# INFS 3204/7204
# Service-Oriented Architecture

A/Prof Heng Tao SHEN
ITEE, UQ
Semester 2, 2011

**M6: WS Composition**



## M6 Topics

- Motivations
- Concepts
- Design Principles
- Composition Models
  - Orchestration
  - Choreography
- A travel example



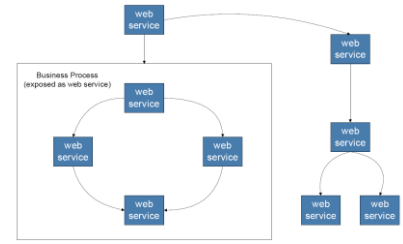
## Motivations

- Integration of intra-enterprise services
- Alliances with other enterprises
  - Offering value-added integrated services by combining existing web services
- Re-use and extension of existing services
- Increasing number of on-line services
- Just-in-time integration of existing services via internet
- Support for planning, definition and implementation of composite e-services



## WS Composition

- The ability of one business to provide value-added services through composition of basic Web Services, possibly offered by different companies
- A way to master **complexity**, via sequential, parallel, iterative and recursive manners



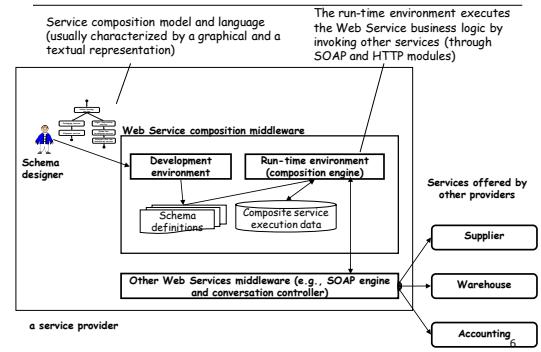



## WS Composition Middleware

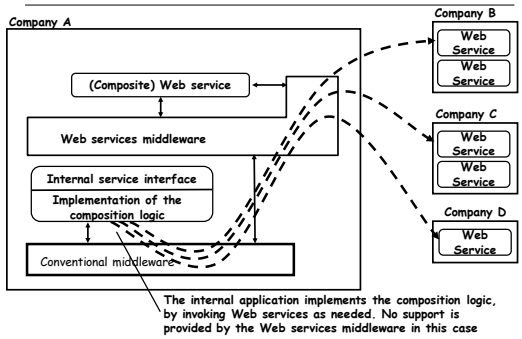
- Consists of abstraction and tools that facilitate the definition and execution of a composite service
- Allows developers to focus on the business logic rather than on low-level details.
- Includes:
  - A **composition model and language**
    - What services and how they are invocated via composition schema which defines the business logic
  - A development environment
    - Generally a graphical interface
  - A run-time environment
    - Composition engine that executes the business logic by invoking services defined in schema



## WS Composition Middleware

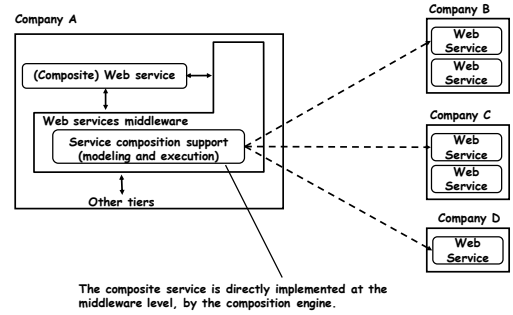


## WS Composition without Middleware






## WS Composition with Middleware






## Limitations of Conventional Composition

- Flexible and generic, but
  - Application-specific adapters
  - Ad hoc development
  - Price in complexity and cost
  - Lack of a standard composition model
    - Many of them, but never consolidate
    - A components in one system cannot be reused in others
    - E.g., WfMC, only a handful of vendors implemented, and standards are too generic



## Opportunities of WS composition

- Adequate components
  - Well-defined interface, well-described behaviours
- Standardized
  - Described by WSDL, invoked by SOAP,
- Lightweight, easy to use, rapid design and development
  - "Dream" of workflow management



## Design Principles of WS Composition

- Asynchronous service invocation
  - You do not want to hold everything up if you think the call to the method may take time
- Transactional integrity
  - Traditional ACID (atomicity, consistency, isolation and durability) are not sufficient for long-lived transactions
- Dynamic, flexible and adaptable to meet changing business needs
  - A clear separation between the process logic and Web services promotes flexibility
- Persistence and correlation
  - Maintain states for cross Web Services requests
- Exception handling



## WS Composition Models

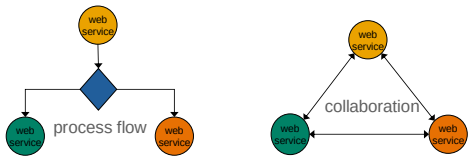
### Orchestration

- Refers to an **executable business process** that can interact with both internal and external Web Services
- Represents control from one party's perspective
- The interactions
  - Occur at the message level
  - Include business logic and task execution order
  - Can span applications and organizations to define a long-lived, transactional, multi-step process model

### Choreography

- Refers to **an abstract** Process
- Tracks the sequence of messages that may involve multiple parties including customers, suppliers and partners
- Shows the public message exchanges that occur between Web Services – rather than a specific business process that a single party executes

## Orchestration vs. Choreography



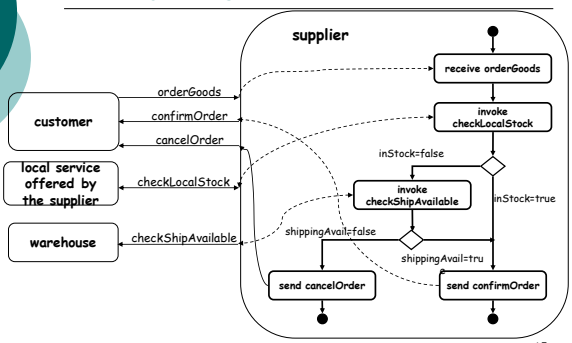

13

## Orchestration Models

- Categorized by business logic
  - Activity Diagram
  - State Chart
  - Petri Net
  - Activity Hierarchy
  - ...

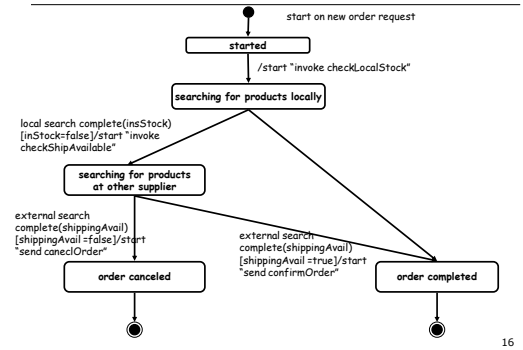
14

## Activity Diagram



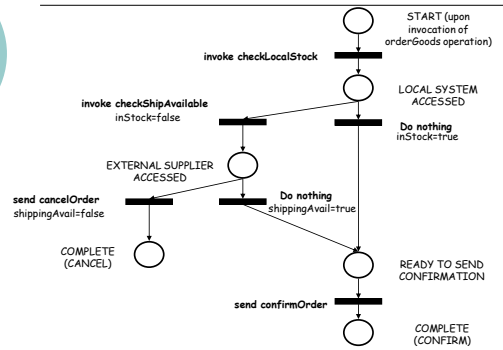

15

## State Chart



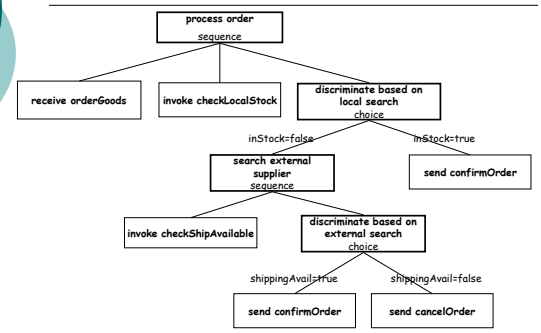

16

## Petri Net



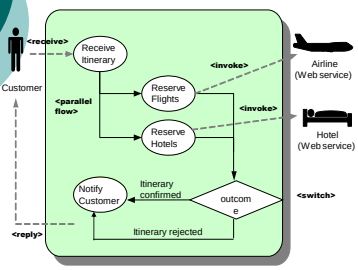

17

## Activity Hierarchy




18

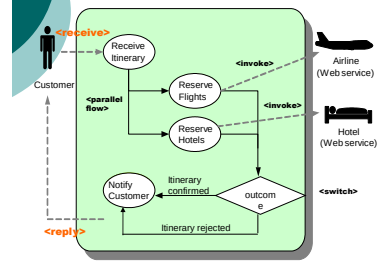
## Orchestration (Activity diagram): A travel agent example




- From the perspective of a travel agent
- Business logic
- Task execution order



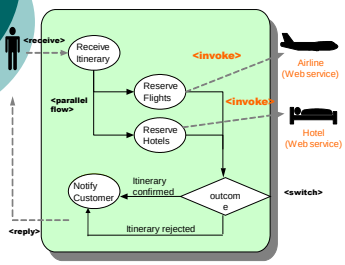
## <receive> and <reply>




- <receive> and <reply> activities receive messages from and give feedback to customers respectively
- The travel agent could deal with more than one client concurrently, so correlation identifiers are needed for different instances of a business process



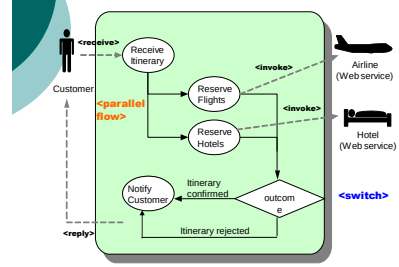
## <invoke>




- <invoke> activities are used to trigger internal and/or external web services, so that the flexibility for adapting the rapid business changes can be achieved
- Thanks to a clear separation between the process logic and Web services



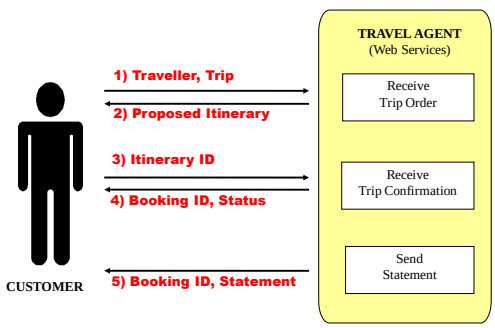
## <parallel flow> & <switch>




- <parallel flow> activity allows tasks to be executed concurrently
- <switch> activity allows conditional behaviours in business process



## Choreography: The travel agent example






## The Choreography example

The example should concern the following:

- Choreography
  - The sequence of the public message exchanges is regulated in Web Choreography while there are no restrictions on the order of triggering the operations of Web Services
- Transaction
  - A traveller will be informed that a trip request would be logically rolled back in case of any error happening during the process
- Possible Choices
  - The travel agent may be able to send either a Bill or a Notification that the trip cannot be confirmed based on the availability of the trip as communicated from the Airline Reservation system

## Workflow vs. WS Composition

| Workflow | Service composition |
|---|---|
| • Tasks | • Service types |
| • Resources (closed world) | • Service providers (open, dynamic) |
| • Resource rules | • Service selection rules |
| • Pull (worklists) | • Push (messages) |
| • Basic security (for pull) | • Certificates |
| • C++, Java, Corba, RMI | • HTTP, XML |
| • Static process | • Adaptive process |
| • "Standard" Black box | • Interface, Conversations |
| • No task transaction model | • Transactional models for e-services |

Fabio Casati, Mehmet Sayal, Ming-Chien Shan. "Developing E-Services for Composing E-Services", CAiSE 2001



## Composite Service Description Language (CSDL)

- Concepts developed by the workflow community have been extended by new features:
  - 2 levels: service flow and methods flow
  - Data mapping from input / output parameters into XML
  - Certificates – which certificate should be used
  - Service templates for compositions
  - Dynamic conversation – the ability of dynamically selecting the best available service from the repository of conversations (concept similar to service communities)

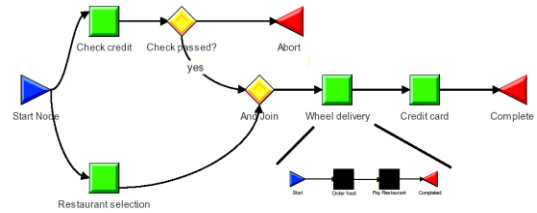


## CSDL

- Composite service as a process schema modelled by a graph
- Service nodes – invocations of basic or composite services
- Decision nodes – alternatives and execution flow control
- Event nodes – send and receive notifications
- Composite service may include the definition of input and output data (Java basic type or vectors, generic objects, **XML documents**)



## CSDL: An example






## Summary

- This week:
  - Web Service Composition
    - Motivations
    - Concepts
    - Design Principles
    - Composition Models
      - Orchestration
      - Choreography
    - A travel example
- Next week:
  - **SOA Development**



## References


- Fabio Casati, Mehmet Sayal, Ming-Chien Shan. "Developing E-Services for Composing E-Services", CAiSE 2001.
- Chris Paletz "Web Services Orchestration. A review of emerging technologies, tools and standards", Hewllett Packard White Paper, January 2003.