Homework #3
Web Service Programming

UDDI

Aim

- To learn the followings:

  - The data structure of UDDI
  - Publish and find UDDI business information using Java
  - Setup a private UDDI registry

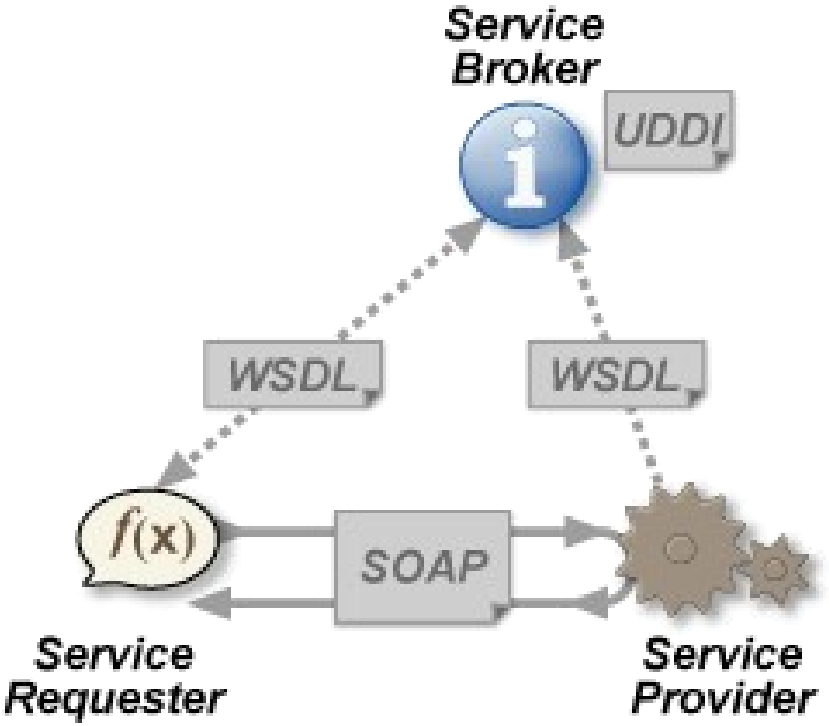
Most of the materials and codes for this presentation is taken form *SUN JWSDP 2.0* .

http://java.sun.com/webservices/downloads/previous/webservicespack.jsp

Part 1

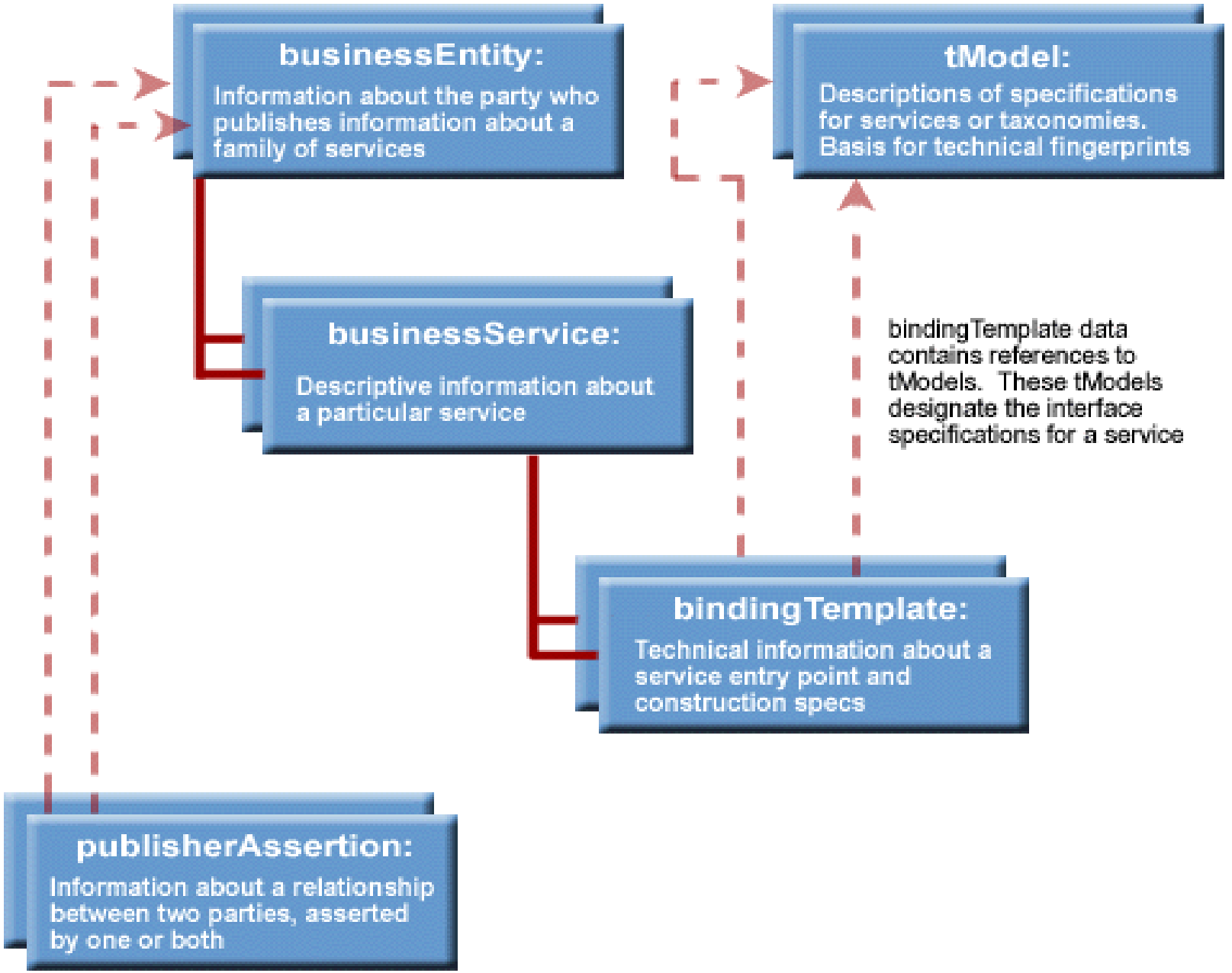
Registry Programming

General Architecture



Picture from ***WebPet*** : http://cit3.ldl.swin.edu.au/wikka/wikka.php?wakka=WebServices

UDDI Schema



Picture from ***SUN*** documents: http://java.sun.com/developer/technicalArticles/WebServices/WSPack/

Managing Registry Data

Steps to adding a web service description into registry :

1. Make Connection
2. Getting Authorization from the Registry
3. Create a Business Entity (Organization obj.)
4. Setting up a Classification
5. Adding Business Service to Business Entity
6. Adding Service Binding Template to Business Entity

6.Publishing Business Entity (Organization obj)

7. Publishing a Specification Concept (e.g. WSDL)

Make Connection (1)

```
// -------------------- Set Connection properties --------------------

Properties connProps = new Properties();
connProps.setProperty("javax.xml.registry.queryManagerURL",
http://localhost:8080/RegistryServer/);
connProps.setProperty("javax.xml.registry.lifeCycleManagerURL",
                      http://localhost:8080/RegistryServer/);
connProps.setProperty("javax.xml.registry.factoryClass",

"com.sun.xml.registry.uddi.ConnectionFactoryImpl");
```

Make Connection (2)

```
// -------------------- Set Connection Factory --------------------
ConnectionFactory factory =   ConnectionFactory.newInstance();
factory.setProperties(connProps);
Connection conn = factory.createConnection();



// ---------------- Getting Registry service Object ---------------

RegistryService rs = conn.getRegistryService();
BusinessQueryManager bqm = rs.getBusinessQueryManager();
BsinessLifeCycleManager blcm = rs.getBusinessLifeCycleManager();
```

Authorization

```
String username = "testuser";
String password = "testuser";

// Get authorization from the registry
PasswordAuthentication passwdAuth =
new PasswordAuthentication(username, password.toCharArray());

HashSet<PasswordAuthentication> creds =
new HashSet<PasswordAuthentication>();

creds.add(passwdAuth);
connection.setCredentials(creds);
```

Creating Business Entity - 1

Organization Data Structure:

1. Name : name of organization
2. Description : business description
3. Business key : ID by which Organization is known to Registry. The ID is created by Registry
4. Contacts : The contact information for the business. It includes phone, address, email,...
5. AuthorizedName: Individual who published the information. It should be a Registry authorized user .
6. Category bag: classifications info of the business entity
7. Business Service : list of services offered by business entity.

....

Creating Business Entity - 2

```
//------- Define Name and Description
InternationalString s = blcm.createInternationalString("AddNumber");
Organization org = blcm.createOrganization(s);

s = blcm.createInternationalString("Add Number service developed just
for testing purpose");
org.setDescription(s);

//------- Create primary contact, set name
User primaryContact = blcm.createUser();
PersonName pName = blcm.createPersonName("Shahab");
primaryContact.setPersonName(pName);

//-------- Set primary contact phone number
TelephoneNumber tNum = blcm.createTelephoneNumber();
tNum.setNumber("(08) 111-1111");
Collection<TelephoneNumber> phoneNums =  new
ArrayList<TelephoneNumber>();
phoneNums.add(tNum);
primaryContact.setTelephoneNumbers(phoneNums);
```

Creating Business Entity - 3

```
//------ Set primary contact email address

EmailAddress emailAddress =  blcm.createEmailAddress("shahabm@kth.se");

Collection<EmailAddress> emailAddresses =  new ArrayList<EmailAddress>();
emailAddresses.add(emailAddress);
primaryContact.setEmailAddresses(emailAddresses);

...

// Set primary contact for organization
org.setPrimaryContact(primaryContact);
```

- Next Step is Classifying the created Business Entity under either a known Taxonomy like (NAICS) or our own Custom Taxonomy.

Adding Classification

Steps:

1. Find parent taxonomy for the organization.
2. Create new classification based on the located Taxonomy

```
// Set classification scheme  (Taxonomy) to NAICS
ClassificationScheme cScheme =
bqm.findClassificationSchemeByName(null,"ntis-gov:naics:1997");

// Create and add classification
InternationalString sn =
  blcm.createInternationalString("Simple Data Processing Services");
String sv = "514211";
Classification classification =
  blcm.createClassification(cScheme, sn, sv);

Collection<Classification> classifications =  new
ArrayList<Classification>();
classifications.add(classification);

// Set organization's classification
org.addClassifications(classifications);
```

Adding Business Service Information (yellow pages)

Business Service Data Structure :

Name : name of service family

Description:

Service Key (generated by registry): uinque identifier for a particular service

Business Key : attribute that references the businessKey of the businessEntity

Category bag : Classification (like organization classification ,optional)

Binding Template : technical information about a service

Adding Service and Service Binding - 2

```
Collection<Service> services = new ArrayList<Service>();

InternationalString istr = blcm.createInternationalString("Add Number
Service ");

Service service = blcm.createService(istr);

istr = blcm.createInternationalString("Add Number Service Description");
service.setDescription(istr);

....
```

Adding Service Binding Template (green pages)

Service Binding Data Structure :

- bindingKey – a required attribute that contains a unique identifier that is assigned by the operator node upon registration
- serviceKey – an attribute that references the *serviceKey* of the *businessService* element.
- description – an optional element that contains brief descriptions of Web services
- accessPoint – an element that states where to access Web Service

- tModelInstanceDetails – collection of tModelKey attributes found in the tModelInstanceInfo elements together form the "technical fingerprint" of a Web service that can be used to identify compatible services.

Adding Service Binding Template-2

```
// Create service bindings
Collection<ServiceBinding> serviceBindings =  new
ArrayList<ServiceBinding>();

ServiceBinding binding = blcm.createServiceBinding();
istr = blcm.createInternationalString("Add Number Service Binding " +
"Description");
binding.setDescription(istr);

// allow us to publish a fictitious URI without an error
binding.setValidateURI(false);
binding.setAccessURI("http://localhost:10080/jaxws-fromjava/addnumbers");
serviceBindings.add(binding);

// Add service bindings to service
service.addServiceBindings(serviceBindings);

// Add service to services, then add services to organization
services.add(service);
```

Publishing Business Entity

```
Collection<Organization> orgs = new ArrayList<Organization>();

orgs.add(org);

BulkResponse response;
response = blcm.saveOrganizations(orgs);

if (response.getStatus() == JAXRResponse.STATUS_SUCCESS) {

      Collection keys = response.getCollection();
      Iterator keyIter = keys.iterator();
        while (keyIter.hasNext()) {
              Key orgKey = (Key) keyIter.next();
              String id = orgKey.getId();
              System.out.println("Organization key is " + id);
          }
}
```

Creating tModel and referencingto WSDL -1

Service binding can have a technical specification (e.g WSDL).

Steps:

1- Create Concept object

2- Add WSDL document to Concept object as an ExternalLink

3- Classify the Concept object as a WSDL document.

```
//--------- Create Concept and as an External Link ---------------

Concept specConcept;
specConcept = blcm.createConcept(null, "AddNumberConcept", "");
InternationalString s =  blcm.createInternationalString("Concept
description for AddNumber Service");

specConcept.setDescription(s);
ExternalLink wsdlLink = blcm.createExternalLink("
http://localhost:10080/jaxws-fromjava/addnumbers?wsdl","AddNumber
WSDL document");

specConcept.addExternalLink(wsdlLink);
```

Adding tModel and WSDL -2

```
/*--- Find the uddi-org:types classification scheme define by
the UDDI specification, using well-known key id.*/

String uuid_types = "uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4";
ClassificationScheme uddiOrgTypes =  (ClassificationScheme)
bqm.getRegistryObject(uuid_types, LifeCycleManager.CLASSIFICATION_SCHEME);

/*---Create a classification, specifying the scheme and the
taxonomy name and value defined for WSDL documents by the UDDI
specification.*/

Classification wsdlSpecClassification =
blcm.createClassification(uddiOrgTypes,"wsdlSpec", "wsdlSpec");

specConcept.addClassification(wsdlSpecClassification); <---------------

// Define classifications
Collection<Concept> concepts = new ArrayList<Concept>();
concepts.add(specConcept);

// Save Concept
BulkResponse concResponse = blcm.saveConcepts(concepts);

String conceptKeyId = null;
Collection concExceptions = concResponse.getExceptions();
```

Adding WSDL -3

```
//  Retrieve the (assigned ) Key from save concept

Key concKey = null;
if (concExceptions == null) {
  System.out.println("WSDL Specification Concept saved");
  Collection<Key> keys = concResponse.getCollection();
  Iterator<Key> keyIter = keys.iterator();
  if (keyIter.hasNext()) {
     concKey =  keyIter.next();
     conceptKeyId = concKey.getId();
     System.out.println("Concept key is " + conceptKeyId);
  }
}

// Retrieve the concept from Registry

Concept retSpecConcept =
(Concept)bqm.getRegistryObject( onceptKeyId,LifeCycleManager.
CONCEPT);
```

Adding tModel to ServiceBinding

```
// Associate  concept to Binding object

SpecificationLink retSpeclLink =
blcm.createSpecificationLink();

retSpeclLink.setSpecificationObject(retSpecConcept);

binding.addSpecificationLink(retSpecLink);
```

Registry Queries

1.findOrganizations

which returns a list of organizations that meet the query specified criteria

2.findServices

which returns a set of services offered by a specified organization

3. findServiceBindings

which returns the service bindings (information about how to access the service) that are supported by a specified service

Find Organization by Name -1

```
// Define find qualifiers and name patterns
Collection<String> findQualifiers = new ArrayList<String>();
findQualifiers.add(FindQualifier.SORT_BY_NAME_DESC);

Collection<String> namePatterns = new ArrayList<String>();
// qString refers to the organization name we are looking for
namePatterns.add("%" + qString + "%");

// Find orgs with names that matches qString
BulkResponse response = bqm.findOrganizations(findQualifiers,
namePatterns, null,null, null, null);
```

Find Organization by Name -2

```
//Iterate over discovered organizations and collect their
service binding
Collection orgs = response.getCollection();
Iterator orgIter = orgs.iterator();

   while (orgIter.hasNext()) {
     Organization org = (Organization) orgIter.next();
     Collection services = org.getServices();
     Iterator svcIter = services.iterator();
     while (svcIter.hasNext()) {
       Service svc = (Service) svcIter.next();
       Collection serviceBindings = svc.getServiceBindings();
       Iterator sbIter = serviceBindings.iterator();

        while (sbIter.hasNext())
              ServiceBinding sb = (ServiceBinding)
sbIter.next();
     }
   }
```

Find Organization by Classification -1

```
// NAICS classification(taxonomy) uuid
String uuid_naics ="uuid:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2";

ClassificationScheme cScheme = (ClassificationScheme)
bqm.getRegistryObject
(uuid_naics,LifeCycleManager.CLASSIFICATION_SCHEME)

/** we are looking for Simple Data Processing Services  with
514211 identifier in NAICS taxonomy
*/

InternationalString sn = blcm.createInternationalString("Simple
Data Processing Services");
String sv = "514211";

Classification classification =
blcm.createClassification(cScheme, sn, sv);
```

Find Organization by Classification -2

```
Collection<Classification> classifications = new
ArrayList<Classification>();

classifications.add(classification);

// Submit query to Registry
BulkResponse response = bqm.findOrganizations(null, null,
  classifications, null, null, null);

Collection orgs = response.getCollection();

// Display information about the organizations found
   Iterator orgIter = orgs.iterator(  );
   while (orgIter.hasNext() ) {
      Organization org = Organization) orgIter.next();
      System.out.println("Org.name:"+ org.getName().getValue());

System.out.println("Org. key id: " + org.getKey().getId());
    }
```

Part 2

JAXR API and Private Registry

JAXR
(Java API for XML Registries)

- JAXR provides uniforms and standard JAVA API to write client programs to access various kinds of XML registries (UDDI, ebXML).

- JAXR is part of JWSDP and also is coming with GlassFish.

Installing Private Registry on Tomcat -1

GlassFish does not come with UDDI registry service. We have to setup UDDI registry on **Tomcat**.

Steps to setup private registry in Tomact 5 :

1. Download Java WSDP 1.5 from the following URL:
   http://java.sun.com/webservices/downloads/1.5/index.html

2. Download Tomcat 5 bundle for Java WSDP 1.5 from the following URL:
   http://java.sun.com/webservices/containers/

3. Install Tomcat bundle .

4. Install a "Custom" Java WSDP 1.5 installation, specifying the installed Tomcat as the container and Registry Server as the only component. (Several other components will also be installed: JAXB, JAXP, JAXR, and SAAJ.)

5. Start Tomcat (*bin/catalina.sh start*)

Reference http://java.sun.com/webservices/docs/2.0/ReleaseNotes.html

Installing Private Registry on Tomcat -2

1. While installing JWSDP, If you get an exception like "... problem tail can not open +386 for reading..." enter following command:

export _POSIX2_VERSION=199209

2. Put the following "jars" into the classpath while compiling the "UDDI Registery" applications in JAVA:

/.../tomcat50-jwsdp/saaj/lib/saaj-api.jar"

.../tomcat50-jwsdp/saaj/lib/saaj-impl.jar

.../tomcat50-jwsdp/jaxr/lib/jaxr-api.jar

.../tomcat50-jwsdp/jaxr/lib/jaxr-impl.jar

.../tomcat50-jwsdp/jaxb/lib/jaxb-api.jar"

/.../tomcat50-jwsdp/jaxb/lib/jaxb-impl.jar

/.../tomcat50-jwsdp/jaxb/lib/jaxb-libs.jar

.../tomcat50-jwsdp/jaxb/lib/jaxb-xjc.jar

.../tomcat50-jwsdp/jwsdp-shared/lib/activation.jar

.../tomcat50-jwsdp/jwsdp-shared/lib/commons-beanutils.jar

Installing Private Registry on Tomcat -3

.../tomcat50-jwsdp/jwsdp-shared/lib/relaxngDatatype.jar
/.../tomcat50-jwsdp/jwsdp-shared/lib/xsdlib.jar
.../tomcat50-jwsdp/jwsdp-shared/lib/commons-logging.jar
/.../tomcat50-jwsdp/jwsdp-shared/lib/jaas.jar
.../tomcat50-jwsdp/jwsdp-shared/lib/jax-qname.jar
.../tomcat50-jwsdp/jwsdp-shared/lib/jta-spec1_0_1.jar
.../tomcat50-jwsdp/jwsdp-shared/lib/mail.jar
.../tomcat50-jwsdp/jwsdp-shared/lib/namespace.jar
.../tomcat50-jwsdp/jwsdp-shared/lib/PackageFormat.jar
/.../tomcat50-jwsdp/jwsdp-shared/lib/commons-collections.jar
/.../tomcat50-jwsdp/jwsdp-shared/lib/commons-digester.jar

Installing Private Registry on Tomcat -4

1. Run the sample java program in ".../tomcat50-jwsdp/registry-server/samples" to make sure that the Registry is working.

2. The program also shows how to add use/pswd to Registry . You will use the user/paswd to get access to registry and publish data.

3. You can find more samples on : .../jwsdp-1.5/jaxr/samples
"jwsdp-1.5" is the place you have already installed the WSDP 1.5

4. You can browse Registry content using "RegistryBrowser" tool placed in : "../tomcat50-jwsdp/jaxr/bin "

The access URI of the registry would be something like:
http://localhost:8080/RegistryServer/

Part 2

Tasks and Deliverables

Tasks - 1

1- Extend your previous code (in HW2) such that all web services are published in a UDDI registry.

Thus for creating a profile, first all services should be discovered, then invoked and finally the applicant profile is generated.

Tasks -2

2- Extend the previous scenario such that:

- Add Recruiting Company service, advertising jobs (via a web service) in several areas and looking for employee. Thus you need to create a Schema for advertised jobs (job name, description, location, company name, start date, salary..) and a Web service which gets job Search criteria (by field like IT, Deriver,...) and returns list of advertised jobs

Tasks -3

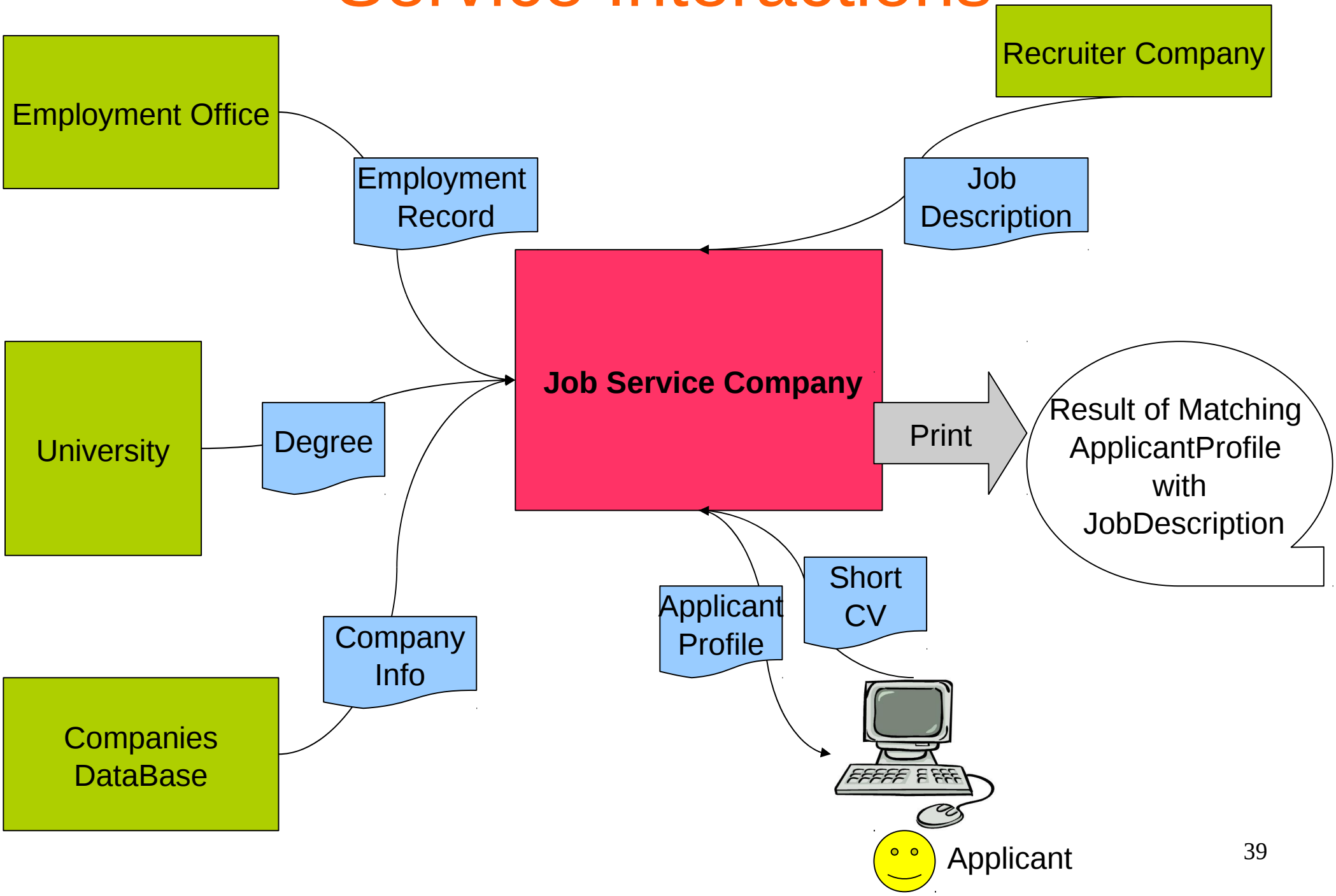
- Assume several applicants graduated from different universities in different field are looking for job through “Job Service Company”.
- They submit their short CV to “Job Service Company”.
- Thus a “Profile” fro each applicant should be created by invoking services from University, EmploymentOffice, DompanyDB.

Tasks -4

- The Job Service Company discovers the end-points (WSDLs) of those companies from Registry and then invokes discovered web service of the job advertising company, gets the job description, match them with applicant profiles and prints out the result of matching (successful or not).
- You can assume that the web service of Recruiting Company gets the some keywords (Java, programmer,...) as input and returns back the description of advertised jobs which match those input keywords as output.

3- Use both "*findByName*" and "*findByClassification*" methods and for finding the entities in registry.

Service Interactions



Tasks- 4

So in overall your system expected to do the following:

- As soon as it starts, it registers all developed services in Registry.

- Whenever a web service is needed to be invoked, (for the first time) it should be discovered from Registry . The subsequent calls (if any) can used the already discovered end-point ones.

-To simplify the process, assume that your system already knows how to invoke the discovered web services (you already have the code for web service client in your system).

- In order to match an ApplicantProfile with a Job Description, use any simple approach you like.

Deliverables

1- Source code of your system(previous system + above mentioned tasks), WSDLs and precise instructions on how to deploy and run your system.

Send it by Email with subject: PWS-HW3 to both :

shahabm@kth.se , nimad@kth.se.

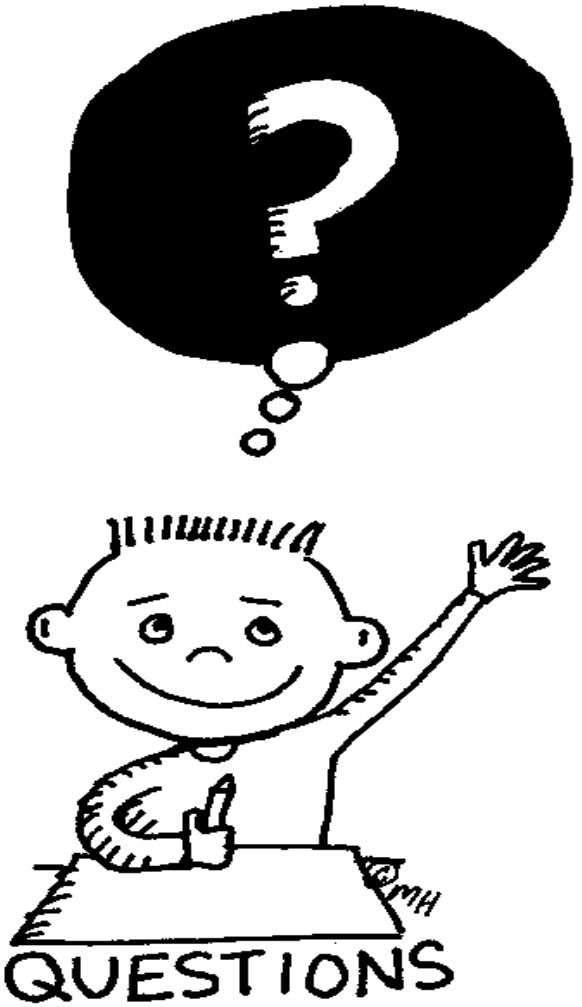
2- Present your working system .

Deadline: 28 Feb 2010

Presentation: 1 March 2010

GOOD LUCK!

Question?



References

1. Apache Tomcat JWSDP 1.5 bundle Link:

http://java.sun.com/webservices/containers/

2. The JavaServices Tutorial “For Java Web Services Developer’s Pack, v2.0”:

http://java.sun.com/webservices/downloads/previous/webservicespack.jsp

3. The JavaServices Tutorial “For Java Web Services Developer’s Pack, v1.6”:

http://java.sun.com/webservices/docs/1.6/tutorial/doc/index.html

Useful Links

Some JAXR Sample codes:

http://www.onjava.com/pub/a/onjava/excerpt/jws_6/index3.html

http://java.sun.com/developer/EJTechTips/2005/tt0322.html#2

http://www.javapassion.com/webservices/index.html#JAXR

NAICS classification:

http://www.census.gov/epcd/naics/naicscod.txt

You can find a simple UDDI browser at:

.../tomcat50-jwsdp/jaxr/bin/jaxr-browser.sh