Software Engineering
G22.2440-001

Session 10 - Main Theme
BME Advanced Topics

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

• Business Process Interoperability
• EDOC and ebXML
• Component Collaboration Architecture
• ECA Entity Profile
• ECA Business Events
• EDOC Business Processes
• Patterns
Summary of Previous Session

• Software Engineering Tools Primer
  – Build Tools (e.g., Ant)
  – Continuous Build Process Frameworks (e.g., CruiseControl)
  – Unit Testing Frameworks (e.g., jUnit)
  – Refactoring Browsers (e.g., IntelliJ’s IDEA)
  – Selecting Appropriate Tools

• Summary
  – Individual Assignment #5
  – Readings

Part I

Business Process Interoperability
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

Generic Interoperability Methodology

- Example: ECIMF methodology
Generic Interoperability Methodology

- Example: Relationship between the ECIML and other modeling standards.

Semantic Translation Layer

- Mapping concepts from different ontologies
Semantic Translation Layer

- Semantic Translation meta-model

Business Process Mediation Layer

- Example scenario that requires Process Mediator.
• Business Context model as seen by the shipping agency

• Business Context model as seen by the customer
**Process Mediation**

<table>
<thead>
<tr>
<th>Party</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration Task</td>
<td>Payment Collaboration</td>
</tr>
<tr>
<td>Framework</td>
<td>RIN 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>PPSM0: Request for quote</td>
<td>Initiator</td>
<td>Quotation</td>
<td>QuotationConfirm</td>
</tr>
<tr>
<td>PPSM1: Request Purchase Order</td>
<td>Responder</td>
<td>PURCHASEORDER</td>
<td>PURCHASEORDERConfirm</td>
</tr>
<tr>
<td>PPSM3: Notice of Invoice</td>
<td>Initiator</td>
<td>Invoice</td>
<td>Invoice</td>
</tr>
<tr>
<td>PPSM4: Notice of Remittance advice</td>
<td>Initiator</td>
<td>RemittanceAdvice</td>
<td>RemittanceAdvice</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Party</th>
<th>Shipping Agency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration Task</td>
<td>Payment Collaboration</td>
</tr>
<tr>
<td>Framework</td>
<td>EDF/FACT</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>Purchase Order</td>
<td>Responder</td>
<td>ORDERS</td>
<td>ORDERSConfirm</td>
</tr>
<tr>
<td>Notice of Invoice</td>
<td>Initiator</td>
<td>INVOICE</td>
<td>INVOICE</td>
</tr>
<tr>
<td>Notice of remittance advice</td>
<td>Responder</td>
<td>REMADV</td>
<td>REMADV</td>
</tr>
</tbody>
</table>

| Message delivery control | Any | Secure Flow |

**Process Mediation**

![Diagram of Process Mediation]

- Customer (RNIF)
  - Transaction boundaries (also legal)
  - SecureFlow QuoteReq
  - SecureFlow QuoteConfirm
  - SecureFlow POReq
  - SecureFlow POConfirm
  - SecureFlow Invoice
  - SecureFlow RemAdv

- Payment

- Process Mediator
  - SecureFlow QuoteReq
  - SecureFlow QuoteConfirm
  - SecureFlow POReq
  - SecureFlow POConfirm
  - SecureFlow Invoice
  - SecureFlow RemAdv

- Shipping Agency (EDI)
  - Transaction boundaries (also legal)
  - SecureFlow Quotes
  - SecureFlow ORDERS
  - SecureFlow ORDERSConfirm
  - SecureFlow INVOICE
  - SecureFlow REMADV
  - APERAK
  - CONTROL

? ?
### Semantic Translation

- **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: needs to be obtained from a product catalogue (external resource)</td>
<td>Height</td>
</tr>
<tr>
<td>Width</td>
<td>Box → TV_set: needs to be obtained from a product catalogue (external resource) using productID</td>
<td>Width</td>
</tr>
<tr>
<td>Depth</td>
<td>Box → TV_set: not needed</td>
<td>Depth</td>
</tr>
<tr>
<td>Not available</td>
<td>TV_set → Box: needs to be obtained from a product catalogue (external resource)</td>
<td>N/A</td>
</tr>
<tr>
<td>N/A</td>
<td>Box → TV_set: not needed</td>
<td>N/A</td>
</tr>
<tr>
<td>Fragile</td>
<td>TV_set → Box: always set to True</td>
<td>N/A</td>
</tr>
<tr>
<td>Marks the payload as fragile (requiring special care during transportation)</td>
<td>Fragile</td>
<td></td>
</tr>
<tr>
<td>StackingLevels</td>
<td>TV_set → Box: not needed</td>
<td>StackingLevels</td>
</tr>
<tr>
<td>Represents the number of levels the boxes can be stacked, one on top of the other.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>TV_set → Box: needs to be obtained from a product catalogue (external resource)</td>
<td>Weight</td>
</tr>
<tr>
<td>Represents the weight of the box with the contents.</td>
<td>TV_set</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td>TV_set → Box: not needed</td>
<td>Color</td>
</tr>
<tr>
<td>Box → TV_set: needs to be obtained from a product catalogue (external resource)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SecureFlow

- **Document**
- **Signal**
- **Receipt**
- **Exception**

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

<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 PURRequest ↔ ORDERS PURConfirm ↔ 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>APERAN</td>
</tr>
<tr>
<td><strong>ReceiptAck</strong></td>
<td>This signal means that the document business data has been accepted for further processing (which implies also well-formedness)</td>
<td>ORDERS</td>
</tr>
<tr>
<td><strong>ReceiptAckException</strong></td>
<td>This signal means the document was not well-formed (parsing errors). Business data was not considered.</td>
<td>ORDRESP</td>
</tr>
<tr>
<td><strong>Exception</strong></td>
<td>The semantics of both messages is identical, which means a 1:1 mapping can be applied, both ways.</td>
<td>INVOC</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>In this particular case, the EDI system uses APERAN and CONTROL messages only to signal exceptions. Acknowledgements are implicit, in the form of response business documents.</td>
<td>REMADV</td>
</tr>
</tbody>
</table>

- N/A – implementation choice (positive acknowledgements are implicit).

- EDI → RNIF: needs to be synthesized from the response document. Possible problems with timing constraints: ... (ack. too late)

- RNIF → EDI: not needed – don’t forward.

- SecureFlow: This message is sent when parsing errors occur. Business data was not considered at all.
Syntax mapping

- Message syntax mapping

Shared ontology approach to semantic translation

Multiple ontologies + labels

Shared ontology

local ontology

local ontology

local ontology
Part II

EDOC and 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
  - Not technical transactions

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

Metadata (model) interoperability

Each can be transformed

Drilling down – inside a role

- 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
Standards for collaboration

<table>
<thead>
<tr>
<th>Standards</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

Parts of ebXML

- 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
Summary of points thus far

- 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
Part III

*Component Collaboration Architecture*  
*(the model of doing)*
The Seller’s Detail

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
The Community Process

- Identify a “community process”, the roles and interactions
- Using CCA Notation

Component structure

Component structure
Defines the “outside”
Contract of a component
Protocol Example

- Specification of a protocol

Choreography of Protocol
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

Use of an interface

Composition defines the “inside” of a component
Part IV

ECA Entity Profile
(the model of things)

Sample Information Model
Adding Entities

- Entities are added to manage entity data
- Entity Roles are managers that provide 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

Part V

ECA Business Events
(the model of when …)
Event Based Business Processes

Point to point Event Notification
Pub/Sub Event Notification

Event Example
Part VI

EDOC Business Processes
(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

- **DataFlows** are special CCA Connections
  - Uni-directional between ProcessFlowPorts
  - DataFlows indicate
    - data dependency & transmit values at run-time, or
    - temporal dependency (aka control flow)

Part VII

Patterns
Part VIII

Conclusion
Course Assignments

- Individual Assignments
  - Problems and reports based on case studies or exercises

- 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 consists 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.

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.
Sample Project Methodology
Very eXtreme Programming (VXP)

• 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

Sample Project Methodology
Very eXtreme Programming (VXP)
(continued)

• 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
Revised Project Concept (Tips)

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

2nd Iteration Plan (Tips):
Requirements

• 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”
2nd 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 1st iteration?
  - If not, how will you know you didn’t break something that previously worked?
- 2nd iteration report and demo to be presented before the end of the semester

2nd Iteration Report (Tips):
Requirements

- 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)
2nd Iteration Report (Tips): Testing

- 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?

Project Presentation and Demo

- All Iterations Due
- Presentation slides (optional)
Readings

• Readings
  • Slides and Handouts posted on the course web site
  • Slides and Handouts posted on the course web site
  • Documentation provided with software engineering tools

• Project Frameworks Setup (ongoing)
  • As per references provided on the course Web site

• Individual Assignment
  • See Session 9 Handout: “Assignment #5”

• Team Assignment
  • See Session 10 Handout: “Team Project” (Part 3)

Next Session
Building Software

• Software Engineering Teams
• Pair Programming
• Pair Programming Effects
• Refactoring
• Test Driven Development (TDD)
• Distributed Development and Agile
• Methods Scalability
• Languages, Platforms, and Component Infrastructures
• Summary
• Project (Part 3)