Extracting Business Processes from Three-Tier Architecture Systems

Maokeng (Alex) Hung and Ying (Jenny) Zou

Department of Electrical and Computer Engineering
Queen's University

Introduction

- Organizations employ information systems to automate business processes and perform tasks
- Fast reaction to rapid requirement changes is the key to maintain their competitive edges
- Software maintenance cost is high because workflow extraction is performed by programmers manually
- Automatic extraction will reduce both of the cost and time and increase the performance

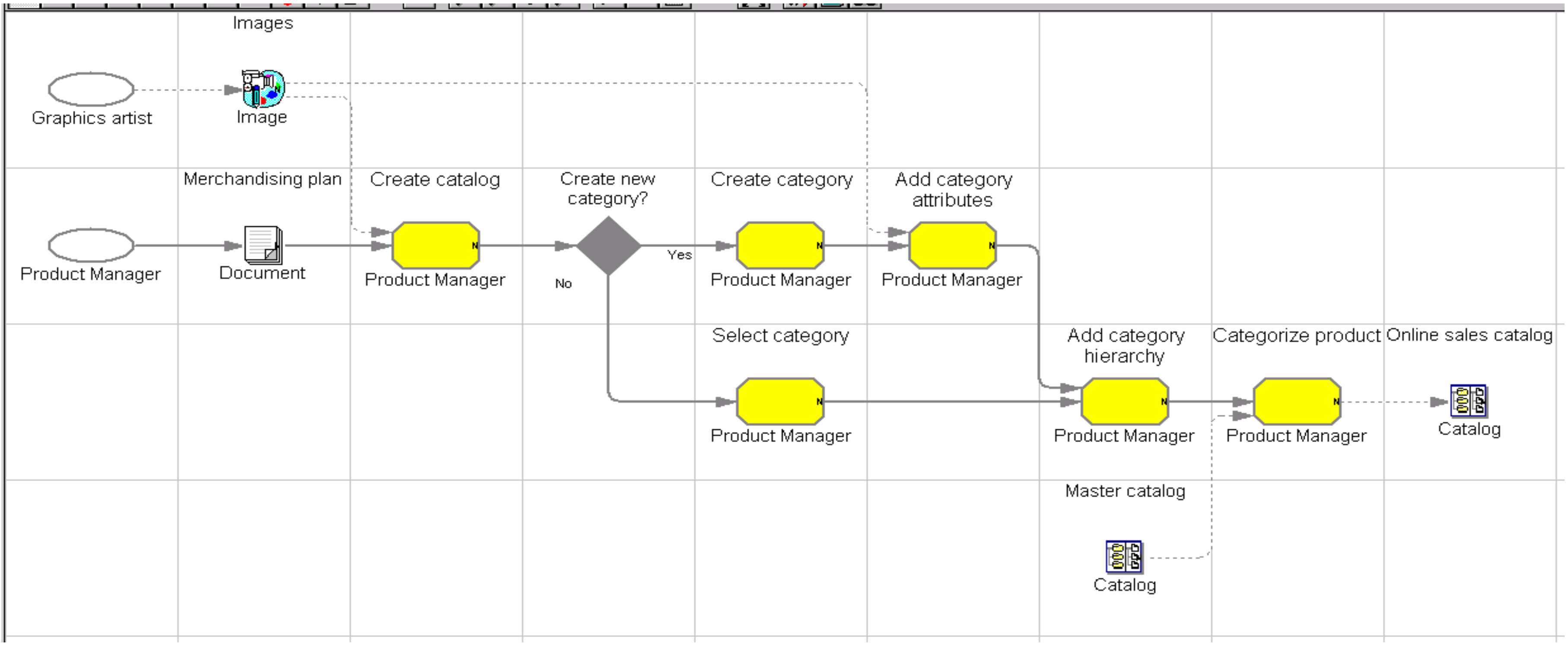
Challenges and Objectives

- Business logics and business policies are hard-coded
- Programmers must inspect the source code before making changes and updates
- Our objectives
  - To analyze control and information flows in the source code
  - To identify business logics and extract business processes

Business Logics and Processes

- ***Business logic*** is "a requirement on the *conditions* or manipulation of *data* expressed in terms of the business enterprise or application domain"
  - For example, selecting a book from a catalog, shipping the book to the customer
- ***Business policy*** specifies the rules and conditions on when and where the business logic should be executed
  - For example, if the book is in stock, ship the book to the customer
- ***Business process*** is "communication of the knowledge of business policies and business logics"

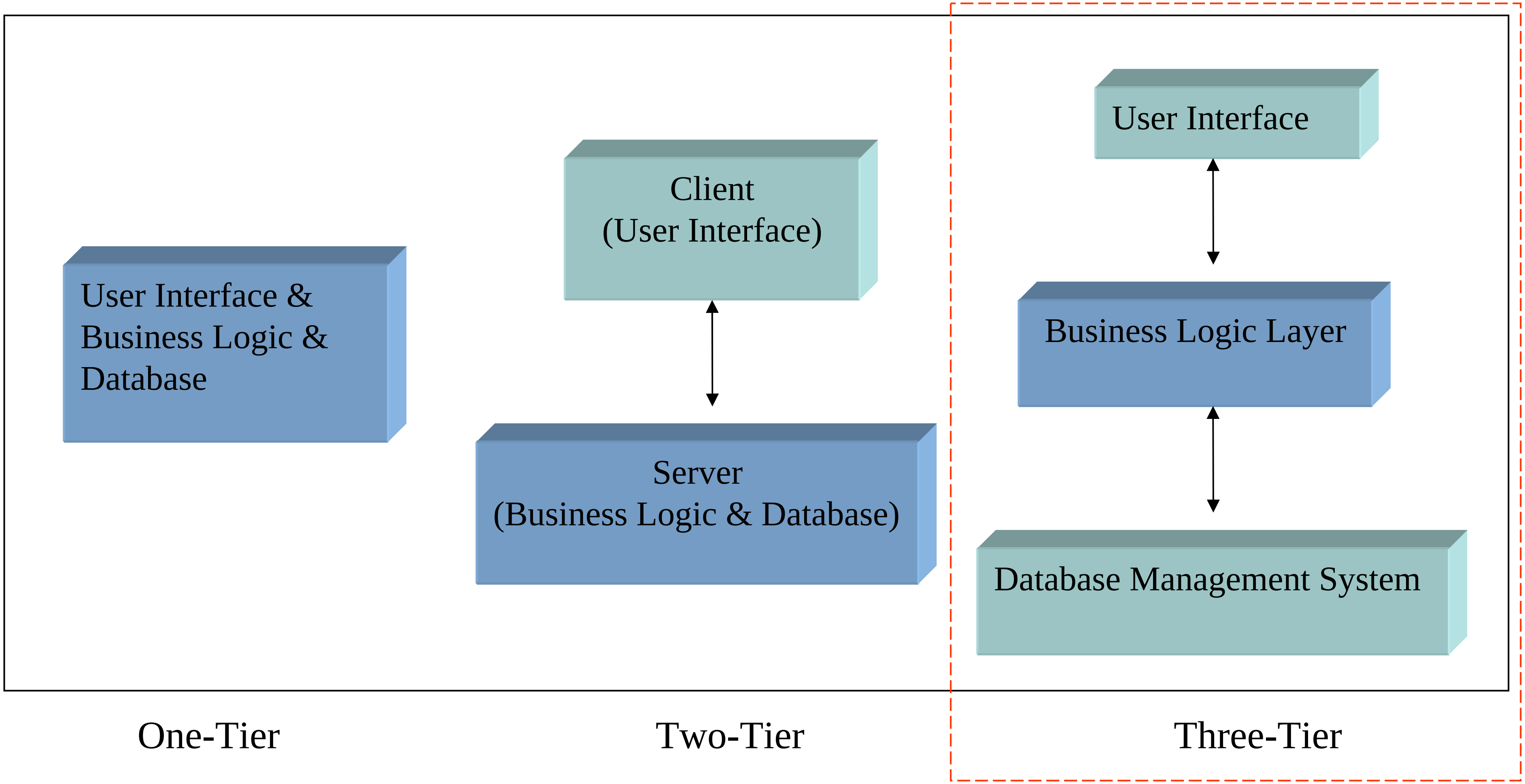
Example Workflow – Develop Sales Catalog



Business Logic Identification by Business Policies and Data

- Business logics normally take input *data* and generate output *data*
- Execution flow of business logics depends on *business policies*
- As a result, presences of business policies and data signal business logics
  - Business policies can be identified from the conditional expressions in the source code
  - Business relevant data can be identified by analyzing database operations

Architectures and Business Logics



User Interface & Business Logic & Database

Client (User Interface)

Server (Business Logic & Database)

User Interface

Business Logic Layer

Database Management System

One-Tier

Two-Tier

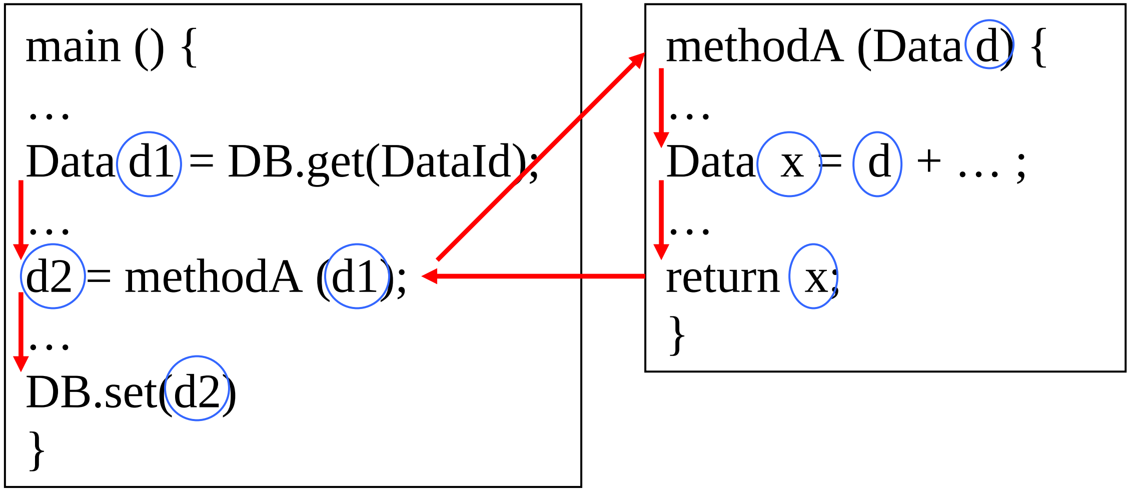
Three-Tier

Business Data

- Database Operations are explicitly defined
  - FETCH and UPDATE
- The input data of a business logic are fetched
- The output data of a business logic are updated to the database
- Once database operations are captured, we will use forward/backward tracing to locate business logics

Static Tracing (Forward)

- Analyze dataflow *toward* the same direction of the execution flow

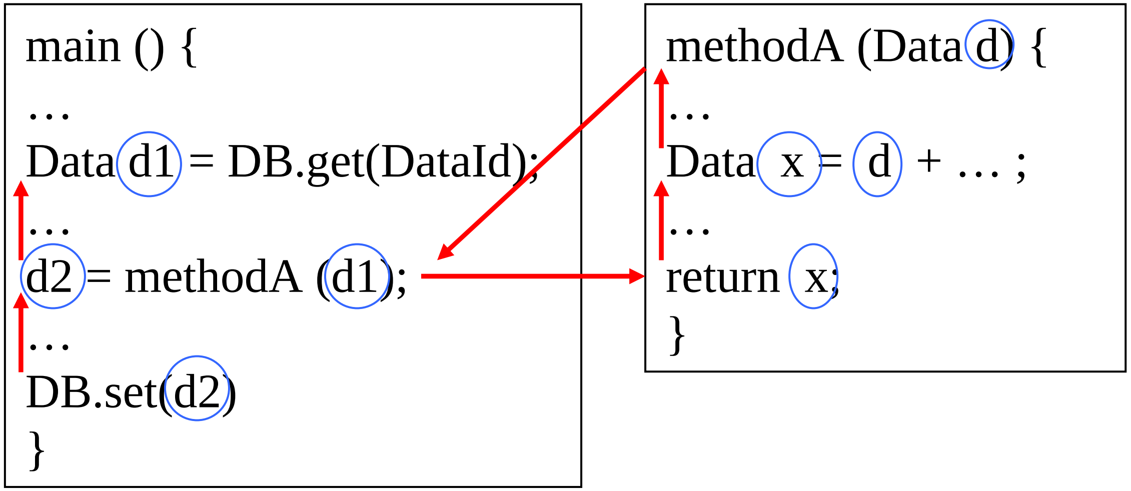


```
main () {
…
Data d1 = DB.get(DataId);
…
d2 = methodA (d1);
…
DB.set(d2)
}
```

```
methodA (Data d) {
…
Data x = d + … ;
…
return x;
}
```

Static Tracing (Backward)

- Analyze dataflow *against* the direction of the execution flow



```
main () {
…
Data d1 = DB.get(DataId);
…
d2 = methodA (d1);
…
DB.set(d2)
}
```

```
methodA (Data d) {
…
Data x = d + … ;
…
return x;
}
```

Examples of DB Operations and Business Data

Fetch Operation

```
input = DB.get(id);
if (input <= threshod){
…
}
```

```
input1 = DB.get(id1);
input2 = DB.get(id2);
if (some condition){
output = input1 + input2 + …
}
```

```
input = DB.get(id);
if (some condition){
  output = aMethod(input)
}
```

Update Operation

```
output1 = …
output2 = …
DBObject.set(output1);
DBObject.set(output2);
DBObject.commit();
```

```
if (condition1) output1=…
if (condition2) output2=…
DBObject.set(output1);
DBObject.set(output2);
DBObject.commit();
                    Or
if (condition1) DB.set(output1);
if (condition 2) DB.set(output2);
DB.commit();
```

Business Policies

- Business policy determines the execution of the business logics
- Not all conditional expressions affect the execution sequence
- We consider the following three cases:
  - Business policy specifies the *constraints* that affect the behaviors
  - Business policy specifies the *conditions* under which the computation is performed
  - Business policy specifies the *derivation of conditions* that affect the execution flow

Business Policies and Logics

- The *object with different behaviors (methods)* or *same method with different parameters* in the different branch of the same choice

```
if (condition1){
  object.action1();
} else if (condition2){
  object.action2();
}
```

```
if (condition1){
  object.action(value1);
} else if (condition2){
  object.action(value2);
}
```

Business Policies and Logics

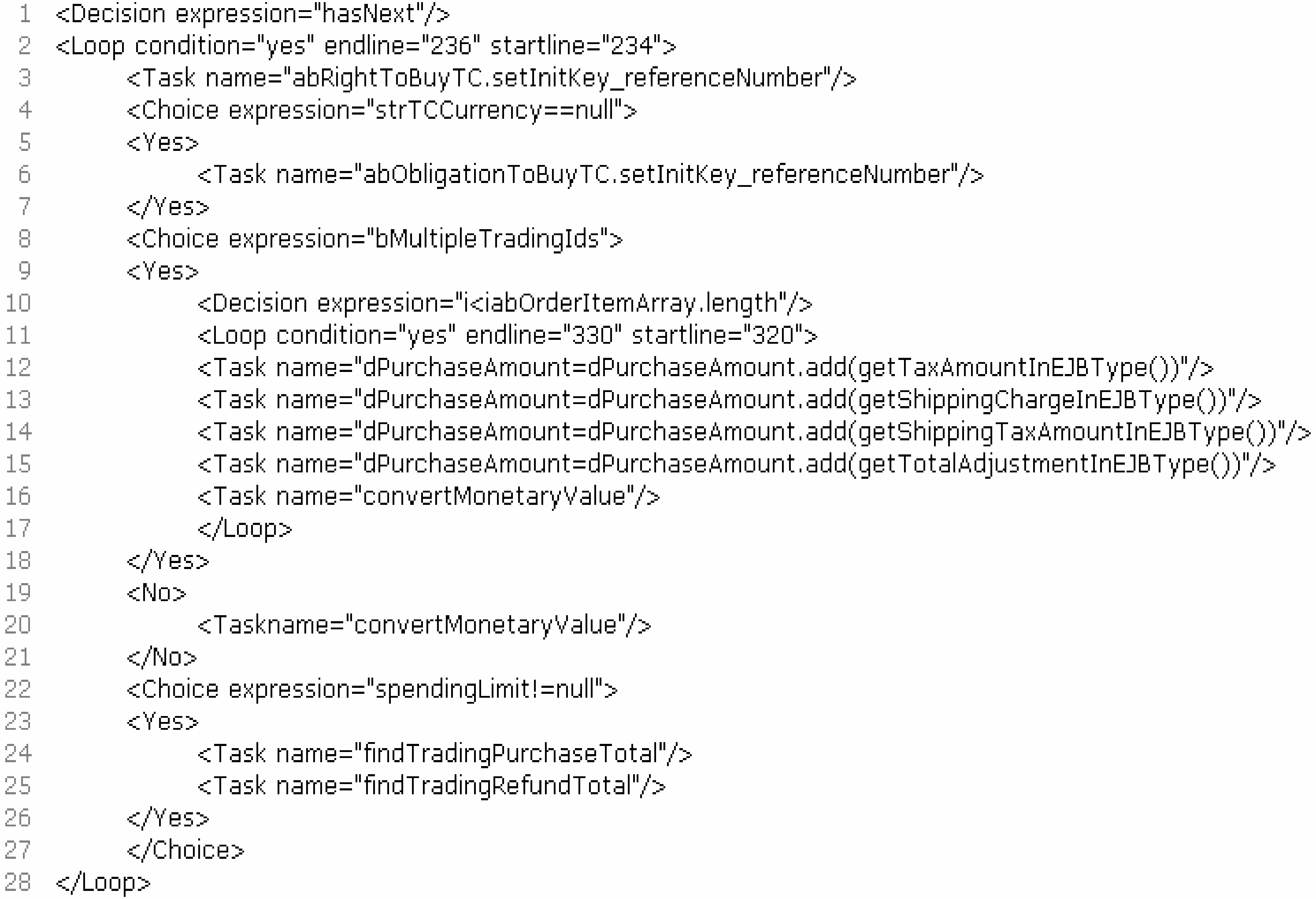
- The same *variable computed by different values* in the different branch of the same choice

```
if (condition){
 value1 = value2 + value3;
} else if (condition2){
  value1 = value4 + value5;
}
```

Business Policies and Logics

- The *condition of the choice derived* from a business data in advance

```
condition = isConditionMet (data);
if (condition){
     …
}
```



```
<Decision expression="hasNext"/>
<Loop condition="yes" endline="236" startline="234">
        <Task name="abRightToBuyTC.setInitKey_referenceNumber"/>
        <Choice expression="strTCCurrency==null">
        <Yes>
                <Task name="abObligationToBuyTC.setInitKey_referenceNumber"/>
        </Yes>
        <Choice expression="bMultipleTradingIds">
        <Yes>
                <Decision expression="i<iabOrderItemArray.length"/>
                <Loop condition="yes" endline="330" startline="320">
                <Task name="dPurchaseAmount=dPurchaseAmount.add(getTaxAmountInEJBType())"/>
                <Task name="dPurchaseAmount=dPurchaseAmount.add(getShippingChargeInEJBType())"/>
                <Task name="dPurchaseAmount=dPurchaseAmount.add(getShippingTaxAmountInEJBType())"/>
                <Task name="dPurchaseAmount=dPurchaseAmount.add(getTotalAdjustmentInEJBType())"/>
                <Task name="convertMonetaryValue"/>
                </Loop>
        </Yes>
        <No>
                <Taskname="convertMonetaryValue"/>
        </No>
        <Choice expression="spendingLimit!=null">
        <Yes>
                <Task name="findTradingPurchaseTotal"/>
                <Task name="findTradingRefundTotal"/>
        </Yes>
        </Choice>
</Loop>
```

Conclusions

- Three-tier architecture defines explicit interfaces to database management systems
- The interfaces indicate the input and output for the business logics
- Business data and policies can be identified from database operations
- Business process and logics can be extracted from data and policies

Questions?