What is the class about?

- **Course description and syllabus:**
  - http://www.nyu.edu/classes/jcf/g22.2440-001/
  - http://www.cs.nyu.edu/courses/spring14/G22.2440-001/

- **Textbooks:**
  - *Software Engineering: A Practitioner’s Approach*
    Roger S. Pressman
    McGraw-Hill Higher International
    - http://highered.mcgraw-hill.com/sites/0073375977/information_center_view0/
Icons / Metaphors

- Information
- Common Realization
- Knowledge/Competency Pattern
- Governance
- Alignment
- Solution Approach
See Session 7 Handouts:

» Handout 30 – OMG’s BPMN Resource Page
» Handout 31 – BPMN 1.0 Specification
» Handout 32 – BPMN Patterns
» Handout 33 – Business/Application Arch. Engineering
» Handout 34 – Sample Business Architecture
» Handout 35 – Sample BPMS (Lombardi TeamWorks)
Agenda

1. Introduction
2. Business Model Representation
3. Business Process Modeling
4. Capturing the Organization and Location Aspects
5. Developing a Process Model
6. BPMN, BPML, and BPEL4WS
7. Business Process Interoperability
8. Summary and Conclusion
Section Outline

- Business Process Modeling background
- Business Process Modeling Phases
Discussion

- Has anyone built something that missed the mark?
  - It failed to meet the business need
  - End-users did not use it

- How do you know that what you are doing is the right thing to do?

- Who do you build your interfaces for?

- How can you ensure that you are “doing the right thing”?
What is BPM?

- A standardized approach to **analyze**, **streamline** and **integrate** business processes **independent of organizational boundaries**, to provide maximum operating efficiencies via clearly defined business and IS projects.

- A **business process** focused methodology
  - Not Business Process Reengineering
  - Led by the business
  - Facilitated by IS
Why BPM?

- Identify and document business process improvement opportunities
  - To gain operating efficiencies through analysis of current and future business processes

- Align business and IS
  - To promote integration of business processes and technologies by identifying strategic initiatives

- Identify and reuse processes
  - To identify core business processes
  - To share and reuse business processes across organizational boundaries
BPM Objectives

- **Analyze** business processes independent from organization
- **Quantify** the benefits and savings for each proposed initiative
- Identify opportunities for process elimination and/or reassignment
- Identify opportunities for business process improvement through **automation** and optimization
Risks without BPM

- Systems could be implemented as a “forced fit”
- Potential loss of business and technology alignment
- Increased manual processes and “workarounds”
- Building unnecessary user interfaces
- Not identifying the best possible solutions
The Value of BPM

- Business area and IS develop a common understanding of business requirements
  - Simple modeling notation
  - Consistent, repeatable approach

- **Short**, focused BPM efforts
  - Swift identification of process improvement initiatives

- Promotes seamless integration and reuse of **processes, data and technologies**

- Captures current business processes that can be used as a knowledge-base for training
BPM Methodology Phases

- Business Process Modeling (BPM)
  - Pre-Project Assessment
  - Project Planning
  - Business Needs Analysis
  - Current Process Understanding
  - New Process Design
  - Transformation
- Develop project plan
  - High-level scope definition
- Build project team
- Conduct project kick-off meeting
  - Core team
  - Stakeholders
- Initiate project management activities
- Training of Core team and subject matter experts
Define BPM detailed scope

- Define project purpose, goals and objectives
- Identify problems to be addressed
- Determine key measures
Current Process Understanding

- Develop business process models
- Identify business process information needs
- Capture current process metrics and characteristics
- Identify areas of opportunity for improvement
Develop Global Business Function Model
  » Based on annotated CPU models from Optimization
  » Incorporate process improvement opportunities
• Document recommended business initiatives
• Produce list of recommended projects
• Identify sequence for recommended IS projects
Sample Business Process Map (Trading)

Client

1. **Client logs in**
2. **Personalized "Web Page**
3. **Submits Order**
4. **"Not a valid order"**

Front/Back Office

- **Validate Order**
- **Record Order**
- **Is it Domestic?**
- **Trigger FX**
- **Electronic Trading Available?**
- **Manual Review Needed**
- **Route to Trading Desk**
- **Electronic Execution**
- **Monitor Execution**
- **Update customer account**
- **Confirm Execution (price, qty..)**
- **E-mail Confirmation**
- **Monitor Electronic execution**
- **Non Automated Execution**

OTC Market

- **Sales Desk Contacts Client**
- **Contacts Client**
- **Record Order**
- **Update customer account**
- **Monitor Electronic execution**
- **Non Automated Execution**

"Real-time alert"

*Client Calls in Order*

Is this a product in your inventory or available on ECN.

No

Yes

Route to Trading Desk

Electronic Submission execution

"Electronic pass through"
Understanding Business Model Engineering

- Data conversion
- Supplied volume data
- Projects creation and update
- Projects approval
- KPI creation
- Project information retrieval
- System administration
- Implementation Team
- Training Team
- Data
- Business Process
- Application
- Technology
- Location
- Hardware
- Standard system software
- Reporting software
- Ad-hoc spreadsheet functions
- Security and performance
- Site navigation design
- Site content design
- Reusable components
- Security workflow
- Help
- Office hosting development
See:

http://xml.coverpages.org/ni2003-08-29-a.html
What is a Business Process?

A business process defines how an organization achieves its purpose.

The process represents the flow of work and information throughout the business.

This involves:
- People
- Machines
- Materials
- Products
- Roles
- Documents
Models of Business Processes

UML became the standard modeling language for business processes. Of the 9 UML diagram types, 3 are particularly important:

- **use cases**,  
- **activity diagrams** and  
- **scenarios** (sequence or collaboration diagrams).

**Use cases** show informally how the business process interacts with outside actors (customers, stakeholders, suppliers etc.)

The process dynamics is modeled by **activity diagrams**.

Often only **specific scenarios** are of interest. E.g. service monitoring, system reconfiguration, performance degradation etc.
A Use Case

Room Reservation

WEB

Reservation System

<<includes>>

Room Service

<<extends>>

Hotel Desk

<<extends>>

<<extends>>

John

Markus
Activity diagrams constitute the core of a business model.

**Activity Diagram**: state transition diagram with concurrent states (Statechart)

**Activity**: state with internal action

**BPM**: activity = Mealy state
BPM
Activity Diagram

Illustrates flow of work done by different actors; events cause them to interact.
A UML Business Process Model: Different Views

Class Diagram

Statechart

Activity Diagram

Use Case
Quantitative Evaluation of BPMs

UML is a semi-formal language: there is little formal semantics attached to the individual diagrams.

In order to evaluate a visual UML-model we have, therefore, to enrich it with additional information and then transform it to a model with precise semantics.

This mapping supplies additional semantics to the source language.

\[
\text{UML- Model} \quad \Rightarrow \quad \text{Analytical Model}
\]
Activity Diagram Applications

- Intended for applications that need control flow or object/data flow models …
- … rather than event-driven models like state machines.
- For example: business process modeling and workflow.
- The difference in the three models is how a step in a process is initiated, especially with respect to how the step gets its inputs.
Each step is taken when the previous one finishes
...regardless of whether inputs are available, accurate, or complete.
Emphasis is on order in which steps are taken.
Each step is taken when all the required input objects/data are available …
… and only when all the inputs are available.
Emphasis is on objects flowing between steps.
- Each step is taken when events are detected by the machine …
- … using inputs given by the event.
- Emphasis is on reacting to environment.

Not UML Notation
Kinds of Steps in Activity Diagrams

- **Action (State)**
- **Subactivity (State)**

- Just like their state machine counterparts (simple state and submachine state) except that ...
- ... transitions coming out of them are taken *when the step is finished*, rather than being triggered by a external event, ...
- ... and they support dynamic concurrency.
An action is used for anything that does not directly start another activity graph, like invoking an operation on an object, or running a user-specified action.

However, an action can invoke an operation that has another activity graph as a method (possible polymorphism).
A subactivity (state) starts another activity graph without using an operation.

Used for functional decomposition, non-polymorphic applications, like many workflow systems.

The invoked activity graph can be used by many subactivity states.
POEmployee.sortMail

Deliver Mail

POEmployee

| sortMail() |

Deliver Mail

Check Out Truck

Put Mail In Boxes
THE model (application) is completely OO when all action states invoke operations, and all activity graphs are methods for operations.
Invokes an action or subactivity any number of times in parallel, as determined by an expression evaluated at runtime..

Upper right-hand corner shows a multiplicity restricting the number of parallel invocations.

Outgoing transition triggered when all invocations are done.

Currently no standard notation for concurrency expression or how arguments are accessed by actions. Attach a note as workaround for expression.
A special sort of step (state) that represents the availability of a particular kind of object, perhaps in a particular state.

No action or subactivity is invoked and control passes immediately to the next step (state).

Places constraints on input and output parameters of steps before and after it.
- Take Order must have an output parameter giving an order, or one of its subtypes.

- Fill Order must have an input parameter taking an order, or one of its supertypes.

- Dashed lines used with object flow have the same semantics as any other state transition.
Coordinating Steps

- Inherited from state machines
- Initial state
- Final state
- Fork and join
- **Decision point and merge** are inherited from state machines.

- For modeling conventional flow chart decisions.
- **Synch state** (○ ) is inherited from state machines but used mostly in activity graphs.
- Provides communication capability between parallel processes.
- Forks and joins do not require composite states.
- Synch states may be omitted for the common case (unlimited bound and one incoming and outgoing transition).
- Object flow states can be synch states
- Fork transitions can have guards.

- Instead of doing this:
Convenience Features (2/3)

- **Partitions** are a grouping mechanism.
- **Swimlanes** are the notation for partitions.
- They do not provide domain-specific semantics.
- Tools can generate swimlane presentation from domain-specific information without partitions.
- Signal send icon
  … translates to a transition with a send action.
- Signal receipt icon
  … translates to a wait state (an state with no action and a signal trigger event).
Case Study

Submission Team

- Develop technology specification
- Submit specification draft
- Collaborate with competitive submitters
- Evaluate initial submissions
- Finalize specification

Issue RFP

Task Force

- RFP [issued]
- Specification [initial proposal]
- Specify initial proposal
- Specify final proposal

Revision Task Force

- Initial state
- Fork of control
- Join of control
- Action state
- Control flow
- Object flow
- Input value
Collaborate with competitive submitters

Evaluate initial submissions

Evaluate final submissions

Vote to recommend

[YES] Specification [final proposal]

[NO]

Finalize specification

Implement specification

Specification [adopted]

Enhance specification

Specification [revised]

Revise specification

Recommend revision

Specification [final proposal]

Recommend revision

[Enhanced]

[else]

final state
When to Use Activity Diagrams

- Use activity diagrams when the behavior you are modeling ...
  - does not depend much on external events.
  - mostly has steps that run to completion, rather than being interrupted by events.
  - requires object/data flow between steps.
  - is being constructed at a stage when you are more concerned with which activities happen, rather than which objects are responsible for them (except partitions possibly).
Control flow and object flow are not separate. Both are modeled with state transitions.

Dashed object flow lines are also control flow.

You can mix state machine and control/object flow constructs on the same diagram (though you probably do not want to).
Activity Diagram Modeling Tips (2/6)

From UML User Guide:

- **Customer**
  - Request Return
  - Ship Item
    - Item [returned]

- **Telesales**
  - Get Return Number

- **Accounting**
  - Credit Account
    - Item [available]

- **Warehouse**
  - Receive Item
  - Restock Item
Activity diagrams inherit from state machines the requirement for well-structured nesting of composite states.

This means you should either model as if composite states were there by matching all forks/decisions with a correspond join/merges ...

... or check that the diagram can be translated to one that is well-nested.

This insures that diagram is executable under state machine semantics.
Well-nested:
Not well-nested:

Apply structured coding principles. (Be careful with goto’s!)
Can be translated to well-nested diagram on earlier slide:
1. Introduction
2. Business Model Representation
3. Business Process Modeling
4. Capturing the Organization and Location Aspects
5. Developing a Process Model
6. BPMN, BPML, and BPEL4WS
7. Business Process Interoperability
8. Summary and Conclusion
See:

- http://www.bpmi.org
- http://www.processwave.net/Articles/SoftwareProcess/BusinessModelingArticles.htm
- http://cde.berkeley.edu/resources/bpmn/
- http://www.popkin.com/
- http://www.popkin.com/customers/customer_service_center/demos/demos_with_overview_process_map.htm
## Procedural Knowledge Road Map

<table>
<thead>
<tr>
<th>Type of knowledge/Levels</th>
<th>Conceptual model</th>
<th>Deployment methodology</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Instance level</td>
<td>Process instance concept</td>
<td>BPMN+</td>
<td></td>
</tr>
<tr>
<td>Process definition level</td>
<td>Process definition concept WfMC meta-model+</td>
<td></td>
<td>XPDL+</td>
</tr>
</tbody>
</table>

- **BPMN** - Business Process Modeling Notation, BPMI, 11.02
- **XPDL** - XML Process Definition Language, WfMC, 07.02
- **Wf-XML** - Workflow Interoperability Binding, WfMC, 11.01
ICONS’ conceptual models

- Process definition concepts - WfMC’s workflow meta-model extended with
  - time modeling
  - flexible Workflow Participant Assignment [Momotko2002]

- Process instance concepts
  - advanced process instance and activity instance behavioral models (timing and criticality behavior)
  - advanced time management
BPMN and XPDL extensions

- **BPMN extension (BPMN+)**
  - idea - standard well-known notation for both process definition adjusted to the needs for process instance
  - ICONS’ approach - a BPMN extension to visualise process execution

- **XPDL extension (XPDL+)**
  - performer relationships (WPA)
  - pre & post conditions
Possible Process Execution

Process of a vocation application

Stared: 20-12-2002 9:45  Finished:  
Current Delay: 2 mi

Applicant (Mary_J)

Verifier (System)

Supervisors <all> (1- John_B)

Director

Registration Body

Performance: Peter_H
Start Date: 20-12-2002 9:54
Finish Date:
Working Time:
Waiting time: 30/12 in 4
Current Status: Waiting

Data for record:
Application from: "App45" (Read)
Application executed:
Application: DB_Vac('App45', 'Mary_J')
Semantic Web & Ontology

- **Semantic Web**
  - Machine-processible semantics of data
    - Semantic annotation, XML, RDF
  - Explicit representation of the semantics of data accompanied with domain theories (i.e. ontologies)
  - Lead to a highly knowledgeable world-wide system

- **Ontology**
  - “specifications of a shared conceptualization of a particular domain”
  - Describe the semantics of information exchange
  - Ontology description tools: ontolingua, OIL, DAML
Example: ECIMF methodology
Example: Relationship between the ECIML and other modeling standards.
Semantic Translation Layer

- Mapping concepts from different ontologies
- Semantic Translation meta-model
- Example scenario that requires Process Mediator.
- Business Context model as seen by the shipping agency
Business Context Matching

- Business Context model as seen by the customer
## Process Mediation (1/2)

<table>
<thead>
<tr>
<th>Party</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collaboration Task</strong></td>
<td>Payment Collaboration</td>
</tr>
<tr>
<td><strong>Framework</strong></td>
<td>RNIF 2.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transaction name</th>
<th>Initiator / Responder</th>
<th>Request document</th>
<th>Response document</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIP3A1: Request for quote</td>
<td>Initiator</td>
<td>QuoteRequest</td>
<td>QuoteConfirm</td>
</tr>
<tr>
<td>PIP3A4: Request Purchase Order</td>
<td>Initiator</td>
<td>PORrequest</td>
<td>POConfirm</td>
</tr>
<tr>
<td>PIP3C3: Notify of Invoice</td>
<td>Responder</td>
<td>Invoice</td>
<td></td>
</tr>
<tr>
<td>PIP3C6: Notify of remittance advice</td>
<td>Initiator</td>
<td>RemittanceAdvice</td>
<td></td>
</tr>
</tbody>
</table>

| Message delivery control         | Any                   | Secure Flow      |                   |

<table>
<thead>
<tr>
<th>Party</th>
<th>ShippingAgency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collaboration Task</strong></td>
<td>Payment Collaboration</td>
</tr>
<tr>
<td><strong>Framework</strong></td>
<td>EDIFACT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transaction name</th>
<th>Initiator / Responder</th>
<th>Request document</th>
<th>Response document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request for quote</td>
<td>Responder</td>
<td>REQUOTE</td>
<td>QUOTES</td>
</tr>
<tr>
<td>PIP3A4: Request Purchase Order</td>
<td>Responder</td>
<td>ORDERS</td>
<td>ORDRSP</td>
</tr>
<tr>
<td>Notify of Invoice</td>
<td>Initiator</td>
<td>INVOIC</td>
<td></td>
</tr>
<tr>
<td>Notify of remittance advice</td>
<td>Responder</td>
<td>REMADV</td>
<td></td>
</tr>
<tr>
<td>Message delivery control</td>
<td>Any</td>
<td></td>
<td>APERAK, CONTRL</td>
</tr>
</tbody>
</table>
Process Mediation (2/2)
### Semantics of the two corresponding concepts

<table>
<thead>
<tr>
<th>Customer: TV-set</th>
<th>Semantic Translation</th>
<th>Shipping Agency: Box</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Properties</strong></td>
<td><strong>Mapping Rules</strong></td>
<td><strong>Properties</strong></td>
</tr>
<tr>
<td>Height</td>
<td>Tv_set → Box: dimension values will always be higher, but discrete. Need to be obtained from a cardboard box catalogue (external resource)</td>
<td>Height Width Depth Represent the physical dimensions of the cardboard box used to ship the electronic equipment of any kind. The values are discrete, because only certain box sizes are available.</td>
</tr>
<tr>
<td>Width</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Depth</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Represent the physical dimensions of the TV set chassis.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Not available (N/A)</strong></td>
<td>Tv_set → Box: needs to be obtained from a product catalogue (external resource)</td>
<td>Weight Represents the weight of the box with the contents.</td>
</tr>
<tr>
<td><strong>Box → Tv_set: not needed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N/A</strong></td>
<td>Tv_set → Box: needs to be obtained from a product catalogue (external resource)</td>
<td>StackingLevels Represents the number of levels the boxes can be stacked, one on top of the other.</td>
</tr>
<tr>
<td><strong>Box → Tv_set: not needed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N/A</strong></td>
<td>Tv_set → Box: always set to True.</td>
<td>Fragile Marks the payload as fragile (requiring special care during transportation)</td>
</tr>
<tr>
<td><strong>Box → Tv_set: not needed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Tv_set → Box: not needed</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Box → Tv_set: needs to be obtained from a product catalogue (external resource)</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Analyze the message delivery control mechanisms

<table>
<thead>
<tr>
<th>Customer (RNIF)</th>
<th>Semantic Translation</th>
<th>Shipping Agency (EDI)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SecureFlow</strong></td>
<td>The RNIF business documents map 1:1 to EDI business messages, e.g.: QuoteRequest ↔ REQUOTE QuoteConfirm ↔ QUOTES PORestore ↔ ORDERS POConfirm ↔ ORDRSP etc... However, individual data elements can be missing, and will have to be collected from the previous messages, or supplied explicitly in the rules, or obtained from external resources.</td>
<td><strong>APERAK</strong>&lt;br&gt;<strong>REQUOTE</strong>&lt;br&gt;<strong>QUOTES</strong>&lt;br&gt;<strong>ORDERS</strong>&lt;br&gt;<strong>ORDRSP</strong>&lt;br&gt;<strong>INVOIC</strong>&lt;br&gt;<strong>REMADV</strong></td>
</tr>
<tr>
<td><strong>Document</strong></td>
<td><strong>RcptAck</strong>&lt;br&gt;<strong>Exception</strong>&lt;br&gt;<strong>RcptAckExc</strong>&lt;br&gt;<strong>GeneralExc</strong></td>
<td>In this particular case, the EDI system uses APERAK and CONTRL messages only to signal exceptions. Acknowledgements are implicit, in the form of response business documents.</td>
</tr>
<tr>
<td><strong>Signal</strong></td>
<td></td>
<td><strong>CONTRL</strong>&lt;br&gt;<strong>ORDERS</strong>&lt;br&gt;<strong>ORDRSP</strong>&lt;br&gt;<strong>INVOIC</strong>&lt;br&gt;<strong>REMADV</strong></td>
</tr>
<tr>
<td><strong>ReceiptAck</strong></td>
<td>RNIF → EDI: not needed – don’t forward.</td>
<td>N/A – implementation choice (positive acknowledgements are implicit).</td>
</tr>
<tr>
<td><strong>ReceiptAckException</strong></td>
<td>EDI → RNIF: needs to be synthesized from the response document. Possible problems with timing constraints… (ack. too late)</td>
<td><strong>CONTRL</strong>&lt;br&gt;This message is sent when parsing errors occur. Business data was not considered at all.</td>
</tr>
</tbody>
</table>

- **SecureFlow** consists of a business document (containing business data), and a responding business signal (acknowledgement).

- **ReceiptAck**<br>This signal means that the document business data has been accepted for further processing (which implies also well-formedness).

- **ReceiptAckException**<br>This signal means the document was not well-formed (parsing errors). Business data was not considered at all.
- Message syntax mapping
Shared Ontology Approach to Semantic Translation

Multiple ontologies + labels

- Shared ontology
  - local ontology
  - local ontology
  - local ontology
EDOC & ebXML: Vision

- **EDOC**
  - Simplify the development of component based EDOC systems by means of a modeling framework, based on UML 1.4 and conforming to the OMG Model Driven Architecture.
  - Provide a platform independent, recursive collaboration based modeling approach that can be used at different levels of granularity and different degrees of coupling, for both business and systems modeling.
  - Embrace MDA – Provide design and infrastructure models and mapping

- **ebXML**
  - Creating a single global electronic market
The Internet Computing Model

- Collaboration of independent entities
- Document exchange over internet technologies
  - Large grain interactions, not “method calls”
- No required infrastructure *
- Long lived business processes
- Business transactions
  - Business Party
  - Business Party
  - Portals
Requirements for the “ICM”

- Contract of Collaboration
  - Meta-Model (EDOC-ECA) and representation (I.E. XMI, ebXML-BPSS)
  - Shared Repository for Contracts (MOF, UDDI, ebXML)
  - Tightly coupled systems may simulate the repository with file exchange (I.E. IDL)

- Connectivity which meets requirements of the contract

- Implementation of each contract role providing connectivity (application server)

Contract of collaboration can be mapped to the format of various technologies. (ebXML, Soap, .NET)
Two Levels of Interoperability

Instance data and interoperability

Business Partner

EbXML
Over Soap

Bridge

Biztalk
Over Soap

Business Partner

Metadata (model) interoperability

EbXML

EDOC
ECA

Normal Form

Biztalk

Each can be transformed
Inside one role you frequently find more Collaborating “parts” of the enterprise Until you get to a role within a domain
   » These can share resources!
   » E.G. Common access to a DBMS or Service
   » Exist within a managed domain
   » Can also be a legacy application

ebXML does not go here, Only EDOC-ECA
### Standards for collaboration

<table>
<thead>
<tr>
<th></th>
<th>EDOC-ECA</th>
<th>ebXML-BPSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Collaborations</td>
<td>Yes – Community Process</td>
<td>Yes – Multi Party Collaboration</td>
</tr>
<tr>
<td>Contract of Interaction</td>
<td>Yes – Protocol with Choreography &amp; Object Interface</td>
<td>Yes – Binary Collaboration with Choreography and Business Transactions</td>
</tr>
<tr>
<td>Content Model</td>
<td>Yes – Document Model</td>
<td>Uses external forms, such as XML Schema</td>
</tr>
<tr>
<td>Recursive Composition</td>
<td>Yes – Recursive Composition into Enterprise</td>
<td>No – Only “B2B”</td>
</tr>
<tr>
<td>Detail sufficient to drive communications</td>
<td>No – Requires technology mapping</td>
<td>Yes – As ebXML transport. BPSS includes timing and security parameters.</td>
</tr>
<tr>
<td>Computing Models Supported</td>
<td>Internet document exchange, entities, business processes, objects and events</td>
<td>Internet document exchange</td>
</tr>
</tbody>
</table>
Parts of EDOC

- Enterprise Collaboration Architecture (PIM)
  - Component Collaboration Architecture
  - Business Process Specification
  - Entities
  - Business Events
  - Patterns

- Technology Mapping (PSM)
  - Flow Composition Model (Messaging)
  - EJB & Corba Components
  - ebXML (In progress)
  - Others…

- MAPPING – Models are the standards and are source code
Enterprise Architecture

Supply Chain

Web Browser

HTTP

EAI Applications & B2B E-Commerce

Client Applications

Web Server Applications

Enterprise Components

XML
Corba
EJB
DCOM
MQ

Standard Middleware connects applications to components & components to components

User interface and application logic go here

Business and data rules go here

The data goes here

SQL DBMS, Client/Server & Legacy Applications

Web Browser

Web Browser
- **Business Process Specification (Like CCA)**
  - XML Representation of business process

- **Core Components**
  - Business Data Types

- **Collaboration Protocol Profile**
  - What business partners implement what business processes using what technologies
  - One-One agreement for doing business

- **Transport Routing & Packaging**
  - Messaging Built on Soap

- **Registry & Repository**
  - Finding business partners, document and process specifications
ebXML Architecture

Designtime

Business Process

Context For

Business Messages

Built With

Core Data Blocks

Register

BP Specification

CPP

Designtime

CPA

Transport

Package

CPP

Business Service Interface

Internal Business App

Business Service Interface

Internal Business App

Runtime
Component Collaboration Architecture (The Model of Doing)
The Marketplace Example

Mechanics Are Us
Buyer

Physical Delivery

GetItThere Freight
Shipper

Order
Conformation
Shipped

Process
Complete

Status

Ship Req
Shipped
Delivered

Acme Industries
Seller
The Seller’s Detail

Order Processing

Shipping

Event

Conformation

Ship Req

Shipped

Delivered
Parts of a CCA Specification

- Structure of process components and protocols
  - Process components, ports, protocols and documents
    - Class Diagram or CCA Notation

- Composition of process components
  - How components are used to specify components
    - Collaboration diagram or CCA Notation

- Choreography
  - Ordering of flows and protocols in and between process components
    - Activity Diagram
- Identify a “community process”, the roles and interactions
- Using CCA Notation
Component structure

Defines the “outside” Contract of a component
### Specification of a protocol

- **Protocol Example**

- **Protocol OrderBT**
  - **Order**
    - **OrderData**
    - **OrderDenied** (from OrderBT)
    - **OrderConfirmation** (from OrderBT)
  - **OrderBT**
    - **OrderDenied**
    - **OrderConfirmed**

- **Responder Role**: Seller
- **Initiator Role**: Buyer
Choreography of Protocol

Protocol OrderBT

Order

OrderDenied

OrderConfirmation

ResponderRole
Seller

InitiatorRole
Buyer

<<initiates>> Order

<<responds>> OrderDenied

<<responds>> OrderConfirmation

Failure

Success
Object Interfaces

- Use standard interface notation
- Are a subtype of “Protocol” in the MetaModel
- Allow modeling of and integration with classical and/or existing objects
Composition defines the “inside” of a component.

Use of an interface Composition defines the “inside” of a component.
- Entities are added to manage entity data
- Entity Roles are managers that provides a view of the same identity in another context
- The Entities have ports for managing and accessing the entities
- Non-entities which are owned by (aggregate into) an entity are managed by the entity
ECA Business Events (The Model of When …)
Event-Based Business Processes

Business Process

Business Rules

Business Entity

Business Actions

Business Events

Event Notification

Business Process

Business Rules

Business Entity

Business Actions

Business Events
Point to point Event Notification

Event Notifications
Pub/Sub Event Notification
Event Example

Business Process: Order-to-ship
Start
1. Take Order
2. Schedule
3. Ship
Done

Business Entity: Order
Initial
place
Placed
place
scheduled
schedule
Scheduled
ship
shipped
ship
Shipped
cancel
cancelled
cancel
Cancelled

Business Process: CreditCheck
Start
1. Check Cust
2. OK
Notify
approved
3. Hold
Done

Business Entity: Inventory
schedule
Available
schedule
Reserved
ship
Shipped
ship
shipped

Business Entity: Customer
check
Active
check
hold
Held
hold
held
EDOC Business Process (The Model of How)
Business Process Model

- Specializes CCA
- Activity-centric view of a Process
- Expresses
  - Complex temporal and data dependencies between business activities
  - Iteration of activities
  - Alternative required Inputs and Outputs of activities
  - Roles related to performers, artifacts and responsible parties for activities
Data Flows are special CCA Connections

- Uni-directional between ProcessFlowPorts
- DataFlows indicate
  - data dependency & transmit values at run-time, or
  - temporal dependency (aka control flow)
Patterns: Buyer/Seller

Inheritance

Composition
Agenda

1 Introduction
2 Business Model Representation
3 Business Process Modeling
4 Capturing the Organization and Location Aspects
5 Developing a Process Model
6 BPMN, BPML, and BPEL4WS
7 Business Process Interoperability
8 Summary and Conclusion
Summary – Key Business Model Engineering Objectives

- We must enable the emerging Internet Computing Model
  - Loosely coupled roles exchanging documents based on a contract of collaboration

- Web need interoperability at two levels
  - Messaging for the data
  - Metadata for the contract of collaboration, stored in repositories

- This model of collaborating roles is recursive, extending into the enterprise, into managed domains and into applications
  - Inside the enterprise we want to include resources entities, business events and business processes

- Between EDOC & ebXML we are covering B2B and intra enterprise
Course Assignments

- Individual Assignments
  - Reports based on case studies / class presentations

- Project-Related Assignments
  - All assignments (other than the individual assessments) will correspond to milestones in the team project.
  - As the course progresses, students will be applying various methodologies to a project of their choice. The project and related software system should relate to a real-world scenario chosen by each team. The project will consist of inter-related deliverables which are due on a (bi-) weekly basis.
  - There will be only one submission per team per deliverable and all teams must demonstrate their projects to the course instructor.
  - A sample project description and additional details will be available under handouts on the course Web site
Team Project

- Project Logistics
  - Teams will pick their own projects, within certain constraints: for instance, all projects should involve multiple distributed subsystems (e.g., web-based electronic services projects including client, application server, and database tiers). Students will need to come up to speed on whatever programming languages and/or software technologies they choose for their projects - which will not necessarily be covered in class.
  - Students will be required to form themselves into "pairs" of exactly two (2) members each; if there is an odd number of students in the class, then one (1) team of three (3) members will be permitted. There may not be any "pairs" of only one member! The instructor and TA(s) will then assist the pairs in forming "teams", ideally each consisting of two (2) "pairs", possibly three (3) pairs if necessary due to enrollment, but students are encouraged to form their own 2-pair teams in advance. If some students drop the course, any remaining pair or team members may be arbitrarily reassigned to other pairs/teams at the discretion of the instructor (but are strongly encouraged to reform pairs/teams on their own). Students will develop and test their project code together with the other member of their programming pair.
Document Transformation methodology driven approach

- Strategy Alignment Elicitation
  - Equivalent to strategic planning
    - i.e., planning at the level of a project set

- Strategy Alignment Execution
  - Equivalent to project planning + SDLC
    - i.e., planning a the level of individual projects + project implementation

- Build a methodology Wiki & partially implement the enablers

- Apply transformation methodology approach to a sample problem domain for which a business solution must be found

- Final product is a wiki/report that focuses on
  - Methodology / methodology implementation / sample business-driven problem solution
Team Project Approach – Initial Step

- Document sample problem domain and business-driven problem of interest
  - Problem description
  - High-level specification details
  - High-level implementation details
  - Proposed high-level timeline
Course Project

- **Project Logistics**
  - Teams will pick their own projects, within certain constraints: for instance, all projects should involve multiple distributed subsystems (e.g., web-based electronic services projects including client, application server, and database tiers). Students will need to come up to speed on whatever programming languages and/or software technologies they choose for their projects - which will not necessarily be covered in class.
  - Students will be required to form themselves into "pairs" of exactly two (2) members each; if there is an odd number of students in the class, then one (1) team of three (3) members will be permitted. There may **not** be any "pairs" of only one member! The instructor and TA(s) will then assist the pairs in forming "teams", ideally each consisting of two (2) "pairs", possibly three (3) pairs if necessary due to enrollment, but students are encouraged to form their own 2-pair teams in advance. If some students drop the course, any remaining pair or team members may be arbitrarily reassigned to other pairs/teams at the discretion of the instructor (but are strongly encouraged to reform pairs/teams on their own). Students will develop and test their project code together with the other member of their programming pair.
- After teams formed, 1/2 week to Project Concept
- 1/2 week to Revised Project Concept
- 2 to 3 iterations
- For each iteration:
  - 1/2 week to plan
  - 1 week to iteration report and demo
- Requirements: Your project focuses on two application services
- Planning: User stories and work breakdown
- Doing: Pair programming, write test cases before coding, automate testing
- Demoing: 5 minute presentation plus 15 minute demo
- Reporting: What got done, what didn’t, what tests show
- 1st iteration: Any
- 2nd iteration: Use some component model framework
- 3rd iteration: Refactoring, do it right this time
1. Cover page (max 1 page)
2. Basic concept (max 3 pages): Briefly describe the system your team proposes to build. Write this description in the form of either user stories or use cases (your choice). Illustrations do not count towards page limits.
3. Controversies (max 1 page)
First Iteration Plan (Tips)

- Requirements (max 2 pages):
  - Select user stories or use cases to implement in your first iteration, to produce a demo by the last week of class
  - Assign priorities and points to each unit - A point should correspond to the amount of work you expect one pair to be able to accomplish within one week
  - You may optionally include additional medium priority points to do “if you have time”
  - It is acceptable to include fewer, more or different use cases or user stories than actually appeared in your Revised Project Concept
First Iteration Plan (Tips)

- Work Breakdown (max 3 pages):
  - Refine as *engineering tasks* and assign to pairs
  - Describe specifically what will need to be coded in order to complete each task
  - Also describe what unit and integration tests will be implemented and performed
  - You may need additional engineering tasks that do not match one-to-one with your user stories/use cases
  - Map out a *schedule* for the next weeks
  - Be realistic – demo has to been shown before the end of the semester
Max 3 pages

Redesign/reengineer your system to use a component framework (e.g., COM+, EJB, CCM, .NET or Web Services)

Select the user stories to include in the new system
  » Could be identical to those completed for your 1st iteration
  » Could be brand new (but explain how they fit)

Aim to maintain project velocity from 1st iteration

Consider what will require new coding vs. major rework vs. minor rework vs. can be reused “as is”
2\textsuperscript{nd} Iteration Plan (Tips): Breakdown

- Max 4 pages
- Define engineering tasks, again try to maintain project velocity
- Describe new unit and integration testing
- Describe regression testing
  - Can you reuse tests from 1\textsuperscript{st} iteration?
  - If not, how will you know you didn’t break something that previously worked?
- 2\textsuperscript{nd} iteration report and demo to be presented before the end of the semester
Max 2 pages

For each engineering task from your 2nd Iteration Plan, indicate whether it succeeded, partially succeeded (and to what extent), failed (and how so?), or was not attempted

Estimate how many user story points were actually completed (these might be fractional)

Discuss specifically your success, or lack thereof, in porting to or reengineering for your chosen component model framework(s)
Max 3 pages

Describe the general strategy you followed for unit testing, integration testing and regression testing

Were you able to reuse unit and/or integration tests, with little or no change, from your 1st Iteration as regression tests?

What was most difficult to test?

Did using a component model framework help or hinder your testing?
• All Iterations Due
• Presentation slides (optional)
Assignments & Readings

- **Readings**
  - Slides and Handouts posted on the course web site
  - Textbook: Part Two-Chapters 6-8
- **Individual Assignment (assigned)**
  - See Session 5 Handout: “Assignment #2”
- **Team Project #1 (ongoing)**
  - Team Project proposal (format TBD in class)
  - See Session 2 Handout: “Team Project Specification” (Part 1)
- **Team Exercise #1 (ongoing)**
  - Presentation topic proposal (format TBD in class)
- **Project Frameworks Setup (ongoing)**
  - As per reference provided on the course Web site
Any Questions?