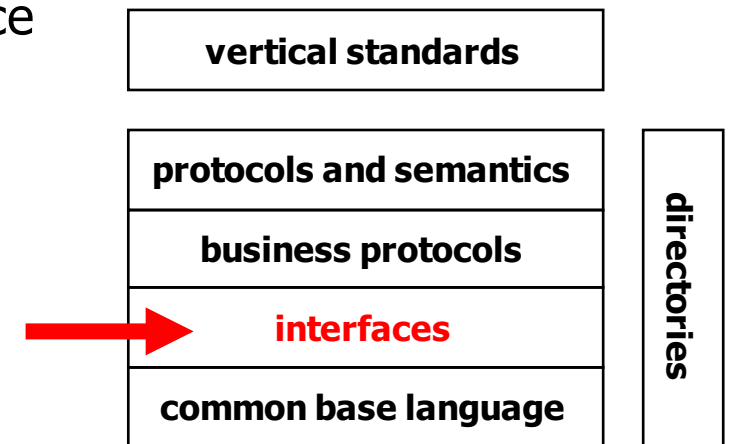
- XML isn't good at carrying non-XML things within it
- Introduces an outer multipart MIME envelope
- Root part is SOAP envelope
- Other parts can be anything: XML, images, ...

- WS-Addressing

- provides a mechanism to place the target, source and other important address information directly within the Web service message
 - decouples address information from any specific transport model
- w3c recommendation

Web Services Description Language (WSDL)

- Provides all information necessary to programmatically access a service
 - documentation for distributed systems
 - recipe for automating the details involved in applications communication
- Description of the logical web service interface
 - operations, parameters, ...
 - similar to IDL in conventional middleware
- Describes mechanism to access the web service
 - which protocol is used
 - SOAP, ...
 - service location
- WSDL standardization pursued by w3c
 - V1.1 specification is a w3c note
 - not an official standard, but most widely used
 - WSDL 2.0 is a w3c recommendation

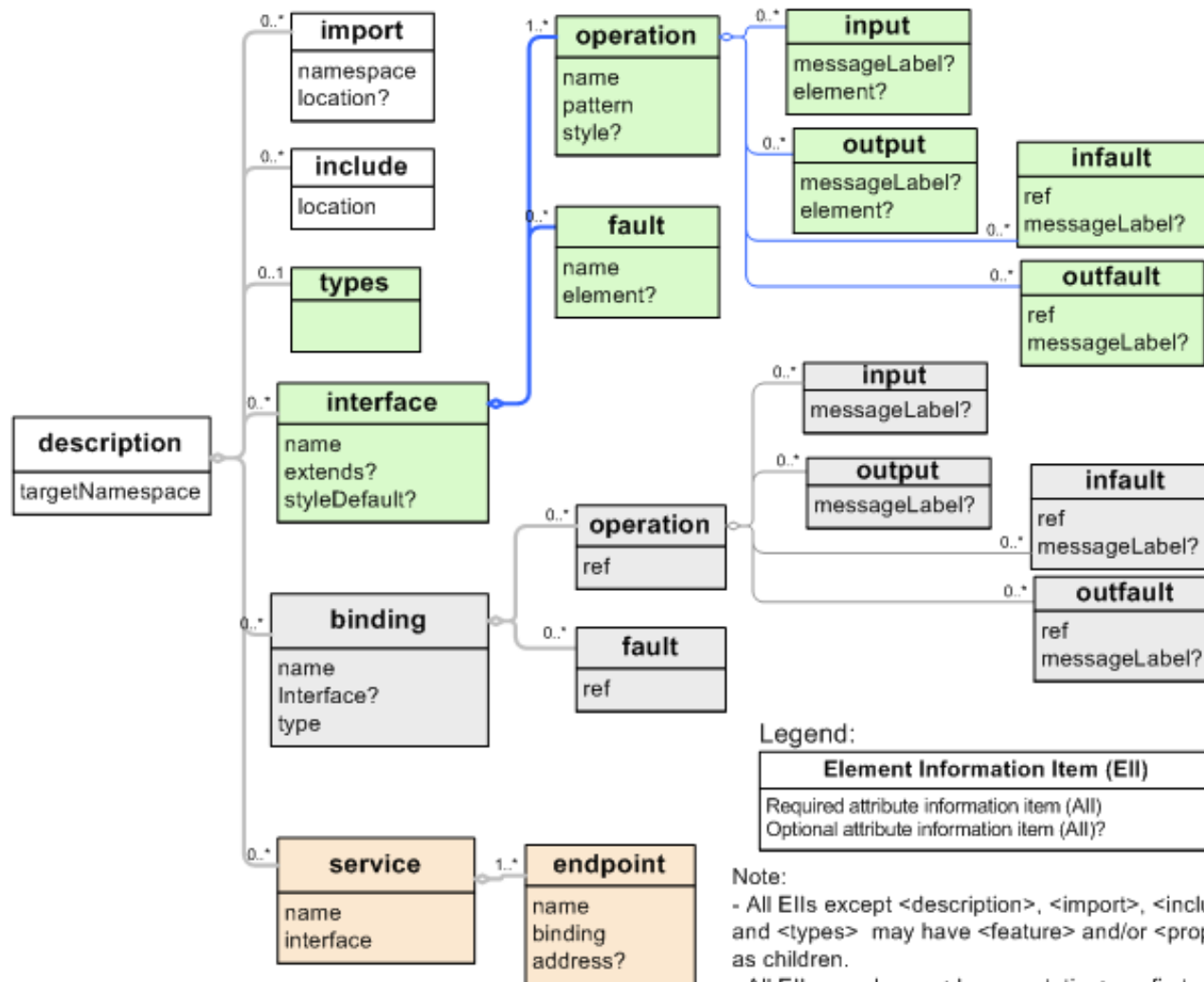


Service Description and Discovery Stack

Ingredients of WSDL

- Abstract part (typical scope of an interface definition language)
 - Types: Definitions of data types needed (e.g., message types)
 - uses XMLSchema as a default language
 - Message Exchange Pattern: Abstract definition of data exchange
 - defines sequence of input and output messages, independent of a specific service
 - Operation: Abstract actions supported by the service
 - Interface: Interface defined as set of operations
- Concrete part
 - Binding: Concrete protocol and data format used to implement an interface
 - E.g., SOAP/HTTP, HTTP GET/POST
 - the same service can be supported by multiple protocols/bindings simultaneously
 - Endpoint: Single individual "end point" identified by a network address supporting a particular binding
 - Service: Collection of related "end points"
 - Group endpoints related to the same service interface but expressed by different protocols (bindings)

WSDL 2.0 Document Structure



Message Exchange Patterns

- Defines interaction paradigms
 - exchange of several asynchronous messages
 - sequence and cardinality of messages in an operation
 - abstract: no message types, no binding-specific information is specified
 - minimal contract
- Standard MEPs defined by WSDL specification
 - in-bound MEPs
 - In-Only, Robust In-Only, In-Out, In-Optional-Out
 - out-bound MEPs
 - Out-Only, Robust Out-Only, Out-In, Out-Optional-In
 - Where to send to? Outside scope of WSDL
 - Information could be provided through another (subscribe) operation or defined at deployment time
- Extensibility – possible to define new MEPs

Interface

- Interface is a set of abstract operations
 - may extend other interfaces (i.e., multiple interface inheritance)
 - faults, operations, etc. are inherited
 - overloading of operations is not supported
 - inheritance conflicts must not occur
 - default style for operations can be specified
- Operation groups a set of abstract messages involved
 - references a MEP that defines sequence of messages
 - defines the structure of input, output, infault, outfault messages by referencing the appropriate (schema) types
 - optionally declares a style
 - rules used for generating messages, e.g., RPC style or Document style
 - may optionally be declared "safe"
 - no further obligations result from an invocation
- Interface Fault
 - definition of faults that can occur in the scope of this interface

Web Service Policies

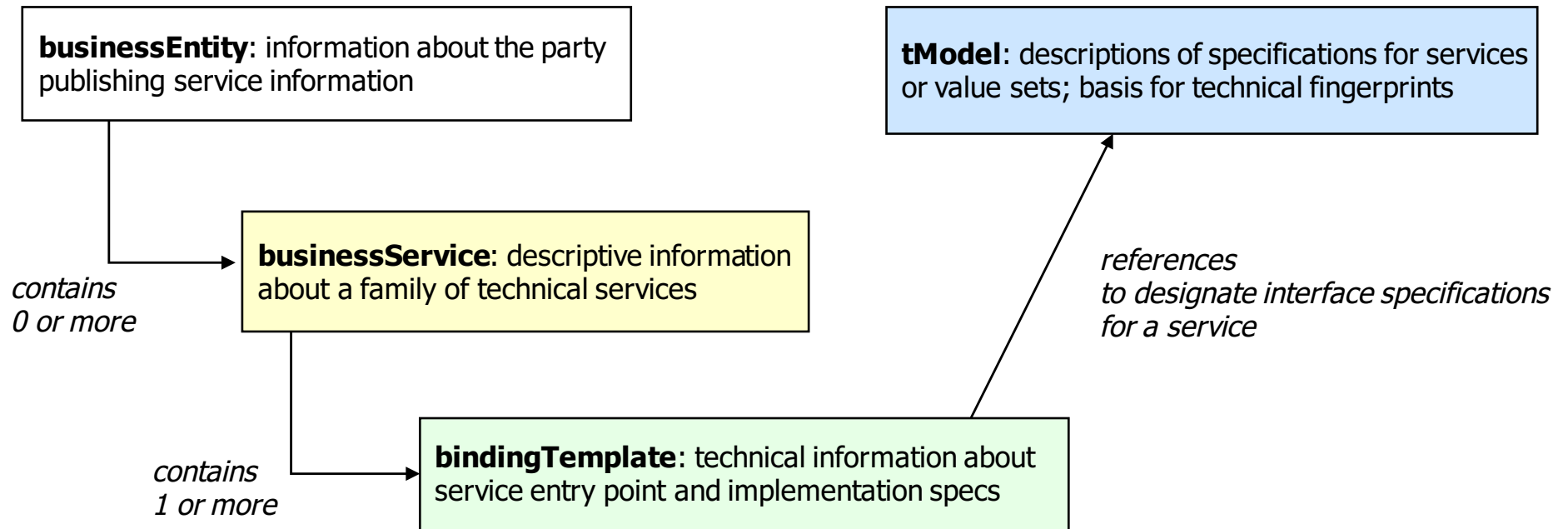
- Web service capabilities and requirements need to be described as (machine-readable) metadata
 - examples: addressing, security, transactions, reliability
 - allows tools to check for service compatibility, generate code
- WS-Policy
 - express capabilities, characteristics of entities in a WS-based system
 - policy **assertions**, **expressions**, statements
 - example:

```
<All>  
  <wsam:Addressing>...</wsam:Addressing>  
  <ExactlyOne>  
    <sp:TransportBinding>...</sp:TransportBinding>  
    <sp:AsymmetricBinding>...</sp:AsymmetricBinding>  
  </ExactlyOne>  
</All>
```
 - allows senders, receivers to specify their security requirements and capabilities
- WS-PolicyAttachment
 - associate policy expressions with subjects
 - reference policies from WSDL definitions or inline them in bindings
 - associate policies with UDDI entities

Universal Description Discovery and Integration (UDDI)

- Goal: enable service discovery
 - catalogue services based on published information of service providers
 - maintain taxonomy(ies) to support searching for appropriate services in business terms
 - specify technical binding information to actually communicate with the selected service
- UDDI registry serves as a directory of web services
 - Allows searching “by what” and “by how” instead of just “by name”
- UDDI defines
 - Set of schemas for describing businesses and their services
 - UDDI data model
 - SOAP API for accessing a UDDI registry
- UDDI initiative
 - Involves more than 300 companies
 - <http://www.uddi.org>

UDDI Core Data Structures

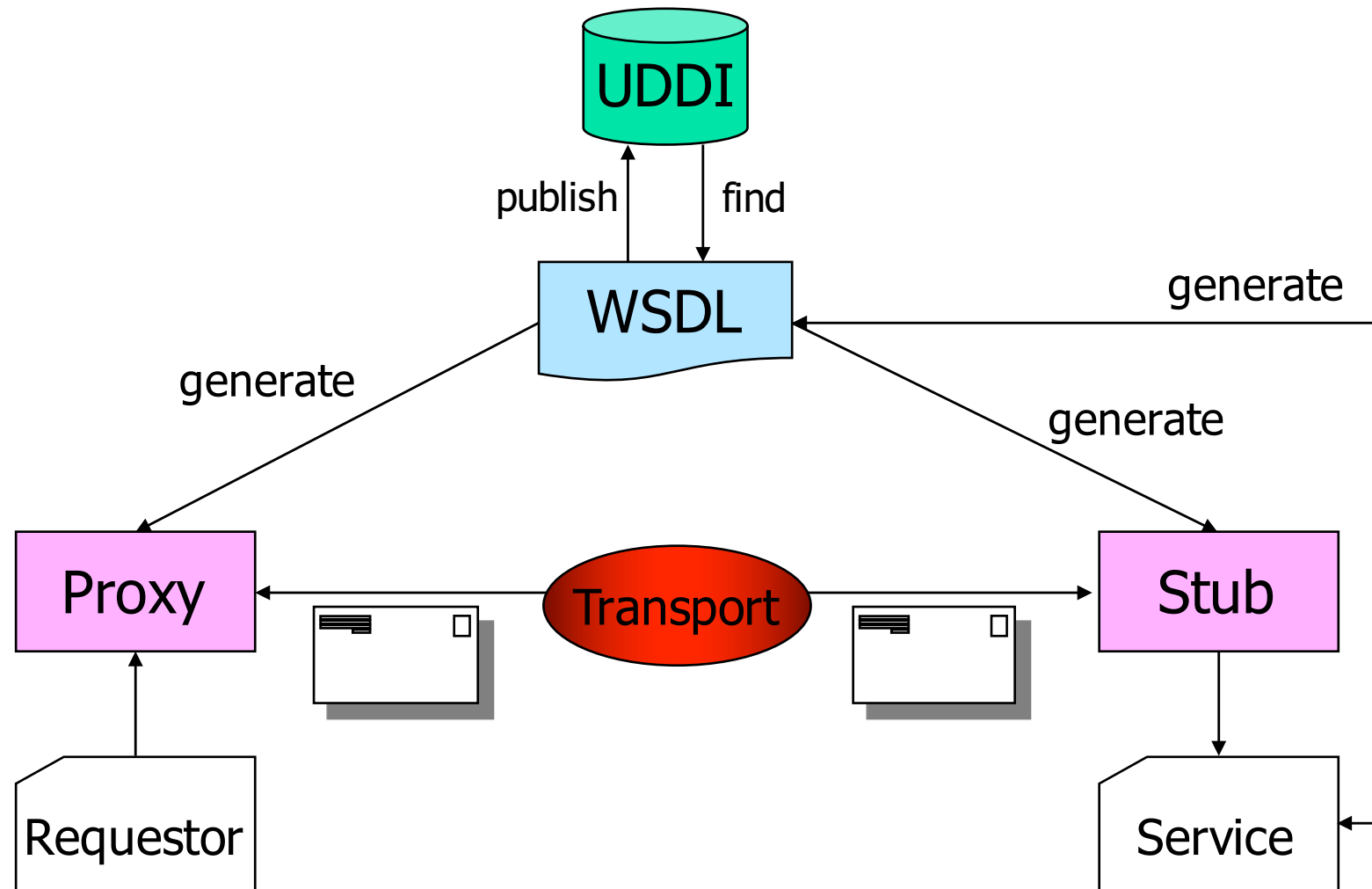


- UDDI key
 - uniquely identifies each instance of core data structures within a registry
 - basis for realizing the containment/referencing relationships (using foreign keys)
- XML Schema definition for UDDI Data Model

Registry Types

- Different types of registries
 - corporate/private (e.g., enterprise web service registry)
 - operates within the boundaries of a single company (or for a restricted number of partners)
 - data is not shared with other registries
 - affiliated (e.g., trading partner network)
 - registry is deployed in a controlled environment
 - limited access by authorized clients
 - data may be shared with other registries in a controlled manner
 - public (e.g., UDDI Business Registry)
 - open, public access to registry data
 - secured administrative access, content may be moderated
 - data may shared, transferred among registries
- UDDI Business Registry
 - public, global registry of businesses and their services
 - master directory of publicly available e-commerce services
 - was initial focus of UDDI effort

Application Development



Java API for XML Web Services (JAX-WS)

- API for building web services and clients based on remote procedure calls and XML
 - Goal: hide all the complexities of SOAP message processing
 - APIs for supporting XML based RPC for the Java platform
 - Define web service
 - Use web service
 - Defines
 - WSDL/XML to Java mapping
 - Java to XML/WSDL mapping
 - Core APIs
 - SOAP support (including attachments)
 - Client and Server Programming models involving generated stub classes
- Client side invocation (standard programming model)
 - Application invokes web service through generated stub class
 - JAX-WS runtime maps the invocation to SOAP, builds the SOAP message, processes the HTTP request
- Server side processing
 - JAX-WS runtime processes HTTP, SOAP message, maps to RPC and dispatches to target (class implementing the web service)

Mapping WSDL <-> Java – Example

WSDL 1.1 interface definition:

```
<!-- WSDL Extract -->
<message name="getLastTradePrice">
  <part name="tickerSymbol"
    type="xsd:string"/>
</message>
<message
  name="getLastTradePriceResponse">
  <part name="result"
    type="xsd:float"/>
</message>
<portType
  name="StockQuoteProvider">
  <operation
    name="getLastTradePrice"
    parameterOrder="tickerSymbol">
    <input message=
      "tns:getLastTradePrice"/>
    <output message=
      "tns:getLastTradePriceResponse"/>
  </operation>
</portType>
```

Java service endpoint interface:

```
//Java
public interface StockQuoteProvider
  extends java.rmi.Remote {
  float getLastTradePrice(
    String tickerSymbol)
    throws java.rmi.RemoteException;
}
```


Summary

- Service-oriented architectures
 - definition, access, discovery of (web) services
 - web services vs. REST services
- SOAP
 - defines SOAP message structure and messaging framework
 - stateless, one-way
 - more complex patterns "on top" (e.g., request/response)
 - provides convention for doing RPCs using SOAP
 - support for extensibility, error-handling, flexible data representation
 - independent of transport protocols
 - binding framework for defining protocol-specific bindings
 - SOAP/HTTP
 - extensions beyond SOAP for addressing, reliable messaging (see next chapter)

Summary (cont.)

- WSDL
 - supports description of all information needed to access a web service
 - interface, operation, message types
 - binding to specific protocol (e.g., SOAP)
 - protocol extensions
 - endpoint, service
- UDDI
 - registry
 - publish information about business, services provided, and the way to use them
 - white, yellow, green pages
 - tModels provide infrastructure for business and service "name space"
 - identification, classification of business, services, protocols, ...
 - can "point to" detailed service descriptions such as WSDL files
 - APIs for manipulating and inquiring about registry content
 - provided as web services

Summary (cont.)

- Application development
 - Integration with programming languages, existing middleware
 - Tooling support
- Programming language binding
 - WSDL as the "IDL for web services"
 - Mapping WSDL to PL (e.g., Java)
 - enables generation of client proxies, server stubs for web services invocation
 - Mapping PL to WSDL
 - "publish" existing functionality as a web service
- Web services support based on conventional middleware
 - define standards for reusing/extending existing programming models and middleware infrastructure to support web service
 - Java EE: use/publish servlets, stateless session beans to implement WSs (JAX-WS)
- REST (Representational State Transfer)
 - idea: model a set of services as resources with generic interface (get, put, post, delete)
 - "lightweight", stateless web services
 - Java EE: integration similar to web services, is based on JAX-RS API