• [ACKM04] Ch 6, 7
WSDL Specifications

- Abstract part
  - signatures, operations
  - like traditional IDL
- Concrete part
  - bindings / services / ports

WSDL specification

abstract part
- types
- messages
- operations
- port types

concrete part
- bindings
- services and ports
Concrete Part

- Interface Bindings
  - message encoding & protocol bindings
- Ports
  - interface binding + network address
- Services
  - logical groupings of ports
<?xml version="1.0"?>
<definitions name="Procurement"
    targetNamespace="http://example.com/procurement/definitions"
    xmlns:tns="http://example.com/procurement/definitions"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <message name="OrderMsg">
        <part name="productName" type="xs:string"/>
        <part name="quantity" type="xs:integer"/>
    </message>
    <portType name="procurementPortType">
        <operation name="orderGoods">
            <input message = "OrderMsg"/>
        </operation>
    </portType>
    <binding name="ProcurementSoapBinding" type="tns:procurementPortType">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="orderGoods">
            <soap:operation soapAction="http://example.com/orderGoods"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="ProcurementService">
        <port name="ProcurementPort" binding="tns:ProcurementSoapBinding">
            <soap:address location="http://example.com/procurement"/>
        </port>
    </service>
</definitions>
Automatic Generation of WSDL

- **service requestor**
  - application object (client)
  - stub
  - SOAP-based middleware

- **service provider**
  - application object (service provider)
  - skeleton
  - SOAP-based middleware

- **WSDL of service provider**

- **WSDL compiler (client side)**

- **WSDL compiler (server side)**

1. **WSDL generator**

2. SOAP messages
Universal Description Discovery and Integration

- The Web Services Directory Service
- Service Registry
  - for browsing by developers
  - for dynamic binding
- Business registry
  - original goal: global registry, every business and exported service registered there
  - now: support interaction between private and public UDDI registries
Information in UDDI

- Business entity
  - an organization that provides Web services
- Business Service
  - group of Web Services that cooperate in performing some business process
- bindingTemplate
  - technical information needed to use the service
- tModel
  - general container for a specification
Information in UDDI

- **businessEntity**
  - name
  - contacts
  - description
  - identifiers
  - categories

- **businessService**
  - service key
  - name
  - description
  - categories

- **bindingTemplate**
  - binding key
  - description
  - address
  - detailed info
  - references to tModels

- **tModel**
  - key
  - name
  - description
  - overviewDoc
  - identifiers
  - categories

Specs stored at the provider’s site

Stored in the UDDI registry
tModel Description

overviewDoc
(refer to WSDL specs and to API specs)

classification information
(specifies that this tModel is about XML, WSDL, and SOAP specs)

• Note overviewDoc has URL - refers to specification stored outside UDDI server
• Multiple overviewDocs are allowed
Sharing of tModels

- Each tModel is assigned a GUID when it is registered
- The tModels may be shared by multiple bindingTemplates
- Software can use UDDI to search for all instances of a given GUID
UDDI Registry API

- Inquiry
- search
- Publisher
- register and unregister operations
UDDI Registry API - II

- Security
- Custody and Ownership Transfer
  - create/destroy/move objects
- Subscription
  - monitor for changes in UDDI data
- Replication
  - readonly except at “owner” site
Architecture of Distributed UDDI

Service requestor

Service provider

SOAP/HTTP

SOAP/HTTPS

Inquiry API

Publishers API

Web service interface

Service descriptions

UDDI registry A

Subscription, Replication, and Custody transfer APIs (SOAP/HTTPS)

Service descriptions

UDDI registry B
WSDL in UDDI

- Service requestor
- Service provider

- WSDL service descriptions

- Web service interface
- Service descriptions

- UDDI registry

- Service requestor connects to service provider via SOAP/HTTP
- Service provider connects to UDDI registry via SOAP/HTTPS

- Inquiry API
- Publishers API
Private UDDI Registries

- Public Universal Business Registry not required for Web apps that are intra-enterprise or between trusted partners
- Three categories
  - public
  - private
  - shared
- Must support replication between registries where appropriate
Example: Database stored procedures

- Service provider
  - Service implementation
  - WSDL generator
  - Server stub
  - WSDL service descriptions
  - SOAP router
  - WSDL compiler
  - HTTP engine
  - UDDI publisher

- Business entity
  - Business service
  - Binding template
  - tModel

- Inquiry API
- Publishers API
- UDDI registry
“Advances” in SOAP

- BLOBs
  - now Web Services description yields interaction sequence but not data formats, which are agreed to by communicating applications
- Document exchange
  - constructing WSDL requires semantic understanding of the documents
Dynamic Binding

- Fully general
  - e.g. CORBA DII
  - now we have run-time types, introspection

- Restricted
  - e.g. all services using a given tModel
  - but enterprises usually won’t accept a program effectively signing a contract ...
Service Coordination

• Not Yet Implemented!

• But needed - business transactions comprise multiple method invocations that are not independent

• How specify valid and invalid sequences of invocations?
A Simple Conversation

The interaction between clients and services is often formed by a set of operation invocations (i.e., it is a conversation). A service provider may support some conversations while disallowing others.

The internal business logic of clients and Web services must support the conversation, and maintain the state across different operation invocations belonging to the same conversation.

• How ensure e.g. requestQuote happens before orderGoods?
State Machine Approach

- boxes = states
- edge labels = operation calls

- boxes:
  - quote requested
  - goods ordered
  - order canceled
  - order completed

- edge labels:
  - requestQuote
  - orderGoods
  - cancelOrder
  - makePayments
Multiple Servers?

1: requestQuote
2: orderGoods
3: checkShipAvailable
4: confirmOrder
5: makePayment
6: orderShipment
7: getShipmentDetail
8: confirmShipment
9: confirmShipment
• Simple and intuitive ...
• Drawback: cannot handle alternatives
Activity Diagram

Boxes are messages
Activity Diagram Slice

Consider only actions that will affect customer ...
Conversation Controller

- Conversation routing
- Protocol compliance checking
- Example: many clients all use protocol P
Conversational Controller - Routing

The controller dispatches messages to the appropriate implementation object. Clients invoke operations at the same address.
Conversation Controller in SOAP Router

- EJB
- EJB
- EJB
  EJB container

SOAP router
(with conversation controller)

conversation ID/object mapping

HTTP server

SOAP messages on HTTP transport

service provider
Identity of Conversation

- Web services must communicate port references and role information to the conversation controller infrastructure.
WS-coordination

- A framework for supporting coordination
- Standardizes:
  - passing GUID between interacting Web services
  - informing protocol handler about ports
  - informing protocol handler about roles
Coordinators and Participants

- Coordinators and participants
- Central or distributed coordinator

(a) Central coordination

(b) Distributed coordination
Terminology

• Coordination protocol
  • e.g. 2PC
• Coordination type
  • e.g. atomic transaction
• Coordination context
  • data structure identifying messages as belonging to a conversation
Interactions

- **Activation**
  - participant requests new context
- **Registration**
  - participant registers self with coordinator
- **Protocol specific interactions**
  - interactions specific to the protocol :-)

Activation

CreateCoordinationContext
- ... - coordination type - current context

CreateCoordinationContextResponse
- ... - coordination context - identifier - coordination type - registration service - ...

Web service

ActivationRequestorPortType

ActivationCoordinatorPortType

coordinator
Registration

**Web service**

- register
  - ...
  - protocol identifier
  - participant protocol service

- registerResponse
  - ...
  - - coordinator protocol service

**RegistrationRequestorPortType**

**RegistrationCoordinatorPortType**
• Convention for protocol specific messages

Diagram:

- Web service
- Coordinator
  - XCoordinatorPortType
  - protocol-specific messages from participant to coordinator
- Participant
  - XParticipantPortType
  - protocol-specific messages from coordinator to participant
Central Coordinator

1. create CC
2. X1
3. register
4. protocol coordinator
5. operational message
6. register
7. protocol coordinator
8. protocol-specific message
9. protocol-specific message
Distributed Coordinator

Web service A

1. create CC
2. X1
3. register
4. protocol coordinator
5. operational message

Web service B

6. create CC
7. X2
8. register
9. register
10. protocol coordinator
11. protocol coordinator
12. protocol message
13. protocol message
14. protocol message
15. protocol message

coordinator C_A

coordinator C_B
Coordinator Chaining
Summary of WS-Coordination

• Defines needed SOAP extensions
• Metaprotocols
  • activation
  • registration
• Middleware for central or distributed coordination
WS-Transaction

- Extension/Use of WS-coordination
- Two flavors:
  - Atomic transaction
  - Business activity
Atomic Transaction - Ports

- WS-Coordination interfaces
  - ActivationCoordinatorPortType
  - RegistrationCoordinatorPortType

- WS-Transaction interfaces
  - CompletionCoordinatorPortType
  - CompletionWithAckCoordinatorPortType
  - PhaseZeroCoordinatorPortType
  - 2PCCoordinatorPortType
  - OutcomeNotificationCoordinatorPortType

- WS-Transaction interfaces needed for chaining
  - CompletionParticipantPortType
  - CompletionWithAckParticipantPortType
  - PhaseZeroParticipantPortType
  - 2PCParticipantPortType
  - OutcomeNotificationParticipantPortType

- WS-Coordination interfaces needed for chaining
  - RegistrationParticipantPortType
Atomic Transaction - Protocol

Web service A | coordinator Ca | Web service B | coordinator Cb

- create CC
- T1
- register for Completion
- completion coordinator

operational message

create CC
- T2
- register for PhaseZero

register for PhaseZero
- PhaseZero coordinator

PhaseZero
- register for 2PC
- 2PC coordinator

complete

PhaseZero
- PhaseZeroComplete
- PhaseZeroComplete

2PC coordinator
Atomic Transaction - Protocol

- PhaseZero
  - PhaseZeroComplete
  - prepare
    - prepared
    - commit
      - committed
  - completed
Business Activities

- Business Agreement (with Complete)
  - participant initiates by informing the coordinator of its status:
    - exited/completed/faulted
  - Coordinator replies
    - close/complete/compensate/forget
Port Types

**WS-Coordination interfaces**
- ActivationCoordinatorPortType
- RegistrationCoordinatorPortType

**WS-Transaction interfaces**
- BusinessAgreementCoordinatorPortType
- BusinessAgreementWithCompleteCoordinatorPortType
- RegistrationParticipantPortType

**WS-Transaction interfaces needed for chaining**

**WS-Coordination interfaces needed for chaining**
Execution

Web service A
create CC
A1
operational message
Web service B
operational message
Web service C
register for BusinessAgreement
BusinessAgreement coordinator
completed
faulted
compensate
Coordinator R
register for BusinessAgreement
BusinessAgreement coordinator
completed
faulted
compensate
forget