# Model Synchronization and Traceability

Kostas Kontogiannis

*University of Waterloo*
*Canada*

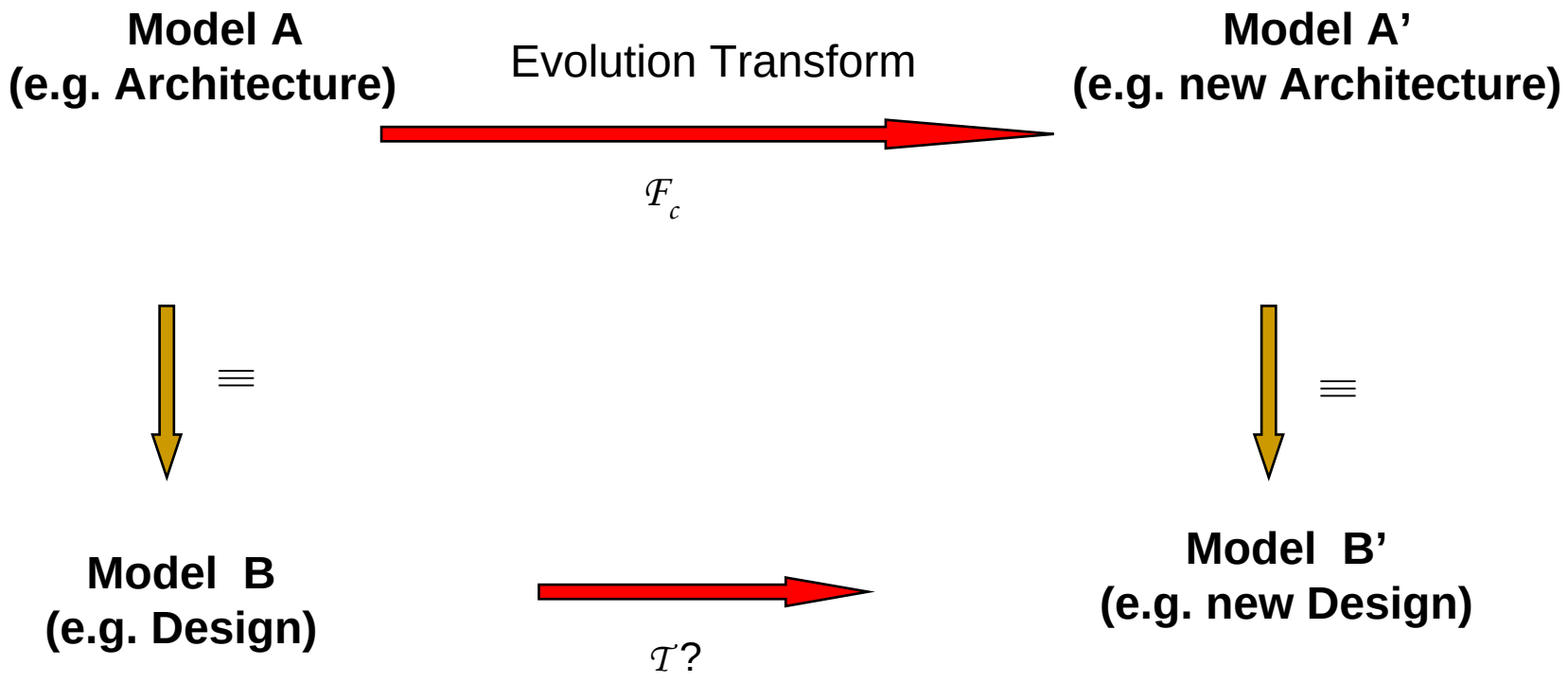
# Motivation

- Software evolution refers to the continuous change that a software-intensive system endures from its inception to its retirement

- Synchronization of various artefacts and models during software evolution is a practical problem:
  - Need to ensure that system requirements, business processes, system architecture, design, and implementation are kept synchronized throughout the software life-cycle
  - Need to devise a systematic approach not only for dealing with the underlying problems but also to fit with modern process models such as RUP and with modern IDE frameworks such as RSA
  - The fundamental premise is that software evolution is also part of the development life cycle and not only part of the software maintenance
  - Synchronization can be achieved using model transformations

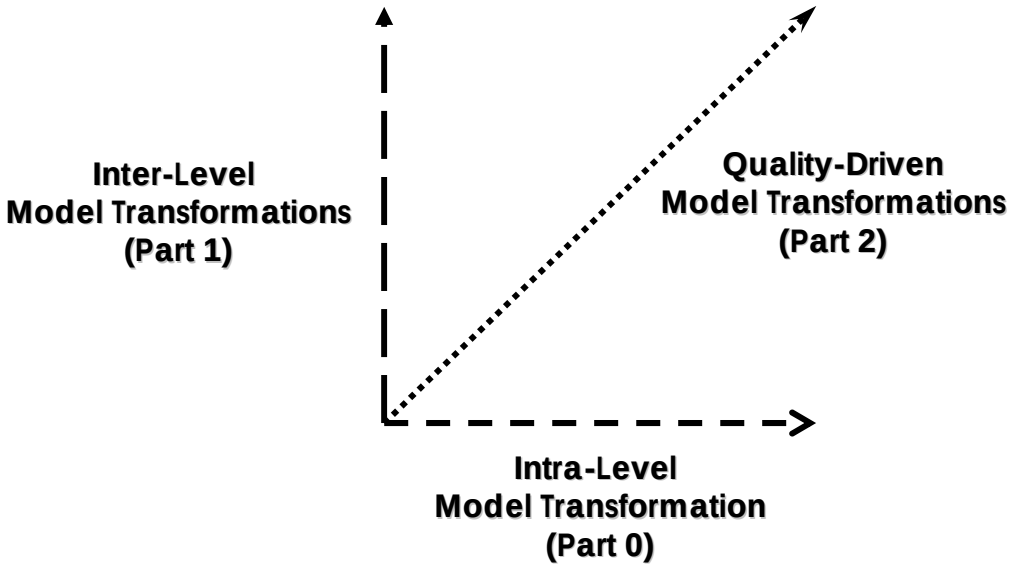
# Model Synchornization

- Model Synchronization: Schematic



# Model Synchronization Scope

- We identify three dimensions of Model Transformations:



# Model Transformation Categories

- **Intra-level Model Transformations**, where changes are propagated across models on the same level of abstraction, e.g., changes at the source code update the call graph

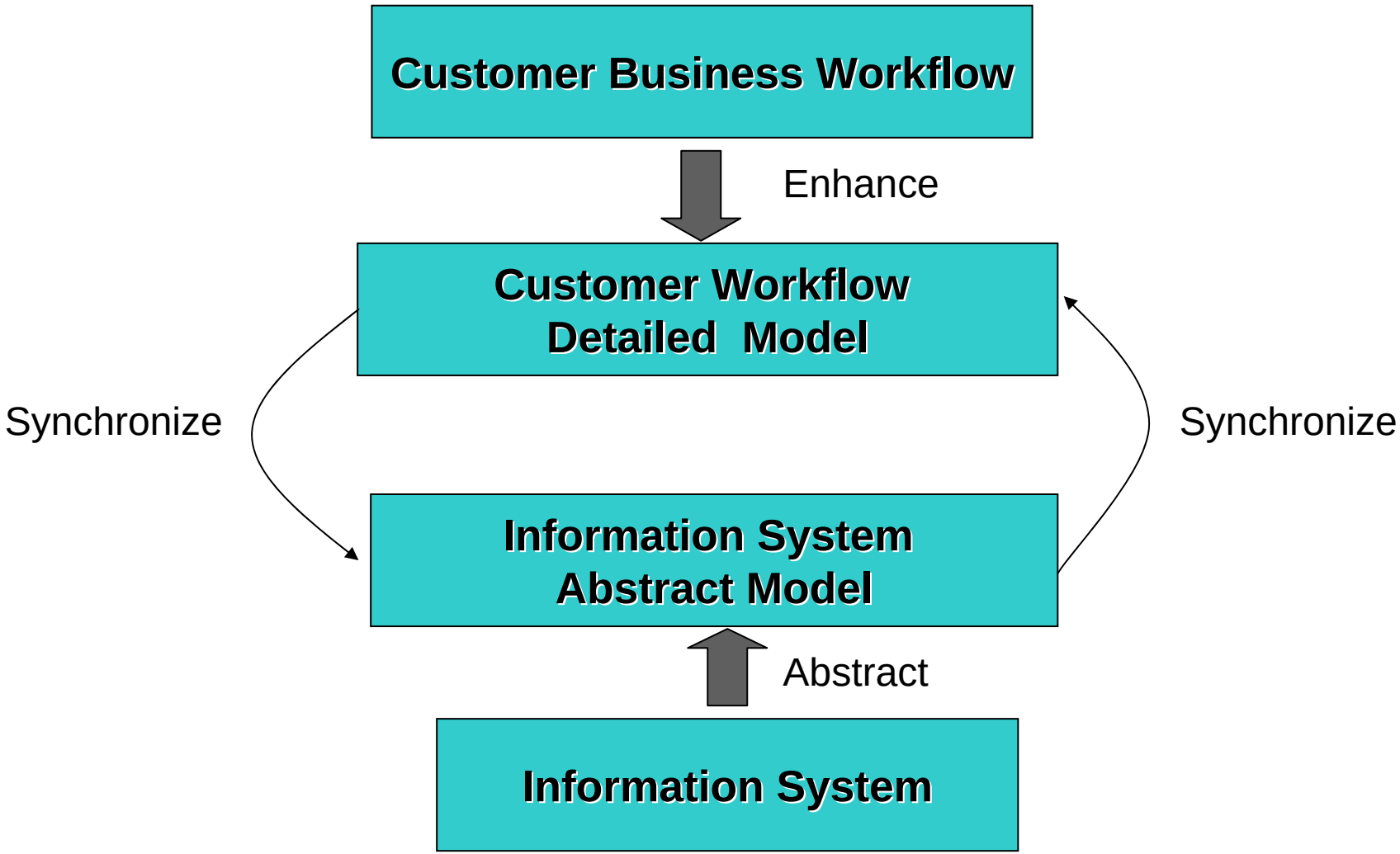
- **Inter-level Model Transformations**, where changes are propagated across models on different levels of abstraction, e.g., changes from UML class diagram are propagated to source code in one direction and to architectural diagrams in the other direction

- **Quality-Driven Model Transformations**, where a particular quality attribute is the key instigator of change e.g., improve maintainability

# Application: Synchronizing Workflow Models

- Business applications are subject to constant changes
  - From the business manager's standpoint:
    - Evolution of workflows and business processes
    - Customization of tasks, activities, and responsibilities
  - From the developer's standpoint
    - Addition of new features
    - Migration to new software technology, updates, and fixes
- Over time, the associations between business workflows and source code are lost
- The objective of this work is to devise a framework that:
  - Assists on the synchronization of business flows with its underlying source code implementation, and vice versa
  - Extend the above for devising a broader model synchronization / model generation technique for commerce

# Top level view: Model Synchronization

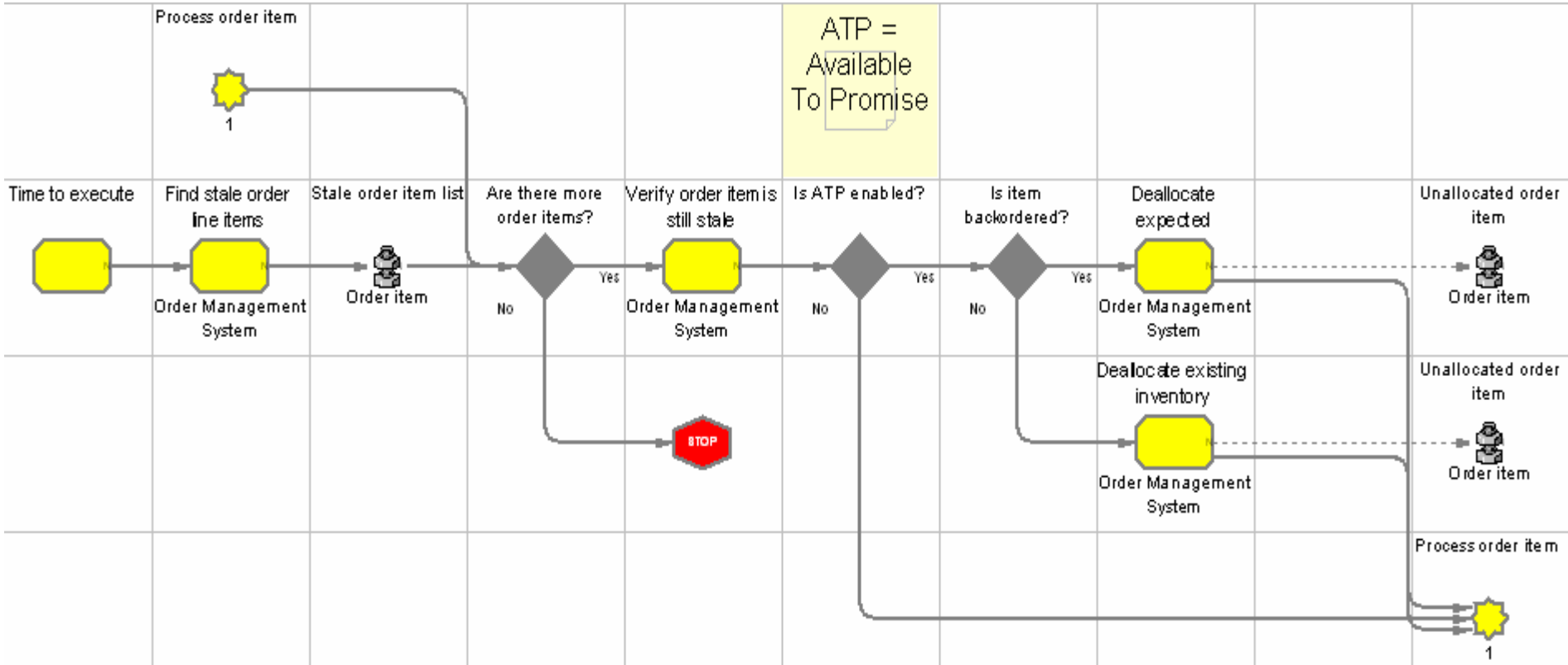


# Challenges

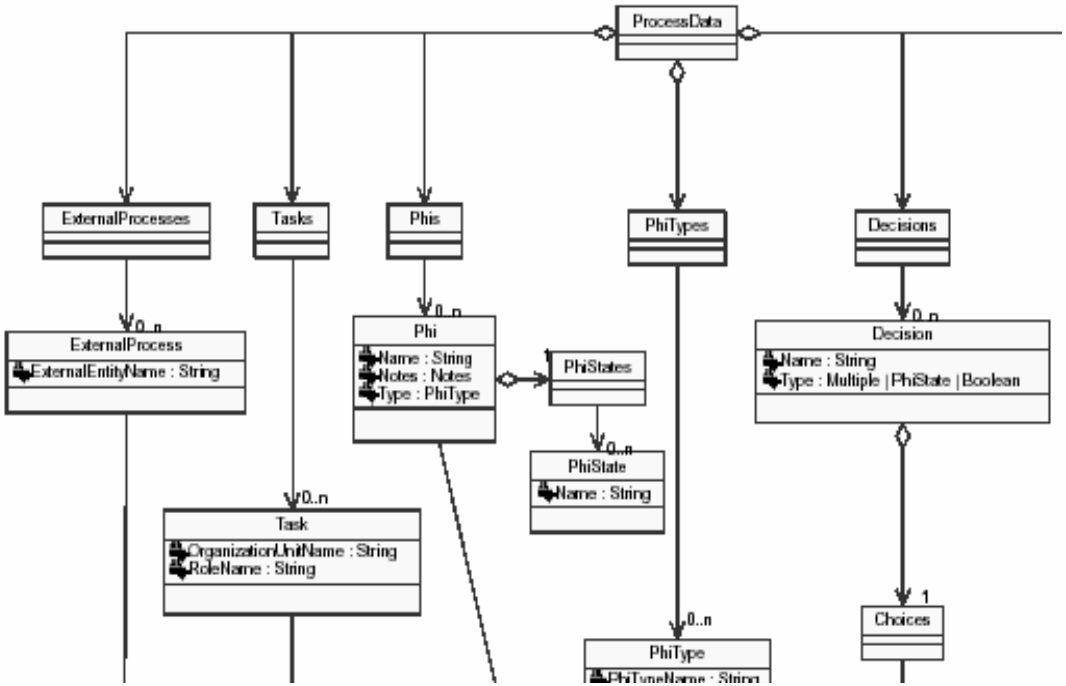
- Modeling of business flows
  - Structural models
  - Behavioral models
- Analysis and modeling of source code
  - Identification of tasks
  - Abstraction of data and control flows
  - Source code representation models
- Identification and modeling of business flow and source code dependencies
- Design and implementation of the synchronization algorithms

# Customer Business Flows

- Depict workflows at a conceptual level
- Contain tasks, data, decisions
- Objective is to annotate, denote, and represent workflows in a form that can be read and processed by a software program

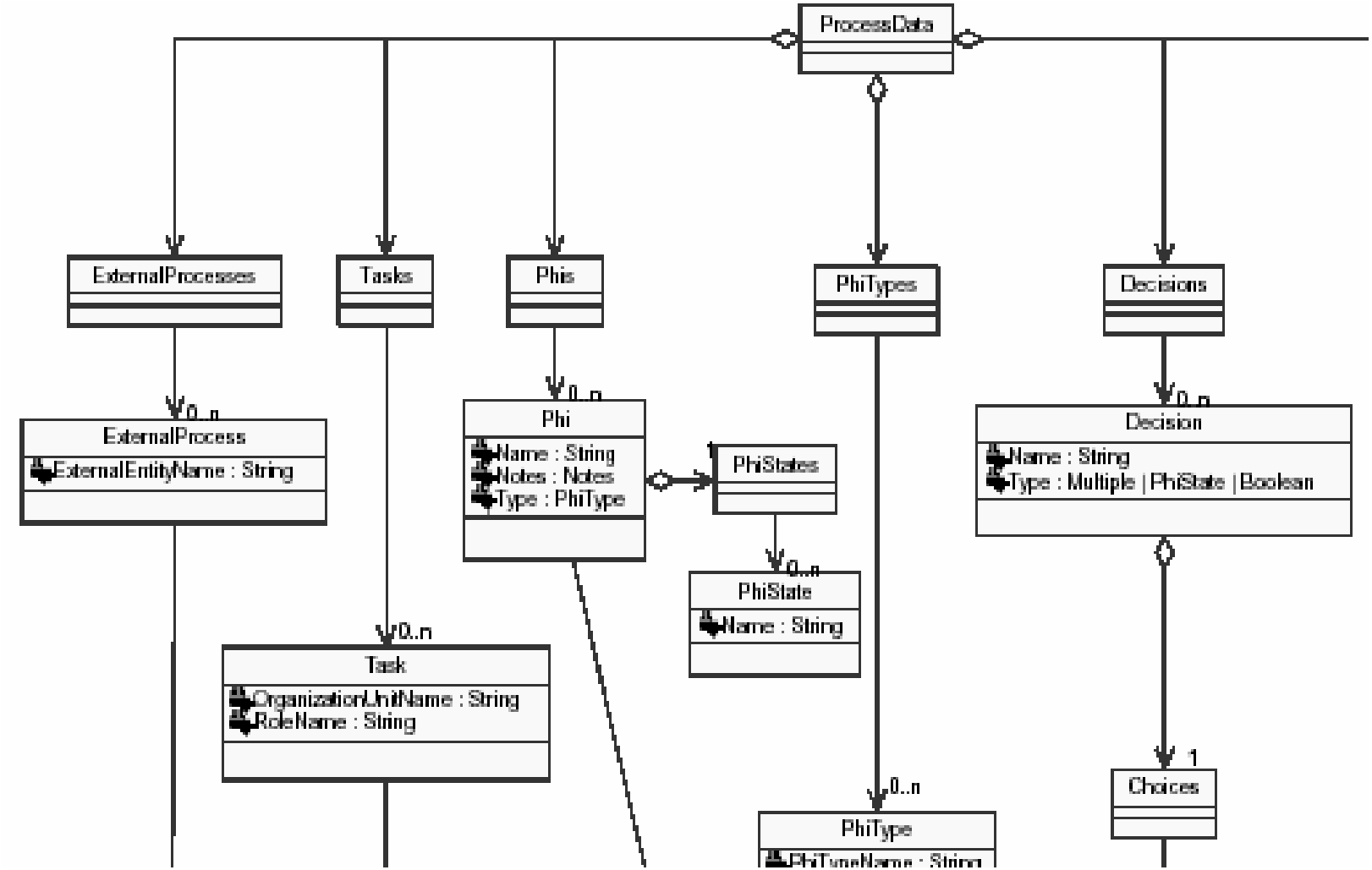


# Business Flow Domain Model




```
- <WFBPR>
  - <Processes>
    - <Process>
        <ProcessName Name="Release expired allocations" />
      - <ProcessNotes>
          <Notes>Objective Recover inventory allocations that are stale or have expired. Description
            This process is a scheduled process that releases allocations. It is typically used to
            recover allocated inventory from abandoned shopping carts. Features -- Auctions --
            Inventory hold Customization Not applicable Edition Professional, Business
            Edition</Notes>
        </ProcessNotes>
        <ProcessType Type="StandardProcess" />
        <ExportType Type="Process" />
      - <ValidFrom>
        - <DateTime>
            <Date Year="2003" Month="05" Day="10" />
            <Time Hour="08" Minute="00" Second="00" />
          </DateTime>
        </ValidFrom>
      - <DataFlow>
        - <InputContainer>
            <DataFieldName Name="Default Data Structure" />
          </InputContainer>
        - <OutputContainer>
            <DataFieldName Name="Default Data Structure" />
          </OutputContainer>
        </DataFlow>
      - <ViewOptions>
        - <TaskObjectViewOptions>
            <UpperText Text="TaskObjectName" />
            <LowerText Text="RoleName" />
          </TaskObjectViewOptions>
        - <PhiObjectViewOptions>
            <UpperText Text="PhiObjectName" />
```

# Business Flow Domain Model



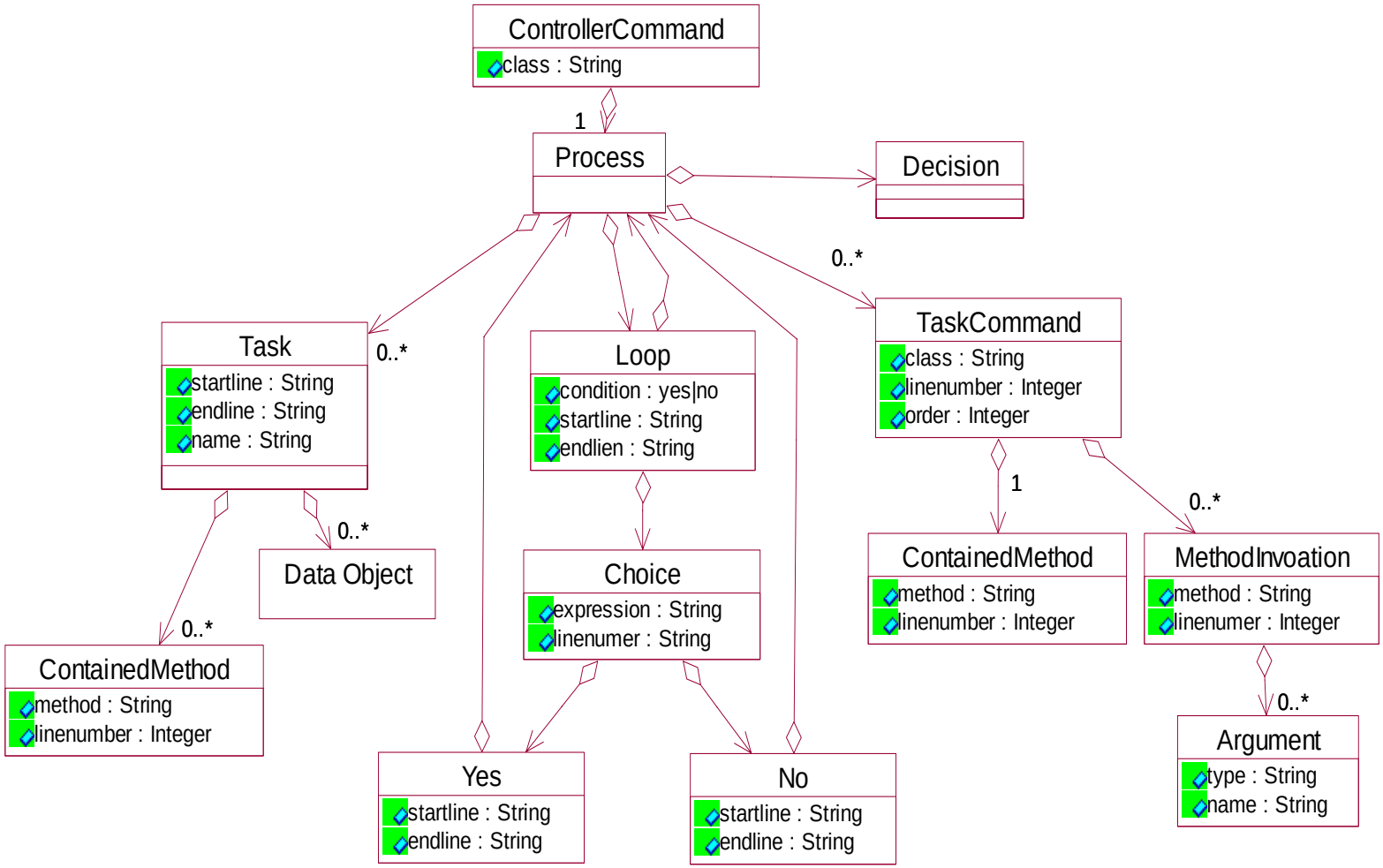
# Workflow Domain Model

```
- <WFBPR>
  - <Processes>
    - <Process>
        <ProcessName Name="Release expired allocations" />
      - <ProcessNotes>
          <Notes>Objective Recover inventory allocations that are stale or have expired. Description
            This process is a scheduled process that releases allocations. It is typically used to
            recover allocated inventory from abandoned shopping carts. Features -- Auctions --
            Inventory hold Customization Not applicable Edition Professional, Business
            Edition</Notes>
        </ProcessNotes>
        <ProcessType Type="StandardProcess" />
        <ExportType Type="Process" />
      - <ValidFrom>
        - <DateTime>
            <Date Year="2003" Month="05" Day="10" />
            <Time Hour="08" Minute="00" Second="00" />
          </DateTime>
        </ValidFrom>
      - <DataFlow>
        - <InputContainer>
            <DataFieldName Name="Default Data Structure" />
          </InputContainer>
        - <OutputContainer>
            <DataFieldName Name="Default Data Structure" />
          </OutputContainer>
        </DataFlow>
      - <ViewOptions>
        - <TaskObjectViewOptions>
            <UpperText Text="TaskObjectName" />
            <LowerText Text="RoleName" />
          </TaskObjectViewOptions>
        - <PhiObjectViewOptions>
            <UpperText Text="PhiObjectName" />
```

# WC Information System Analysis

- Represent the source code flow of controller and task commands
- Model contains software components, database access beans, conditions
- Analyze source code to identify heuristics of business logics
- Extract workflow process models from source code and represent workflow models using XML
- Interpret XML represented flows as a graph
- Source code analysis performed by Ying Zou's team at Queen's University

# WC Source Code Domain Model



# Example Source Analysis

```
OrderJDBCHelperAccessBean abOrderJDBCHelper =
                        new OrderJDBCHelperAccessBean();
Vector vOrderItems = abOrderJDBCHelper.findStaleOrderItems(storeId);

// Turn the Vector into an Enumeration for performance considerations
Enumeration enumOrderItems = vOrderItems.elements();

getCommandContext().getTransactionCache().flush();

try {
     Action.proceed();
     }
catch (javax.transaction.RollbackException ex) {
      throw new ECSystemException(……);
}
```

**Business Logic**

**Java API Class**

**Accessor Method**

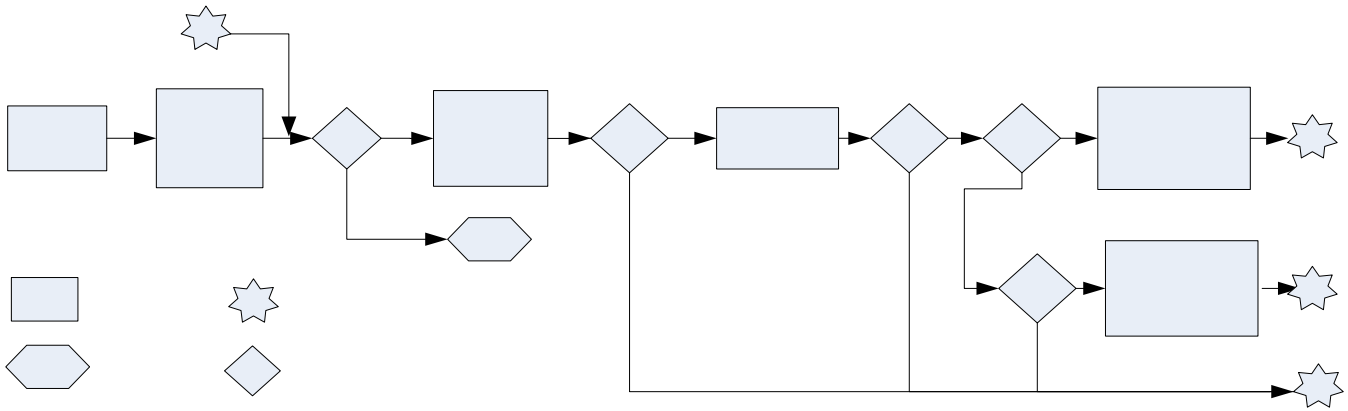
**Routine Class**

**Exception Class**

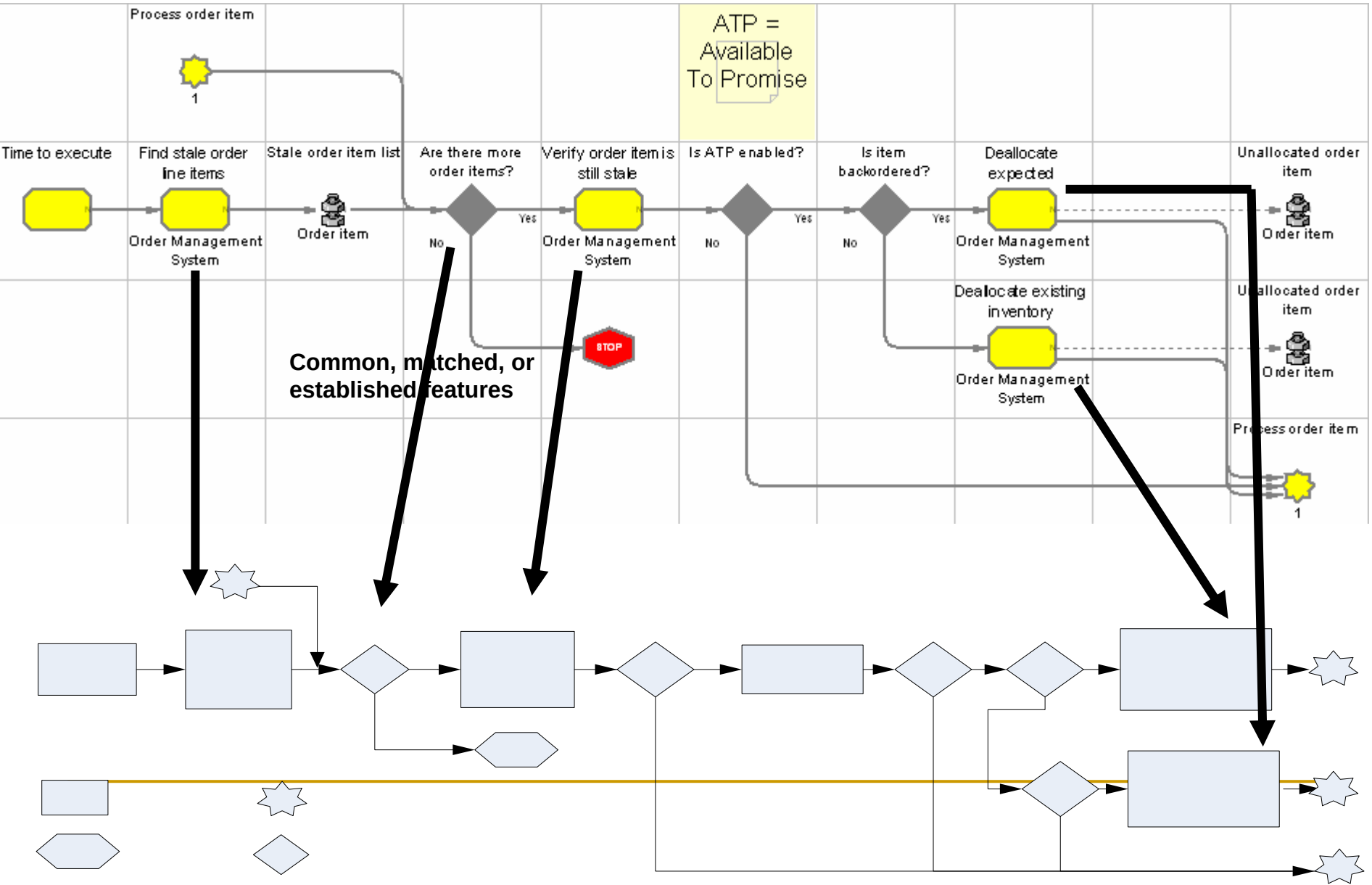
# Source Code Domain Model

```
OrderJDBCHelperAccessBean abOrderJDBCHelper =
                    new OrderJDBCHelperAccessBean();
Vector vOrderItems =
        abOrderJDBCHelper.findStaleOrderItems(storeId);

// Turn the Vector into an Enumeration for performance considerations
Enumeration enumOrderItems = vOrderItems.elements();

getCommandContext().getTransactionCache().flush();

try {
        TransactionManager.commit();
        }
catch (javax.transaction.RollbackException ex) {
         throw new ECSystemException(……);
}
```

```
<Task name="startUse" type="source" linenumber="152" />
<Task name="findStaleOrderItems" type="source" linenumber="163" />
<Decision expression="hasMoreElements" linenumber="177" />
<Loop condition="yes" startline="177" endline="232">
  <Task name="verifyStaleOrderItems" type="source" linenumber="186" />
- <Choice expression="abOrderJDBCHelper.verifyStaleOrderItems(storeId,orderitemsId)" linenumber="186">
  - <Yes startline="186" endline="225">
      <Task name="IsUsingATP" type="source" linenumber="195" />
    - <Choice expression="bATPEnabled" linenumber="196">
      - <Yes startline="197" endline="214">
        - <Choice expression="abOrderitem.getInventoryStatus().toUpperCase().equals(ALLC)"
            linenumber="200">
          - <Yes startline="200" endline="205">
            + <TaskCommand name="DeallocateExistingInventoryCmd" linenumber="201">
            </Yes>
          - <No startline="206" endline="210">
            - <Choice expression="abOrderitem.getInventoryStatus().toUpperCase().equals(BO)"
                linenumber="207">
              - <Yes startline="207" endline="212">
                + <TaskCommand name="DeallocateExpectedInventoryCmd" linenumber="208">
                </Yes>
```



# Establishing Dependencies



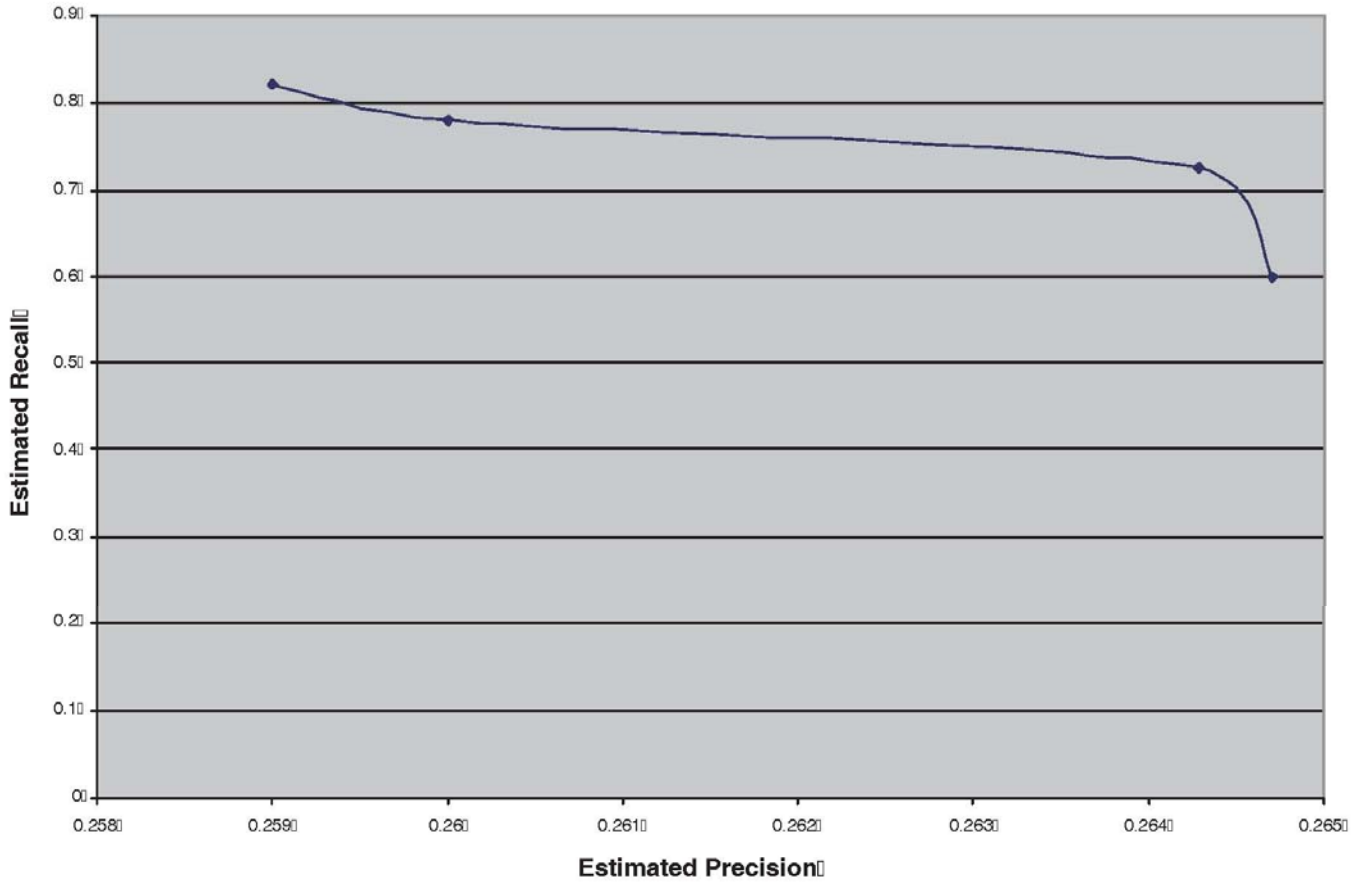
# Establishing dependencies

- Establishing dependencies between models of different type systems and levels of abstraction is based on:
  1. Incremental convergence of the domain models through type association rules
  2. Use of Formal Concept Analysis to cluster elements that relate to the same concept
- Elements that cluster together they are considered that they relate to the same concept or are dependent
- Synchronization becomes a problem of traversing dependencies and maintaining valid associations between model elements

# Modeling Dependencies

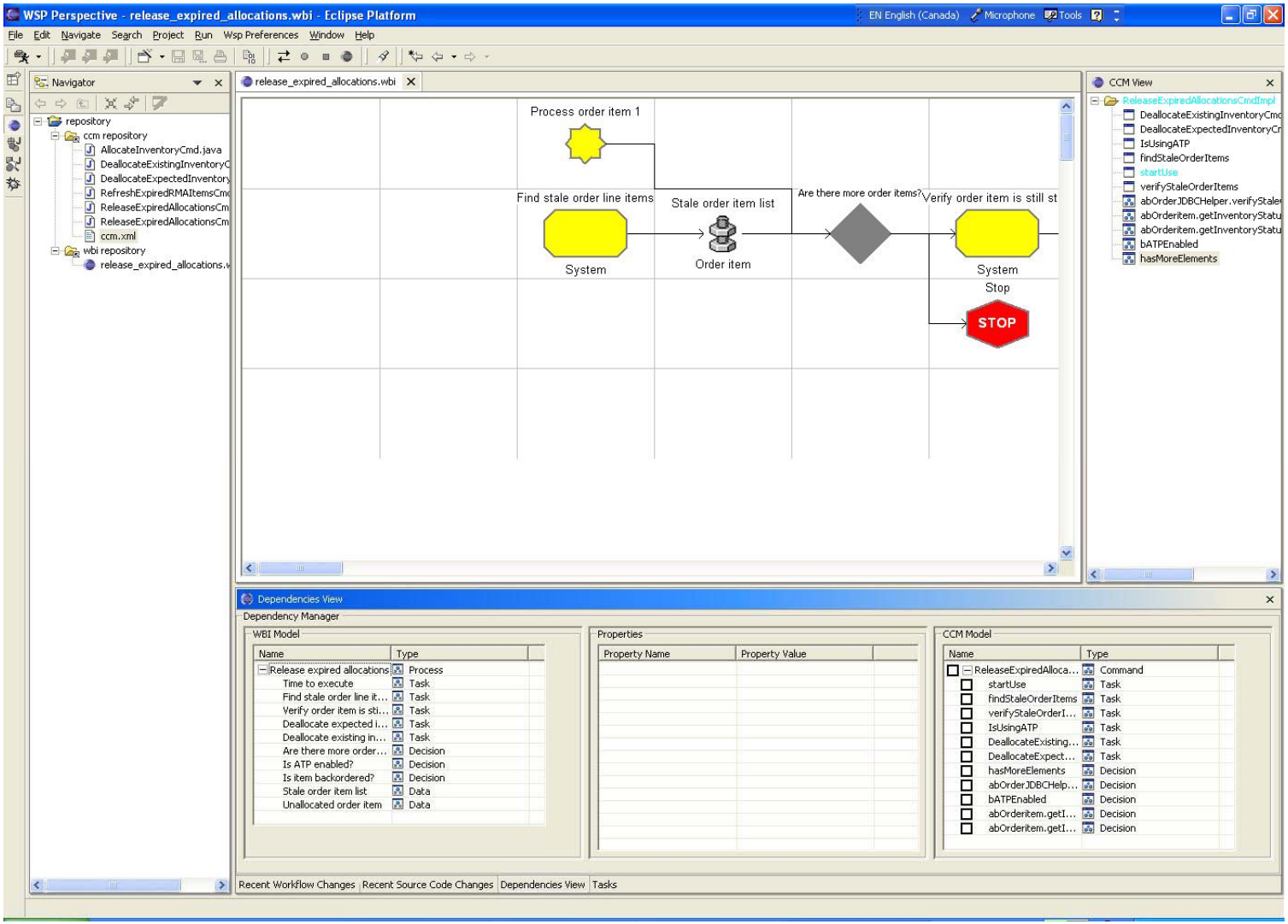
```
- <Dependencies>
  - <Dependency>
      <Source Name="Time to execute" />
      <Target Name="ReleaseExpiredAllocationsCmdImpl" />
      <Ontology>ReleaseExpiredAllocations</Ontology>
      <Functionality>Release the expired allocation</Functionality>
    </Dependency>
  - <Dependency>
      <Source Name="Time to execute" />
      <Target Name="startUse" />
      <Ontology>ReleaseExpiredAllocations</Ontology>
      <Functionality>Entry point</Functionality>
    </Dependency>
  - <Dependency>
      <Source Name="Find stale order line items" />
      <Target Name="ReleaseExpiredAllocationsCmdImpl" />
      <Ontology>ReleaseExpiredAllocations/FindStaleOrder</Ontology>
      <Functionality>Find the expired order</Functionality>
    </Dependency>
  - <Dependency>
      <Source Name="Find stale order line items" />
      <Target Name="findStaleOrderItems" />
      <Ontology>ReleaseExpiredAllocations/FindStaleOrder</Ontology>
      <Functionality>Find the expired order</Functionality>
    </Dependency>
  - <Dependency>
      <Source Name="Verify order item is still stale" />
      <Target Name="ReleaseExpiredAllocationsCmdImpl" />
      <Ontology>ReleaseExpiredAllocations/VerifyOrder</Ontology>
      <Functionality>Verify that the specified order is stale</Functionality>
    </Dependency>
  - <Dependency>
      <Source Name="Verify order item is still stale" />
      <Target Name="verifyStaleOrderItems" />
```

# Results of the Matching Process

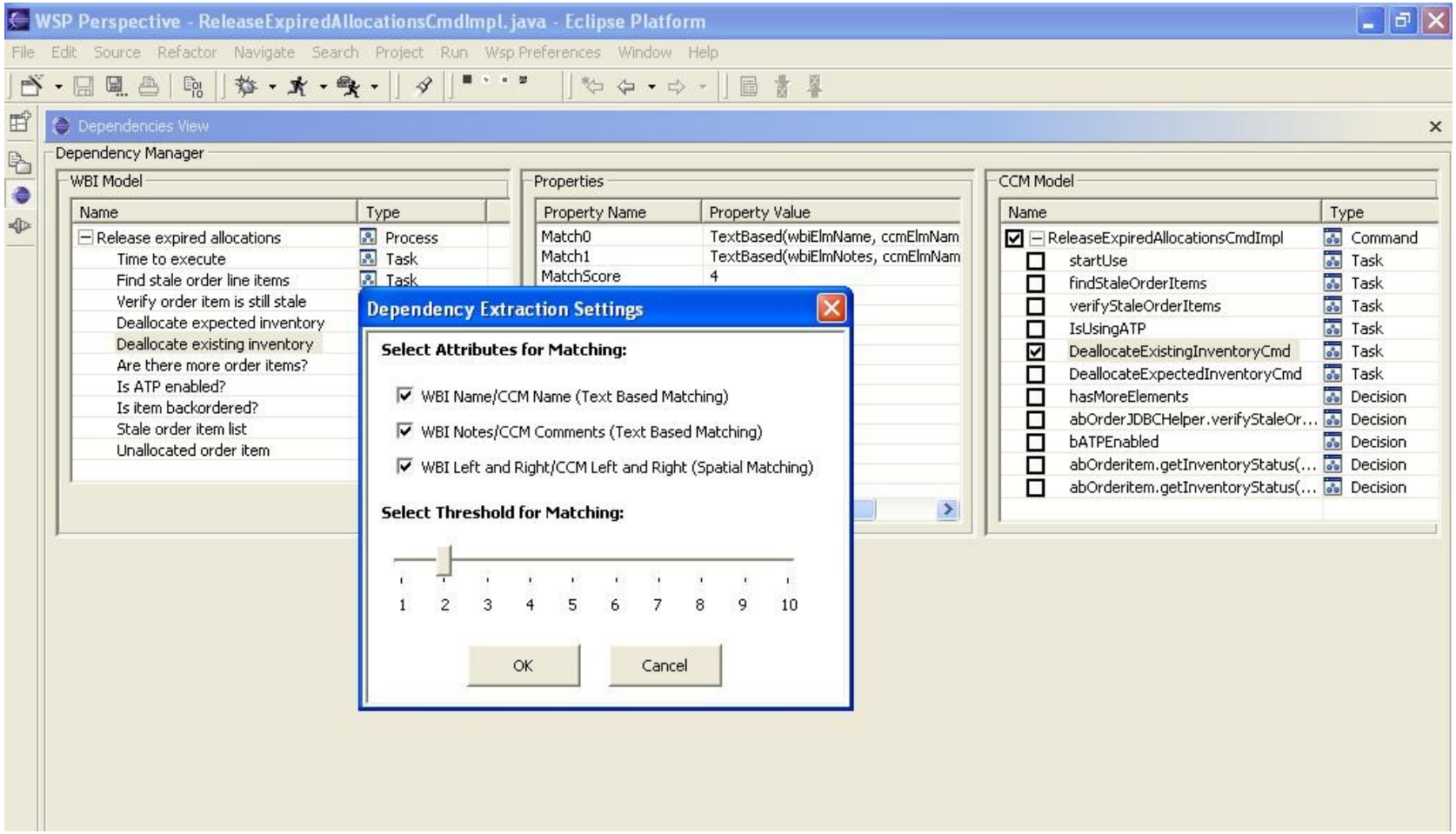



Re

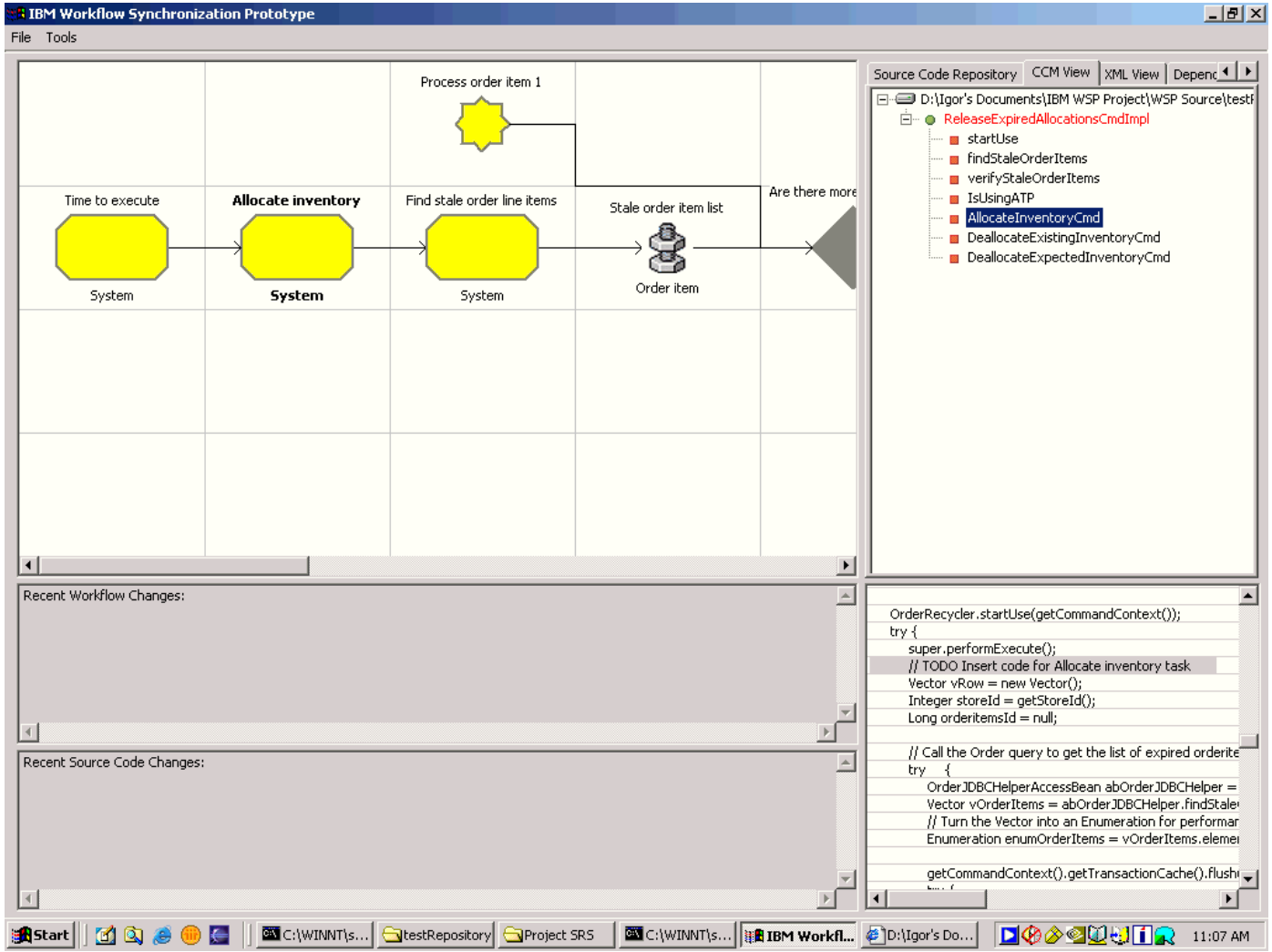
# Prototype System



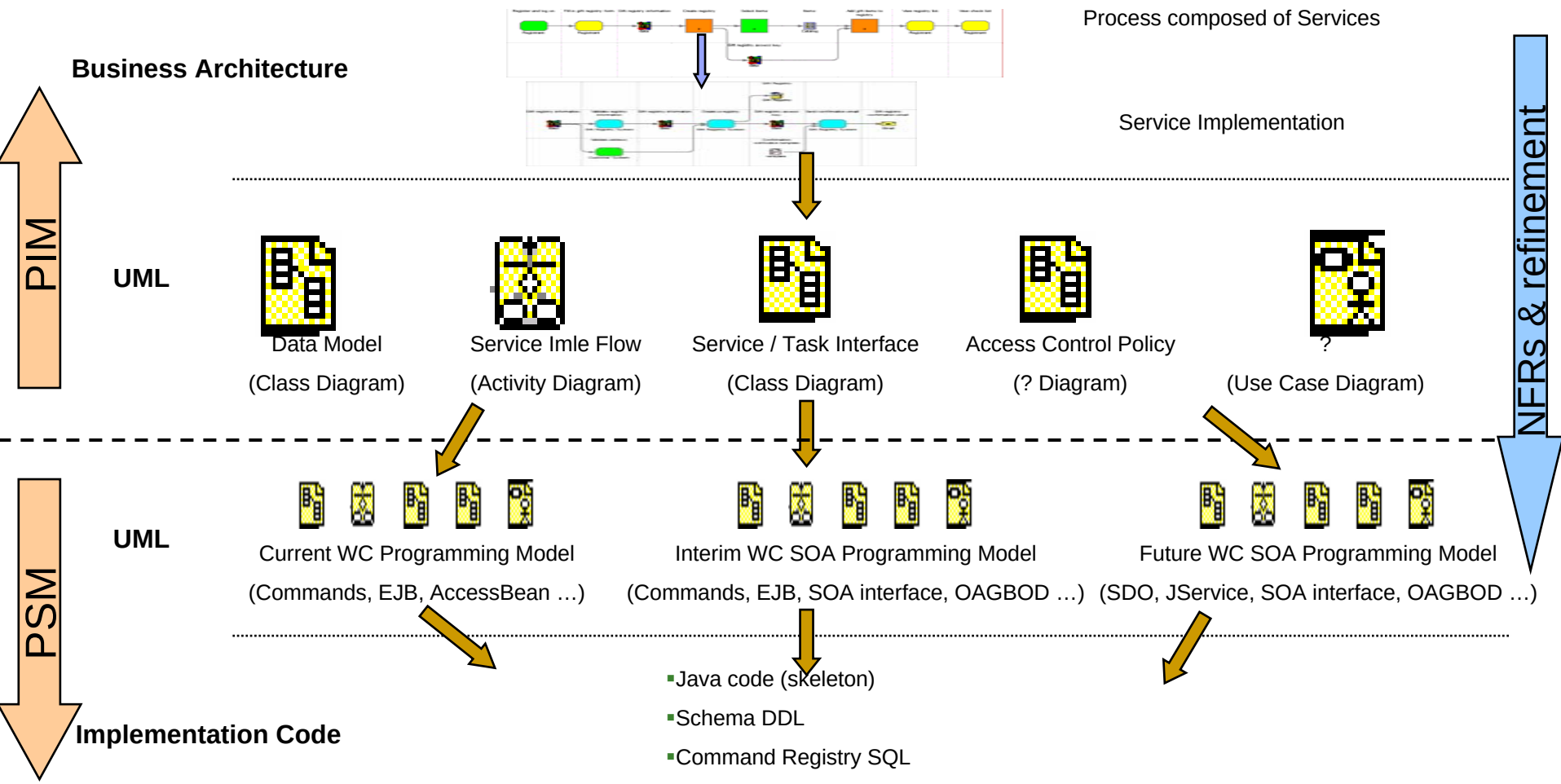
# Prototype System (Dependencies view)



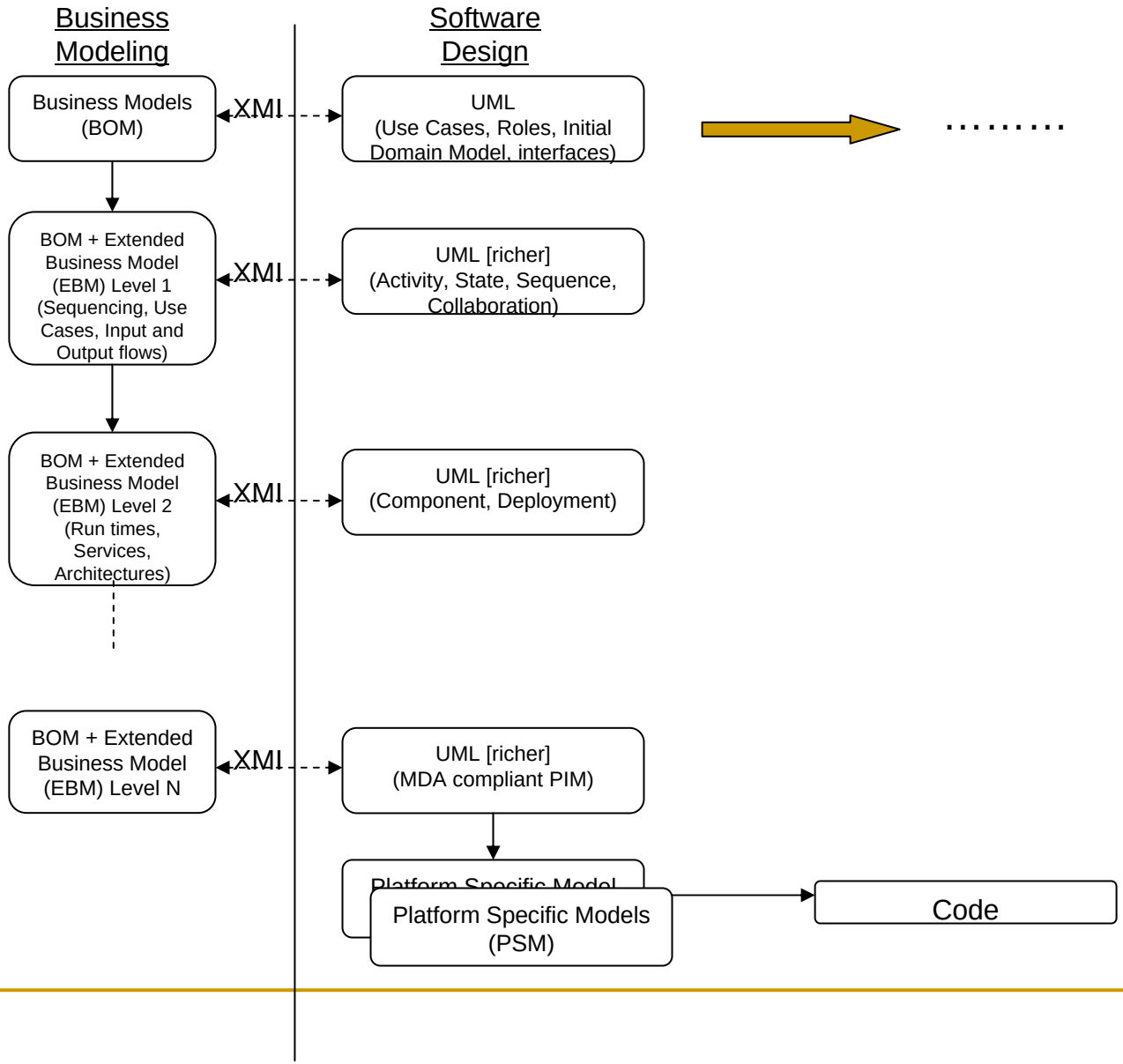
# Prototype System



# The Next Steps: Model Driven System Development



# Iterative Model Enhancements



# Summary and Uses

- Objective is on devising techniques to analyze, synchronize, and simulate Commerce models
- Specific application focuses on synchronizing WBI process models and WCS source code models
- Semi-automatic extraction of dependencies between model elements is possible and synchronization can be automated
- A prototype system is being developed and is being ported as a plug-in to Eclipse
- Potential uses include:
  - Extraction of Process descriptions  from source code (Reverse engineering use)
  - Compliance checking of WBI models with WCS source code models
  - Support of evolution and customization activities (development teams, field teams)
  - Application code generation with emphasis on satisfying specific NFRs

# Model Synchronization and Traceability

Kostas Kontogiannis

*University of Waterloo*
*Canada*