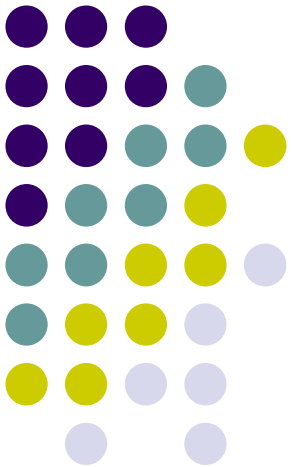
2G1523
Programming Web Services
Project

Course leader:
Professor Mihhail Matskin
misha@imit.kth.se

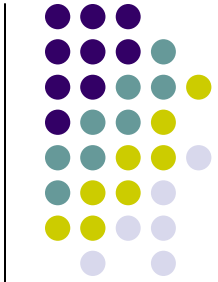
Teaching assistants:
Abdul Haseeb
ahaseeb@kth.se
Nima Dokoohaki
nimad@kth.se
KTH/ICT/ECS , VT08



Project - Deadline

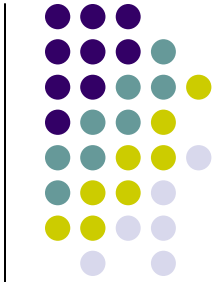
- Due date: **02-03-2008** (**very hard deadline**)
  - Demo: **03-03-2008, 12 – 3 pm**
- Deliverable: Report (See course web)

Project - Aims



- Aims:
  - To apply knowledge obtained during the course to design and implementation of web services
  - To learn how to control the transactional interaction among Web services.
  - To learn how to compose Web services.

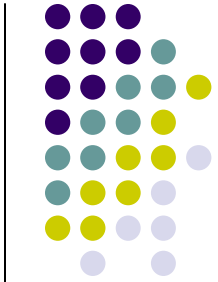
Project - Task



- Design and implement flight ticket reservation services (2 parts)

Or

- Your own project proposal *(**Approval deadline date : 22-2-2008**)*

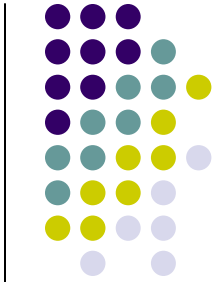


Project – Part 1

Implement the following services:

- Authorization of customers. Only the user with the correct username and password can have access to other services. This service should accept customer's Id and Password, check them with the list of registered customers and if found then give a reply that allows system access.

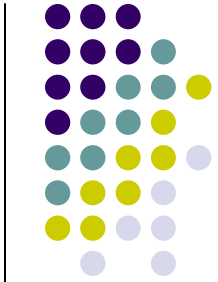
Project – Part 1



- Checking possible itinerary for flights given a departure city and a destination city. In case there is no direct flight from the departure to the destination, the service should find a route that combine several flights.

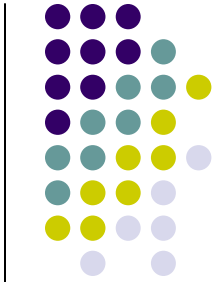
- Checking availability of tickets and finding their price for a given itinerary and given date.

Project – Part 1



- Output the price of available itineraries.
- Booking tickets for requested itinerary. Credit card number is required to book the tickets.
- Issue tickets. The ticket has to be booked before issuing.

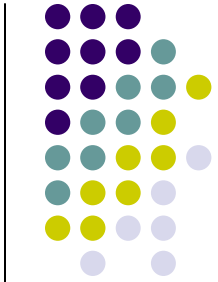
Project – Part 1



- The above-mentioned services must be described in WSDL, implemented in Java and deployed in the Axis.

- Construct both incoming and outgoing SOAP messages for invoking/replying all implemented services.

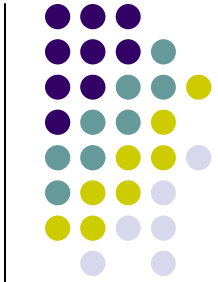
Project – Part 2



Implement the following functionalities:

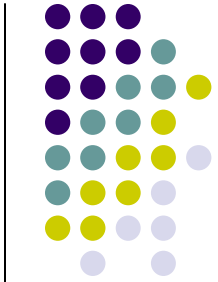
- Using the services implemented on the previous step implement the following composite service in BPEL:
  - The composite service takes the username, password, departure, destination and credit card number as input, and issues the cheapest ticket. If no ticket for the require route available, the service outputs an error message.

Project – Part 2



- You have to consider the transaction control in you reservation services. For example, checking ticket availability and booking ticket is a transaction. Use **WS-Coordination** and **WS-Transaction**.
- You have to make at least one of the above services to be state full (for example Checking availability service - however, you can choose any other service). Use **WS-Resources**.
- Construct both incoming and outgoing SOAP messages for invoking/replying all implemented services.

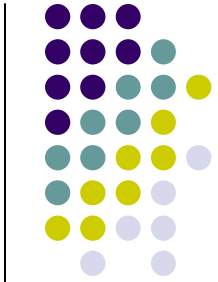
Project - Hints



- You implement functionality of all required services (***except for the composite service***) in Java.

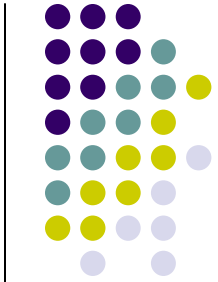
- You can use a database or simply a text file to store the flight tickets information.

Project – Report



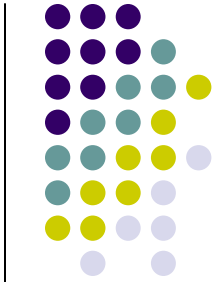
- Description of the problem and design solutions
- WSDL descriptions of all implemented services
- Java source code (or BPEL code) for all implemented services
- Protocols of services deployment

Project – Report



- Texts of all constructed incoming and outgoing SOAP messages for invoking/replying all implemented services.
- Explanation of your solution for implementing transactional control and state full resources in the project.
- Analysis of advantages and drawbacks of your solution, any remarks about implementing and usage of Web services.

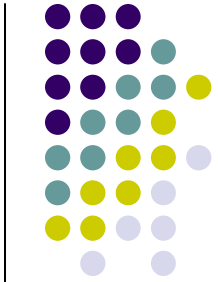
Project - Delivery



- Send your report by e-mail to both nimad@kth.se and ahaseeb@kth.se
- **Deadline: 02-03-2008**
- See course web for more information.

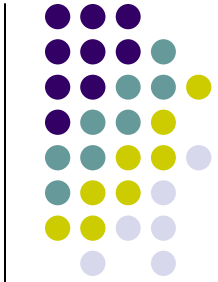
GOOD LUCK!

Industry Track



- A review on technologies involved in the project;
  - **BPEL**
  - **WS-TX**
    - **WS-Coor**
  - **WS-Resources**

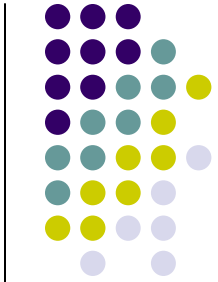
BPEL



- **Business Process Execution Language**

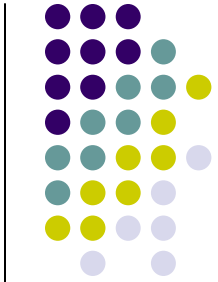
- BPEL is a language for specifying business process behavior based on Web Services.
  - Processes in WS-BPEL export and import functionality by using Web Service interfaces exclusively.
- Important: BPEL is an ***Orchestration*** language, not a *choreography* language;
  - Read about difference in your textbook

BPEL



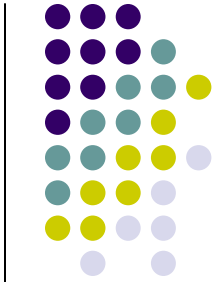
- Main released/adopted versions;
  - **BPEL4WS 1.1**
    - (Originially named **BPEL4WS** by vote from OASIS changed name into **WS-BPEL**.)
      - *“Business Process with BPEL4WS: Understanding BPEL4WS, Part 1”*
        - ***http://www.ibm.com/developerworks/library/ws-bpelcol1/***
  - **WS-BPEL 2.0**
    - *Specification:*
      - ***http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf***

BPEL



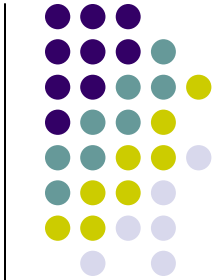
- New invariants:
  - **BPEL4People** specification
    - *WS-BPEL Extension for People*
    - WS-BPEL extension to address human interactions in WS-BPEL;
      - It defines a new type of basic activity which uses human tasks as an implementation, and allows specifying tasks local to a process or use tasks defined outside of the process definition.
      - This extension is based on the WS-HumanTask specification.
  - **WS-HumanTask** specification
    - Introduces the definition of human tasks and notifications, including their properties, behavior and a set of operations used to manipulate human tasks

BPEL 2

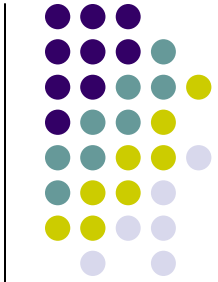


- **New in WS-BPEL 2.0:**
  - New activity types:
    - *repeatUntil, validate, forEach (parallel and sequential), rethrow, extensionActivity, compensateScope*
  - Renamed activities:
    - *switch/case* renamed to *if/else*, *terminate* renamed to *exit*
  - Termination Handler added to scope activities to provide explicit behavior for termination
  - Variable initialization
  - XSLT for variable transformations
  - XPath access to variable data XML schema variables in Web service activities
  - Locally declared *messageExchange* Clarification of Abstract Processes (syntax and most importantly **semantics**)
  - Enable expression language overrides at each activity

BPEL designtime support



- **Oracle BPEL designer**
  - Integrated with Oracle JDeveloper Studio Edition
    - *"Creating Your First BPEL Project using the BPEL Designer"*
    - *http://www.oracle.com/technology/obe/obe_as_1012/integration/bpel/jdev_sect/first_bpel_proj/1st_bpel_prj.htm*
- **Eclipse BPEL Project**
  - *http://www.eclipse.org/bpel/*
- **Netbeans Enterprise stack**
  - *"Developer Guide to BPEL Designer"*
    - *http://www.netbeans.org/kb/55/bpel_gsg.html*



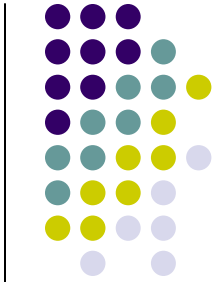
BPEL runtime support

- In order to run BPEL processes:

- **Apache ODE**

- **Oracle BPEL Manager**

Apache ODE

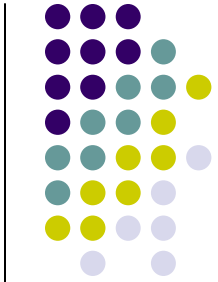


- **Apache ODE**
  - Orchestration Director Engine
  - Executes business processes written following the WS-BPEL standard.
  - ODE supports BPEL2
  - Latest stable version release: **1.1**
  - http://ode.apache.org/index.html

Oracle BPEL Process Manager

- **Oracle BPEL Process Manager**
  - Offers a comprehensive and easy-to-use infrastructure for *creating*, *deploying* and *managing* BPEL business processes.
  - *Oracle BPEL Process Manager for OC4J*
    - *http://www.oracle.com/technology/software/products/ias/bpel/index.html*

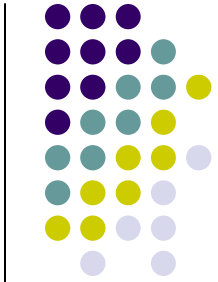
WS-Transactions



- **WS-Coordination**
  - Describes an extensible framework for providing protocols that coordinate the actions of distributed applications.
- **WS-AtomicTransaction**
  - Provides definition for atomic transaction coordination type
    - Three specific agreement coordination protocols for the atomic transaction coordination type: ***completion, volatile two-phase commit, and durable two-phase commit***.
- **WS-BusinessActivity**
  - Provides the definition of the business activity coordination
    - Two specific agreement coordination protocols for the business activity coordination type: ***BusinessAgreementWithParticipantCompletion, and BusinessAgreementWithCoordinatorCompletion***.
- *Robin Cover's technology report on "**Messaging and Transaction Coordination**"*
  - *http://xml.coverpages.org/coordination.html*

Apache Kandula

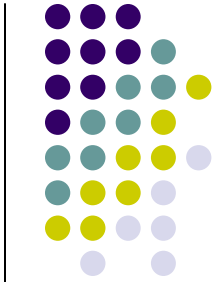
- Kandula **will** provide an open-source implementation of *WS-Coordination, WS-AtomicTransaction* and *WS-BusinessActivity* based on Axis.
  - *Under development!*
- The initial implementation will be in Java using Axis/Java.
- Kandula for Axis 1.x;
  - http://ws.apache.org/kandula/1/index.html
- *Howto for Atomic transactions using kandula:*
  - *http://ws.apache.org/kandula/1/ws-at--how_to_use.html*
- *Howto for business activity using kandula:*
  - *http://ws.apache.org/kandula/1/ws-ba--how_to_use.html*



WS-Resources

- **WS-ResourceFramework**
  - *“a web service is stateless nomially”*
  - WSRF provides a set of operations that web services may implement to become *stateful*
- **WS-BaseNotification**
  - defines the interface WS-Notification clients (consumers) and servers (producers) should expose
- **WS-DistributedManagement (WSDM)**
  - WSDM is a standard for managing and monitoring the status of other services.
- *Robin Cover’s technology report on ”**Stateful Web Services**“*
  http://xml.coverpages.org/statefulWebServices.html

Apache Muse



- A Java-based implementation of the :
  - *WS-ResourceFramework*
  - *WS-BaseNotification*
  - *WS-DistributedManagement*
  - A framework upon which users can build web service interfaces for manageable resources
    - Applications built with Muse can be deployed in *Apache Axis2*
  - http://ws.apache.org/muse/