

CmpE 596: Service-Oriented Computing

Pınar Yolum

`pinar.yolum@boun.edu.tr`

Department of Computer Engineering
Boğaziçi University

Course Information

- Topics
- Work Schedule
- Grading
- Resources
- Academic Integrity

Part I: Architectures

Internet Architectures

- A set of nodes collaborate to carry out a job
 - A node wants to print, but doesn't have access to a printer
 - A node needs data that is available at a different node
- A common language for communication
 - Usually not a full-fledged language
 - Protocol that specifies what to do in specific situations

Protocols

- Set of rules that will be followed by the participants
 - Events that take place
 - The initiators
 - The timing of events
 - The data formats
- Does not specify how the protocol should be implemented
- Example*: Hypertext Transfer Protocol (HTTP)

Traditional Protocol Properties (1)

- Unambiguous
 - The protocol state should state clearly what should be done in a situation
 - No misunderstandings
- Complete
 - The protocol should cover all possible requests
 - Garbled data?
 - Illegal request?

Protocol Properties (2)

- Extendable
 - The protocol should allow new requests and responses to be added
 - Versioning of protocols
 - World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) work to standardize versions of protocols
- Accessible
 - Different clients and different servers may be designed and implemented by different programmers
 - Should still be able to speak the same language

Protocol Types

- Synchronous
 - Client sends a request and blocks
 - Server responds
 - Example: HTTP, SMTP
- Asynchronous
 - Clients and server send information at the same time
 - Example*: TELNET
- Deferred Synchronous
 - Continue operation until a certain point and then wait
 - Example: CORBA

Client/Server Architecture (1)

- Client and server asymmetric in capabilities
- Client*
 - Represents a user
 - Program that request tasks from servers
- Often users interact with a client through a GUI
- Clients translate the user requests to protocol tokens
- Clients initiate the interaction with a server

Client/Server Architecture (2)

- Server: Program that waits for incoming communication requests from a client
- Usually has some resources that the client does not have
 - Bandwidth, access to printer
- Takes the request
 - Process data
 - Perform a task
 - Return results client
- Examples: Mail servers, Print servers, Web servers

Two-Tier Architecture

- Example*: `registration.boun.edu.tr`
 - What does the client have? Database, logic?
 - Who uses the client?
 - What does the server have? Database, logic?
 - Two tiers?

Three-Tier Architecture

- Three separate layers
- Presentation tier
 - Runs on client
 - Provides user interface
 - Invokes business logic
- Business logic tier
 - Runs on server
 - Has application logic, business rules, etc.
- Data tier
 - Runs on a database server
 - Stores and provides access to data
- Advantages?
- N-Tier?

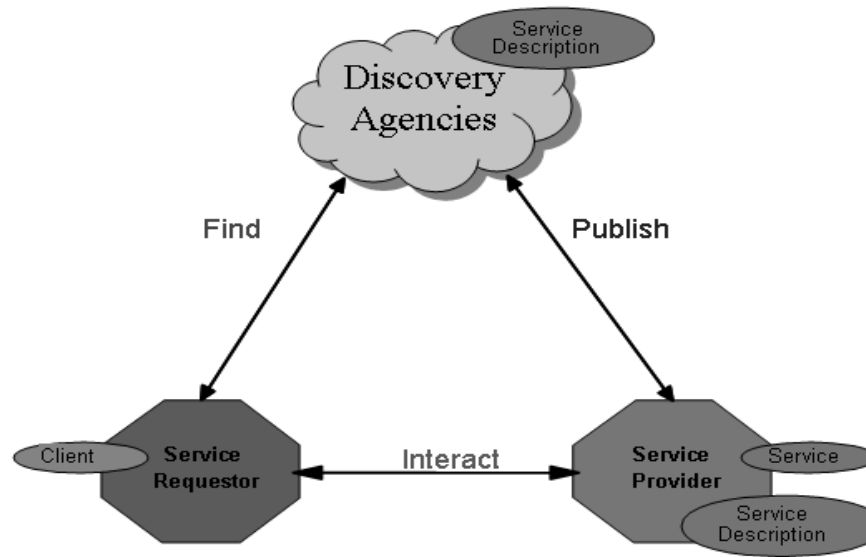
Invocation vs. Message-Oriented

- Invocation
 - Assumes knowledge of the other party
 - Gives access to others' resources
- Message-Oriented
 - Recipient deals with the message
 - Recipient can change its functioning
 - Increased abstraction

Service-Oriented Architectures (SOA)

- Separate service implementation from the interface
 - No need to know the internal implementation
 - Follow a previously agreed protocol
- Find-Bind-Execute Paradigm

Service Oriented Architecture



SOA Entities

- Service Consumer
 - Locates the Producer in the Registry
 - Initiates the communication
 - Follows a Contract
- Service Producer
 - Delivers services
 - Advertises its services in a Registry
- Service Registry
 - Stores advertisements
 - Allows lookup service to Consumers

SOA Concepts

- Service Contract
 - Specifies how the Consumer and the Producer will interact
 - Specifies the necessary conditions for the Producer to execute a task
 - Defines the QoS requirements

SOA Properties (1)

- Entities are autonomous
 - Resources owned and managed by individuals
 - Choose how they will carry out their tasks
 - Choose whom they will carry out business with
- Entities are dynamic
 - Entities can change their behavior
 - Entities may not always be available
- Entities are interoperable
 - Entities can communicate even if they are written in different languages or run on different platforms
 - Standard protocols or data formats should be available

SOA Properties (2)

- Services are loosely coupled
 - Established by contracts and dynamic binding
 - No dependency on the service implementation
- Services are composable
 - Put together to offer a composite service
 - Dependencies between the services should be handled
- Services are negotiable
 - Change service characteristics based on demand
 - Negotiation works both ways

Part II Web Services Overview

Component-Based Development

- Groups of objects
- Provides application functionality
 - Rather than access to individual data items
- Components communicate to yield enhanced functionality
- Components are composed and compiled at design time

Service-Based Development

- Allows late binding
 - Consumer looks up a service in a registry
 - Possibly chooses among several possibilities
 - Binds to the one it selects
 - Enacts the service
- Web-based standards
- Standardized data formats

Web Services Stack (1)

- Service Transport
 - Transfer data between different nodes
 - HTTP used most often
 - Not affected by firewalls
 - Connectionless and stateless: Independent requests and responses
- Service Messaging
 - Extensible Markup Language (XML)
 - Self-describing messages
 - Data structured as a tree
 - Simple Object Access Protocol (SOAP)
 - Defines how data is packaged in an XML message
 - Contains an envelope, a header, and a body

Web Services Stack (2)

- Service Description
 - Functionalities that the service provides
 - Set of acceptable messages
 - Protocol with which consumers can bind and communicate with the service
- Service Registry
 - Universal Description, Discovery and Integration (UDDI) Registry
 - Itself a Web service
 - Allow service providers to publish information
 - Allow service consumers to find Web services for given service characteristics
 - Communication through SOAP messages

Web Services Stack (3)

- Service Composition
 - Each Web service is thought of carrying out small task
 - Combine tasks from different Web services into one large transaction
 - Example:
 - Web service A can be used for booking a flight.
 - Web service B can be used for reserving a hotel room
 - Compose them into one service to arrange the entire trip.
- Business Process Execution Language for Web Services (BPEL4WS)

Quality of Service

- Availability
 - When can it be used? Now? At certain intervals?
 - Metrics for measuring availability.
 - Example: Time-to-repair (TTR)
- Accessibility
 - Extent of finishing the requested service
 - Metrics for measuring accessibility
 - Example: Success rate
- Can a Web service be available but not accessible?

Quality of Service (2)

● Performance

- Throughput: How many service requests can be handled by the Web service in a unit time?
- Latency: How long does it take to get a response to a request?
- Maximize throughput, minimize latency

● Reliability

- Can it guarantee the same performance over a period of time?
- How many failures take place in a period of time?

Quality of Service (3)

● Integrity

- How correctly is the source executed?
- All tasks need to be performed in the correct order
- Otherwise, roll back

● Security

- Provide authorization and authentication for accessing resources
- Preserve confidentiality of private consumer information

Example Applications

- Travel planning
 - Simple, individual services
 - Can be composed in various ways
 - Some service providers may be preferred over other

Part III: Standards

Simple Object Access Protocol (SOAP)

- XML-based protocol for exchanging structured messages
- Independent of platform or programming language
- Can use various transport protocols; commonly HTTP
- Can be extended easily for different needs
 - Basis for other Web service activities (security, trust)
 - Layered design

SOAP Messages

- Is contained in an envelope
- Consists of a header (optional) and a body (mandatory)
- Destined from a SOAP sender to a SOAP receiver along a message path
- There may be intermediaries in between
- Intermediaries may inspect and modify the header
- Messages may be transferred over any network protocol

Internals of SOAP Message

- Body contains main matter
- Header contains control information
 - How should the message be processed by the SOAP receiver?
 - How should the message be handled by intermediaries?
- Header consists of header blocks

Message Pattern

- The message is finally received by whoever claims to be the ultimateReceiver.
 - What would be a problem?
- SOAP Request-Response
 - Response can change the body as well as the header
- SOAP Response
- SOAP requests may involve remote procedure calls (RPCs)

SOAP RPC

- The address of the target SOAP node
- Method to be invoked and its arguments
- Separation of identification data from control data
- Message exchange pattern

Protocol Bindings

- Specification of how messages will be transported among SOAP nodes
 - Serialize the message content; typically by XML serialization
 - Support additional features such as correlating requests with responses
- Must support SOAP message patterns (such as Request/Response)
- SOAP specification allows Web methods such as GET and POST
- Current available binding is HTTP with GET and POST

Web Service Description Language (WSDL)

- XML-Based language for describing services
 - Abstract message formats (like SOAP)
 - Bound to a network protocol such as HTTP
 - Extendible
- Usage
 - Publish the WSDL in UDDI Registry
 - Consumer access the WSDL description of the service provider
 - Automatically read the WSDL and bind to the service

WSDL Terminology

- Types: A container for data type definitions
- Message: Definition of contents; typed XML
- Operation: Definition of an action supported by the service
- Port Type: Set of operations supported by an endpoint
- Binding: Concrete protocol and data format specification
- Port: An endpoint with a network address and binding
- Service: A collection of endpoints

WSDL Types

- Data type definitions for exchanged messages
- WSDL doesn't have its own type system
- Default: XSD, but other type systems can be used

```
<types>
  <schema targetNamespace="http://example.com/test.xsd"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    <!-- Define types and possibly elements here -->
  </schema>
</types>
```

WSDL Definitions

- Contains typed logical parts
- Unique message name within the WSDL document
- Unique part name within each message
- Parts define message content

```
<message name="PO">  
  <part name="composite" type="tns:Composite"/>  
</message>  
</definitions>
```

WSDL Port Types

- Set of abstract operations and abstract messages
- Unique port name inside WSDL document
- Four transmission primitives (operations)
 1. One-way: Endpoint receives a message
 2. Request-response: Endpoint sends a response after receiving the message
 3. Solicit-response: Endpoint sends a message and receives a response
 4. Notification: Endpoint sends a message
- Primitives 2 and 3 can be represented with Primitive 1

WSDL Binding

- Binding of operations and messages to actual protocols
- Defines how a WSDL document will be communicated in another protocol
- Must specify exactly one protocol (e.g., SOAP)
- Should not specify any address information

WSDL Ports

- Defines an endpoint by a single address for binding (e.g., by soap:address)
- Unique name among all ports in a WSDL document
- Contains a defined *binding* attribute
- Must not specify more than one address

WSDL Services

- Groups one or more ports together
- Unique name among all services in a WSDL document
- Ports do not communicate with each other
- Ports with same port types but different bindings (or addresses) are considered alternatives

Service Registry

- Registry to record service providers and service descriptions
 - UDDI
 - ebXML
- Enables lookup from service consumers
- Must be managed by an independent organization or a company

UDDI

- Universal Description, Discovery, and Integration
- Consists of White Pages, Yellow Pages, and Green Pages information
- White Pages
 - Information about the identity of the business
 - Business name, contact information (address, Web site), identification number (Data Universal Numbering System (DUNS))
- Yellow Pages
 - Information about the service offering in terms of taxonomies
 - Type of business, products, location

UDDI

- Green Pages
 - Technical information about service offering
 - Interaction methods for the service; WSDL file of the service
- Stores information for classification and identification
- Registered information can be checked for validity
- Duplicated in other UDDI registeries regularly (similar to DNS)
- Users can lookup the service from any of the UDDI Registeries

UDDI Data Model

- Five basic data types
 - **businessEntity**: Representation of the registered business
 - **businessService**: Name and description of the registered service
 - **bindingTemplate**: Binding information; address for accessing it
 - **tModel**: A set of information that uniquely identifies the service; supports searches
 - **publisherAssertion**: Associates two or more businessEntity representations with a particular relation type

UDDI Data Model

- A Business Entity
 - Requires a unique business key
 - Can contain name, description, URLs for additional information
- A Business Entity can contain zero or more Business Services
- A Business Service
 - Requires a unique business and service key
 - Contains a binding template
 - Can contain name, description, etc.
- A Business Service can contain zero or more Binding Templates

Binding Template

- A Binding Template
 - Requires a unique binding key and a service key
 - Specifies the access point to bind to the service (such as mailto:, http:)
 - Specifies optional redirection information
 - Lists tModel structure
- A Binding Template can reference one or more tModels

tModel

- A tModel
 - Contains a unique tModel key and a pointer to the operator
 - A mechanism to exchange metadata about a Web service (service description, or a reference to a WSDL file)
 - Not specific to WSDL; other protocols are also supported.
 - Assigned a UUID when stored
 - Uniquely identifies a Web service

tModel

- Enables compatibility of services among multiple entries
- Supports registry lookups for services
 1. Search for a tModelKey for the service you are interested in
 2. Search registered services (and thus businesses) for that tModelKey
- A business can have several tModels
- A tModel can be shared by multiple businesses toKey) and their relationship (keyedReference)

Example Scenario

Your company will register its services with a UDDI registry.

- Choose an operator. Updates are done on this operator.
- Get an authorization token from the operator
- Register the necessary information (possibly using a UDDI client)
- Complete the tModel information to enable searches
- Update the information as needed

UDDI Messages

- Three types of messages are exchanged
 - Inquiry Messages: Requesting objects (Ex: FindService, FindtModel)
 - Publish Messages: Create or updata UDDI entry (Ex: DeleteService)
 - Response Messages: Responses of UDDI registry (Ex: BindingDetail)