



# LSDIS

Large Scale Distributed Information Systems



University of Georgia  
Computer Science Department

Two of the  
very best  
TECH

# METEOR-S Process Design and Development Tool (PDDT)

Ranjit Mulye

LSDIS Lab, University of Georgia

(Under the Direction of Dr. John A. Miller)



# Acknowledgements

- Advisory Committee
  - Dr. John A. Miller (Major Advisor)
  - Dr. Maria Hybinette
  - Dr. Amit P. Sheth
- LSDIS/METEOR-S Student Members
  - Kunal, Karthik, Meenakshi, Ivan and others



# Topic Outline

- Introduction
  - Web Services and SOA
  - Web Processes
  - Web Service Composition
  - WSBPEL
  - Semantic Web Services
  - METEOR-S
- METEOR-S Process Design Tool
  - Why
  - Architecture
  - Features
- Related Work
- Conclusion
- Future Work



# Web Services and SOA

- Web Services
  - 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 in conjunction with other Web-related standards. (**W3C Definition**)
- Service Oriented Architecture
  - SOA is an architectural style whose goal is to achieve **loose coupling** among interacting **software agents**.



# Web Services and SOA (Contd.)

- Usability of Web Services
  - Allow **reuse** of Software Components
  - Allow integration of **heterogeneous** and **distributed** applications
- Next Step
  - Combine Web Services to create a **Web Process** that provides a complex functionality



# Web Processes

- Combining various Web Services to achieve a specific goal
- When to use
  - when the expected functionality cannot be combined in a single Web Service
  - When individual activities cross domain boundaries



# Web Process - Example

- Itinerary booking services
- Steps involved
  - Book air ticket
  - Depending on flight arrival, book car rental
  - Hotel reservation
- Each component offered by a different business
- Integrate the functionality by combining individual services to form a process



# Web Service Composition

- Modeling and execution of Web processes with individual **Web Services as its components**
- Reuse – Mix and match existing services to achieve desired functionality
- Composition approaches
  - Choreography
  - Orchestration



# WSBPEL

- WSBPEL defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners.
- Defines how multiple service interactions with the partners are coordinated to achieve a business goal, as well as the state and the logic necessary for this coordination
- Follows the process Orchestration methodology



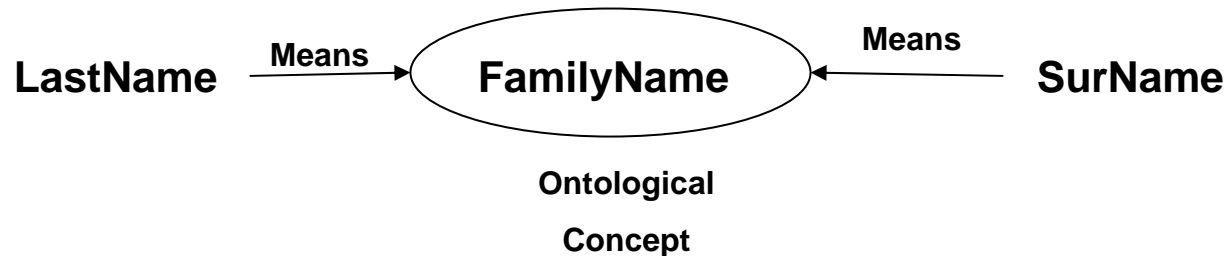
# Semantics

- Semantics focuses on meaning
- Helps to express relationship between word symbols and their intended meaning
- A Service represented in many ways may still have the same context – semantics captures this.



# Expressing Semantics

- Expressing Semantics using Ontologies
  - Ontology Serve as agreed vocabulary of terms and their intended meaning and give meaning to the relationship between such terms
  - They help in modeling a real world domain





# Semantic Web Services

- Describing Semantics of Web Services – METEOR-S Process Design Tool Perspective
  - What the service does, **functionally**
    - Mapping Operations to Ontological Concepts
    - Defining the **Pre-conditions** and **Results** of an operation
  - Describing **Inputs** and **Outputs** of the service
    - Mapping Inputs and Outputs Messages to Ontological Concepts

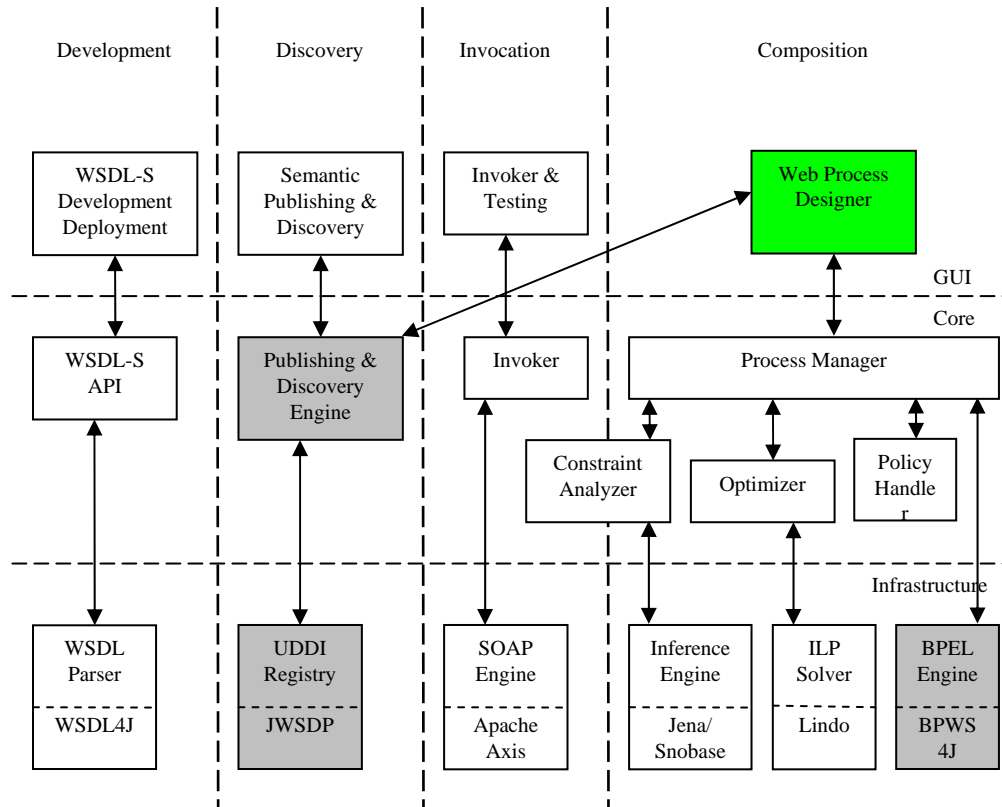


# METEOR-S

- Workflow management for Semantic Web Services is called METEOR-S (Follow on to METEOR)
- Uses semantics in the entire life cycle of Web Processes
  - Applying Semantics in Annotation, Quality of Service, Discovery, **Composition**, Execution
- Provides platform for dynamic composition, execution and discovery.



# METEOR-S Architecture





# METEOR-S Process Design Tool (PDT)

- Graphical Tool to help process developer build complex business processes.
- Developed as an Eclipse plug-in



# Why PDT

- WSBPEL specifications are difficult to learn and remember
- A process developer (domain expert) is conversant with the different BPEL constructs but not with their exact syntax
- Process developer should be able to specify process steps at a higher level



## Why PDT (Contd.)

- No freely available GUI based process design tool
- Part of METEOR-S to make it a complete workflow support system
- In line with METEOR-S ideology – exploiting use of Semantics in the design phase



# PDT Interface

Resource - BPEL Process File - Eclipse Platform

File Edit Navigate Search Project Run Window Help

BPEL Process File

- Select
- Marquee
- Definitions Elements
  - xmlns Namespace
  - partner Partner Link
  - variable Variable
  - correlationSet Correlation Sets
- Basic Activities
  - receive Receive
  - reply Reply
  - invoke Invoke
  - assign Assign
  - copy Copy
  - wait Wait
  - empty Empty
- Structured Activities
  - sequence Sequence
  - switch Switch
  - case Case
  - while While
  - flow Flow
  - link Link
  - pick Pick
  - onAlarm On Alarm
  - scope Scope
  - onMessage On Message
- Extension Elements
  - source Source
  - target Target
  - correlation Correlation
  - compensation Compensation Handler
  - faultHandler Fault Handlers
  - catch Catch
  - catchAll Catch All

sequence /sequence    Namespaces    Variables    Partners    Correlation Sets

Element Palette

Process Canvas

Element Property Sheet

Properties

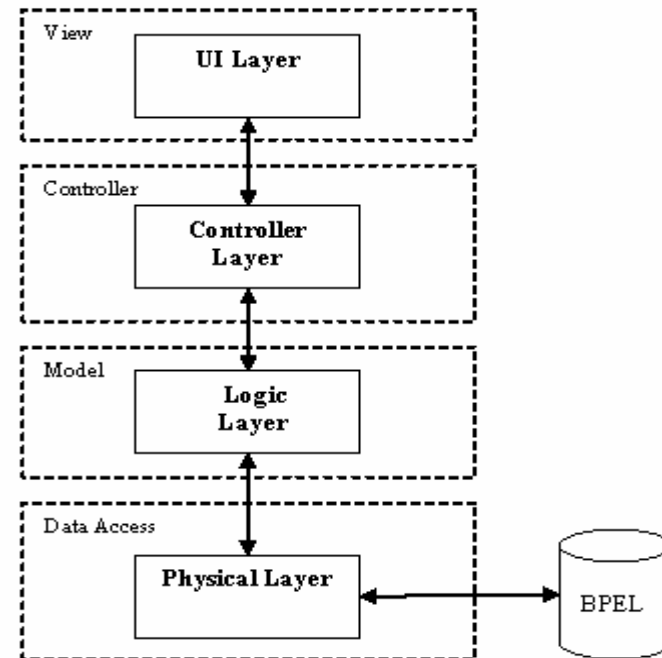
Property

- Process Name
- Target Name Space



# PDT – Architecture

- Follows Model-View-Controller (MVC) Pattern
- MVC Benefits
  - Decoupling of GUI and Logic/Model
  - Changes in model does not affect GUI
- Use of Graphical Editing Framework (GEF) for UI design





# PDT – Architecture (Contd.)

- View/UI Layer – Manage GUI generation and visual feedback to the process designer
- Model – Holds the main logic for BPEL process. In-memory model of the process being designed
- Controller – Keep UI in sync with the Model. Reflect the current state of the process in the UI. Manage user input
- Data Access – Generation of physical BPEL file and process WSDL. Reading and parsing of BPEL opened for modification



# PDT Architecture (Contd.)

- Benefits of using PDT Architecture
  - Adheres to widely used methodology for GUI design – MVC
  - Not tied to a single BPEL implementation. Replacing BPEL API easy.
  - Follows the BPEL model in constructs and properties



# PDT supported BPEL constructs

## **Basic Activities**

Invoke

Receive

Reply

Wait

Empty

Copy

Assign

## **Structured Activities**

Sequence

Switch

Case

While

Pick

Flow

Link

Scope

OnMessage/OnAlarm



# PDT supported BPEL constructs (Contd.)

## **Extensional group**

Source, Target

Correlation

Compensation Handler,  
Fault Handler

Catch, CatchAll

## **Definitions group**

Namespaces

Variable

Partner

Correlation Set



# PDT – Features

- BPEL engine expect Fully Qualified Namespaces. Typing them out can be error prone
- PDT eliminates this issue by restricting Namespace selection to ones that are already defined
- Similarly for Variables and Partners
- This helps generating an unambiguous BPEL process file

Property	Value	
<input type="checkbox"/> Misc		
Create Instance	No	
Name	Receive1	
Operation		
Partner Link		
Supress Join Failure	No	
Variable		
<input type="checkbox"/> Port Type		
PortType Local Part	processRequest	
PortType Namespace P...	processReply	
	serviceReply	
	serviceRequest	



# PDT – Features

- Easy to use drag-and-drop UI
- Error checking for invalid constructs
- Color coding of process elements depending on the type of functionality offered by it
- Easy to comprehend layout design
- Generation of an executable business process – Tested on BPWS4J and ActiveBPEL engines
- Support latest Oasis WSBPEL specification



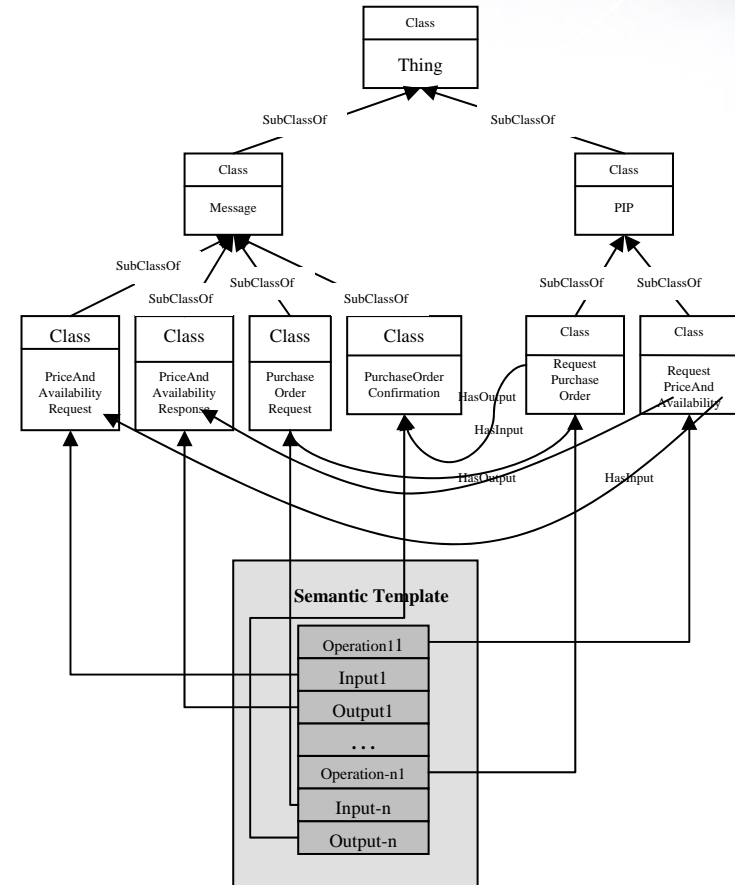
# PDT - Features

- Dynamic Discovery
  - Capability to choose partner services at design time rather than having them pre-selected before the design Process
- Support for Dynamic Discovery
  - Finalizing partners before design time may not be optimal
  - Newer better services could be available between the selection and process design phase
  - The pre-decided services may have gone down in the mean time
  - Not tied to a single partner



# Dynamic Discovery

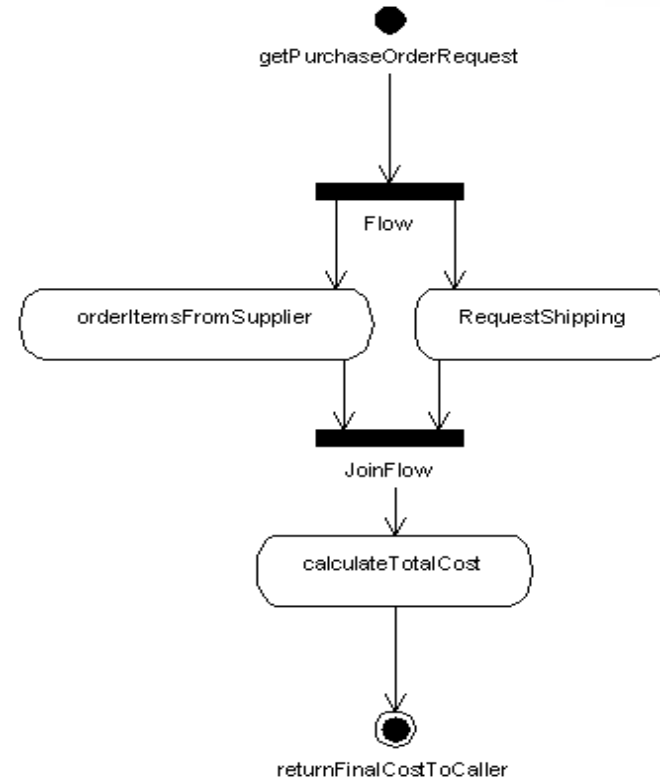
- Supply a template instead of partner
- Template consist of annotations
  - Operation
  - Inputs/Output
- Template in WSDL-S format





# Example Process

- Inventory ordering Process
  - Get request (quantity)
  - Invoke supplier service to order items
  - Invoke shipping service to request shipping of the ordered items
  - Calculate the total cost of the purchase order (item cost + shipping)





# Use Case - Snapshot

The screenshot displays the Eclipse IDE interface for editing a BPEL process file. The main workspace shows a process diagram with the following components:

- sequence** container:
  - receive** activity: purchaseOrder
  - flow** container:
    - invoke** activity: shipItems
    - invoke** activity: orderItems
  - /flow** end tag
  - assign** container:
    - copy** activity: adderInput = costOfItems
    - copy** activity: adderInput = shippingCharges
  - /assign** end tag
  - invoke** activity: add
  - reply** activity: purchaseOrder

Surrounding the diagram are several toolbars:

- Variables:** variable (qtyToOrder, costOfItems, shippingCharges, finalCostIncurred, adderInput)
- Namespaces:** xmlns
- Partners:** partner (caller, Supplier, Shipper, Adder)
- Correlation Sets:** (empty)

The left sidebar contains a palette of BPEL activities and elements, including: Select, Marquee, Definitions Elements, Namespace, Partner Link, Variable, Correlation Sets, Basic Activities (Receive, Reply, Invoke, Assign, Copy, Wait, Empty), Structured Activities (Sequence, Switch, Case, While, Flow, Link, Pick, On Alarm, Scope, On Message), and Extension Elements (Source, Target, Correlation, Compensation, FaultHandler, Catch, Catch All).

The right sidebar shows the Properties view for the selected element, listing properties such as Name, Operation, Output Variable, Partner Link, Suppress Join Failure, Port Type, PortType Local Part, and PortType Namespace Prefix.



# Related Work

- Commercially available Tools
  - Oracle Process Designer
  - WebSphere process Designer
- Fully automated process creation
  - SWORD – Using inputs/output description and rule-based system to auto-generate processes
  - Defining process goals and using state transition system to come up with interaction model
  - Planning techniques using AI approach – Using decision-theoretic planning using States and transition information



# Comparison of Modeling Approaches

Feature	PDT	Automated Composition (SWORD, OWL-S)	Commercial Tools (WebSphere, Collaxa)
Usability	+	+/-	+
Design Freedom	+	-	+
API Independence	+	+	-
Target Dependency	+/-	+	-
License	Open Source	?	Commercial



# Comparison of Process Design Tools

Feature	PDDT	WebSphere Oracle
Specification Support	Latest OASIS specification, WSBPEL 1.1	WSBPEL 1.1
Service Binding	Support for Late/Dynamic binding	Static Binding
Use of Semantics	Yes	No



# Conclusion

- Web Services, SOA and Web processes
- Modeling of Web processes
- Need for high level process creation tool
- This work provides an easy to use GUI based WSBPEL process design tool
- Offers a host of Usability features
- Generates executable BPEL processes (tried on freely available engines)
- Proposes use of dynamic discovery for process optimization



# Future Work

- Universal Description, Discovery and Integration (UDDI) registry Browser integration
- Parsing of partner WSDL and providing partner service elements wherever necessary
- Plugging into BPEL engine to give live feedback of current state of the process for monitoring
- Support for semantic template generation in addition to template selection



# Demo



# Questions





Thank you



# Why not Model using UML

- Not all of the workflow patterns offered by WSBPEL can be realized using UML constructs
- If we use UML Activity Diagram to model Web process, we may not be able to use the entire set of constructs offered by WSBPEL



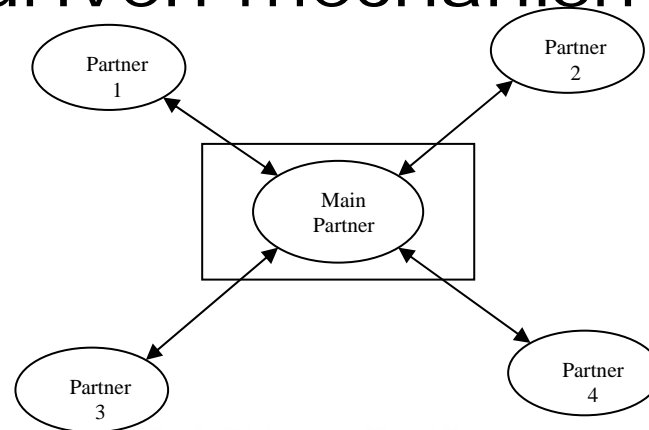
# Functional Ontologies

- Trying to define Verbs
- A Functional Concept Ontology and Its Application
- Functional Representation of Designs



# Web Service Composition - Orchestration

- Defines sequence and conditions in which one Web service invokes other services in order to achieve a specific goal
- Characterized by one central controller
- execution-driven mechanism





# Web Service Composition - Choreography

- Defines a model of sequence of operations, states, and conditions, to achieve a specific purpose or goal.
- Characterized by decentralized control
- more abstract and descriptive

