Topics on Web Services

COMP6017

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk
2013-2014
Module Aims

• Introduce you to service oriented architectures
• Introduce you to both traditional and RESTful Web Services
• Give you in-depth knowledge of Web Services
• Give you practical hands-on experience of RESTful Web Services
People

Nick Gibbins

Enrico Costanza

Office hours for nmg: Friday 1100-1200
Module Structure

One double lecture each week

- Friday 0900-1100, 05/2015
- Note: extra slots in timetable in week 4 are incorrect

Assessment

- 75% Exam
- 25% Group Coursework
Coursework

One coursework worth 25% of your final mark

- Groups of three
- Construct a RESTful web service and client using JavaScript and node.js
- Coursework specification published on Friday of week 5
- Deadline on Tuesday of week 11 (10th December)
- Feedback by the end of week 12
Teaching Schedule

Week 1  Overview
Web Services Architecture
Architecture of the World Wide Web

Week 2  CANCELLED

Week 3  Web Protocols: HTTP
REST and Resource Oriented Architectures

Week 4  Web Protocols: SOAP
REST in Practice

Week 5  Service Description: WSDL
Introduction to JavaScript and node.js
Teaching Schedule

Week 6  Service Discovery: UDDI
        Coursework Briefing

Week 7  Addressing and Policy
        Coursework Support

Week 8  Security
        Coursework Support

Week 9-11  Coursework Support

Week 12  Review
Service Orientation
Service Orientation

• Component-based software design paradigm
• Organise and use heterogeneous distributed capabilities

• Many existing technologies:
  – Java RMI, CORBA, DCOM, WCF, Web Services, REST
What is a service?

• Services as contractually defined behaviours
• Services as task-performing components
• Services as collections of related capabilities
• Services combine information and behaviour
8 Principles for Service Orientation

• Loose coupling
• Shared formal contracts
• Abstraction
• Composability
• Reusability
• Autonomy
• Statelessness
• Discoverability
Loose Coupling

Coupling is a measure of the degree of dependency between components

- Tight coupling limits flexibility
- Loose coupling promotes ad hoc reuse of components
Shared Formal Contract

Services have descriptions that document:

- programmatic interface
- communication requirements and protocols
- constraints
- usage policies
Abstraction

Service contract describes the external view of a service

- service internals are hidden
- limits formation of dependencies (loosens coupling)

Service internals may change with minimal impact on clients
Composability

Service abstraction allows the encapsulation of other services

- internally, a service may be a client of other services
- services may aggregate several services
Reusability

Separation of concerns encourages reuse of components

- Service contracts describe services to clients
Autonomy

Services exist independently

Services control their underlying logic
  - subject to commitments made in service contracts
Statelessness

State consists of data specific to current activity
  - State management consumes resources

Statelessness increases scalability and availability
Discoverability

Service discovery is key to SOA

- service contracts describe services to facilitate consumption by potential clients
- clients may search for services by the features of their contracts
What is an architecture?

• Logical architecture
• Process architecture
• Development architecture
• Physical architecture
Logical Architecture

Primarily supports the functional requirements

- i.e. what the system should provide in terms of services to its users.

The system will be decomposed into a set of abstractions, and their high level interactions will be identified
Process Architecture

Takes into account some non-functional requirements, such as performance and availability

Addresses issues of concurrency and distribution, of system integrity, of fault-tolerance
Development Architecture

Focuses on the actual software module organisation, including libraries
Physical Architecture

Takes into account primarily the non-functional requirements of the system

- availability
- reliability
- performance
- scalability
... what about COMP6017?

- We are essentially going to cover a logical architecture, identifying core functionality offered by Web Services.

- Some elements of process architecture, and in particular concurrency and distribution, will be addressed too.

- We discuss the development architecture, when investigating REST.
Introduction to Web Services
<table>
<thead>
<tr>
<th>구분</th>
<th>프로토콜명</th>
<th>표준화 예상 시기</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Management</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distributed Management</td>
<td>WSDM(OASIS)</td>
</tr>
<tr>
<td>Security</td>
<td>WS-Security(OASIS)</td>
<td>S</td>
</tr>
<tr>
<td>Security Policy</td>
<td>WS-SecurityPolicy(N/A)</td>
<td>S</td>
</tr>
<tr>
<td>Secure Conversation</td>
<td>WS-SecureConversation (N/A)</td>
<td>S</td>
</tr>
<tr>
<td>Trusted Message</td>
<td>WS-Trust(N/A)</td>
<td>S</td>
</tr>
<tr>
<td>Discovery</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Discovery</td>
<td>UDDI(OASIS)</td>
</tr>
<tr>
<td></td>
<td>Publication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inspection</td>
<td>WSIL(N/A)</td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Portal</td>
<td>WSRP(OASIS)</td>
</tr>
<tr>
<td></td>
<td>Transaction</td>
<td>WS-Transactions(N/A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WS-Coordination(N/A)</td>
</tr>
<tr>
<td>Orchestration</td>
<td></td>
<td>BPEL4WS(OASIS)</td>
</tr>
<tr>
<td></td>
<td>WS-Choreography(W3C)</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>Presentation</td>
<td>WSIA</td>
</tr>
<tr>
<td></td>
<td>Policy</td>
<td>WS-Policy(N/A)</td>
</tr>
<tr>
<td></td>
<td>Implementation Interface</td>
<td>WSDL(W3C)</td>
</tr>
<tr>
<td>Transport</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Routing/Addressing</td>
<td>WS-Addressing(N/A)</td>
</tr>
<tr>
<td></td>
<td>Reliable Messaging</td>
<td>WS-ReliableMessaging (OASIS)</td>
</tr>
<tr>
<td></td>
<td>WS-Reliability(OASIS)</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>Packaging</td>
<td>SOAP(W3C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WS-Attachments(IETF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIME(IETF)</td>
</tr>
<tr>
<td></td>
<td>Transport</td>
<td>HTTP, TCP, SMTP, etc</td>
</tr>
</tbody>
</table>
Web Service Definition

• A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network.

• It has an interface described in a machine-processable format (specifically WSDL).

• Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

(Web Service Glossary)
Discussion

• Not universally accepted

• Too minimalist: does not mention policies, choreography, security

• Too specific, refers to standards that are not universally accepted (e.g., wsdl)
Service Definition

• A service is a mechanism to enable access to a set of capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.

• A service is provided by one entity for use by others, but the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider. (OASIS SOA Reference Model)
Engaging a Web Service

1. Parties "become known" to each other

2. Agree on semantics & WSD

3. Input Semantics & WSD

4. Interact

Requester Entity

Provider Entity

Requester
Human

Provider
Human

Requester
Agent

Provider
Agent

Sem + WSD

Sem + WSD

Sem + WSD
Roles (1): service vs. agent

- A **Web Service** is an abstract notion that must be implemented by a concrete agent.

- The **Agent** is the concrete piece of software that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided.
Roles (2): provider vs requester

• The service provides functionality on behalf of its owner (a person or organisation): the **provider entity**.

• A **requester entity** is a person or organization that wishes to make use of a provider entity's Web service.

• A requester entity uses a **requester agent** to exchange messages with the provider entity's **provider agent**.
Roles (3): description

• The mechanics of the message exchange are documented in a Web Service **description**.

• It defines the
  
  - message formats,
  
  - datatypes,
  
  - transport protocols, and
  
  - transport serialization formats

that should be used between the requester agent and the provider agent.
Roles (4): semantics

• The **semantics** of a Web service is the shared expectation about the behaviour of the service, in particular in response to messages that are sent to it.

• In effect, this is the **contract** between the requester entity and the provider entity regarding the purpose and consequences of the interaction.
Engaging a Web Service

1. The requester and provider entities become known to each other (or at least one becomes known to the other);

2. The requester and provider entities somehow agree on the service description and semantics that will govern the interactions between them;

3. The service description and semantics are exploited by the requester and provider agents;

4. The requester and provider agents exchange messages. (I.e., the exchange of messages with the provider agent represents the concrete manifestation of interacting with the provider entity's Web service.)
Web Service Models
Web Service Models

• Message Oriented Model
• Service Oriented Model
• Resource Oriented Model
• Policy Model
Message Oriented Model

Focuses on messages, message structure, message transport and so on
Service Oriented Model

Focuses on aspects of service, action and so on.

- In any distributed system, services cannot be adequately realized without some means of messaging, the converse is not the case: messages do not need to relate to services.
Resource Oriented Model

Focuses on resources that exist and have owners.
Policy Model

Focuses on constraints on the behaviour of agents and services.

- Generalize to resources
- Policies can apply equally to documents (such as service descriptions) as well as active computational resources.
Conclusion

• No agreed definition of what a Web Service is, but a reasonable intuition

• Not a single vision for Web Services (multiple standardisation committees, etc)

• Multiple and competing WS stacks

• Architecture does not discuss resources with state