A Framework for Document-Driven Workflow Systems

Jianrui Wang and Akhil Kumar
Smeal College of Business
The Pennsylvania State University
University Park, PA 16802, USA
{juw107, akhilkumar}@psu.edu

Outline

- Introduction – motivation and objectives
- Data Dependency Analysis
- Architecture for Document-Driven Workflows
- Document Meta-model
- Implementation
- Comparison between Document-driven and Control Flow Workflows
- Conclusion

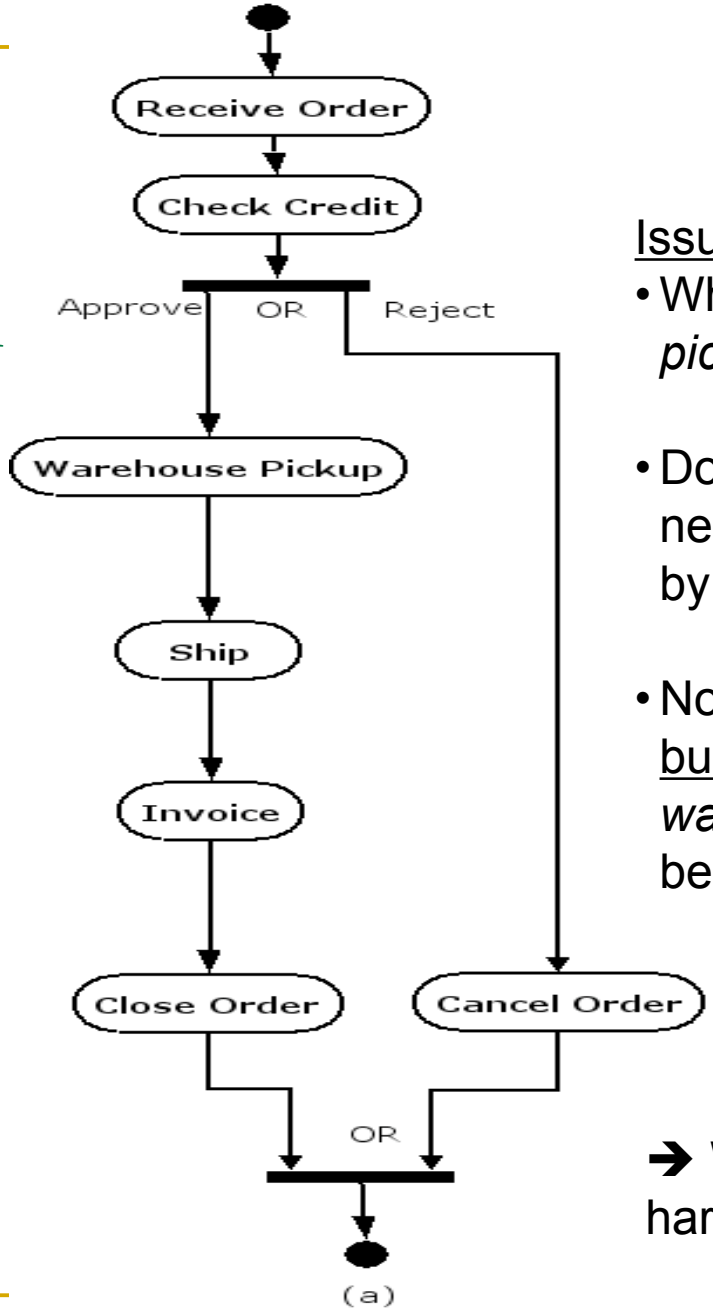
Motivation

- Different perspectives of workflow
  - Process perspective
  - **Information perspective**
  - Organization perspective
- An integrated view of workflow modeling would integrate all the dimensions into a process model
- Our work focuses more on the second perspective but includes the other two

Limitations of control flow based workflow modeling

- Two types of constraints:
  - Data dependencies (hard constraints - control flow related)
  - Business rules (soft constraints - not control flow related)
- Hence, it is difficult to explain the rationale of control flow models
- Difficult to support dynamic process changes.
- Not suitable for Ad hoc workflows.

Example workflow design based on control flow



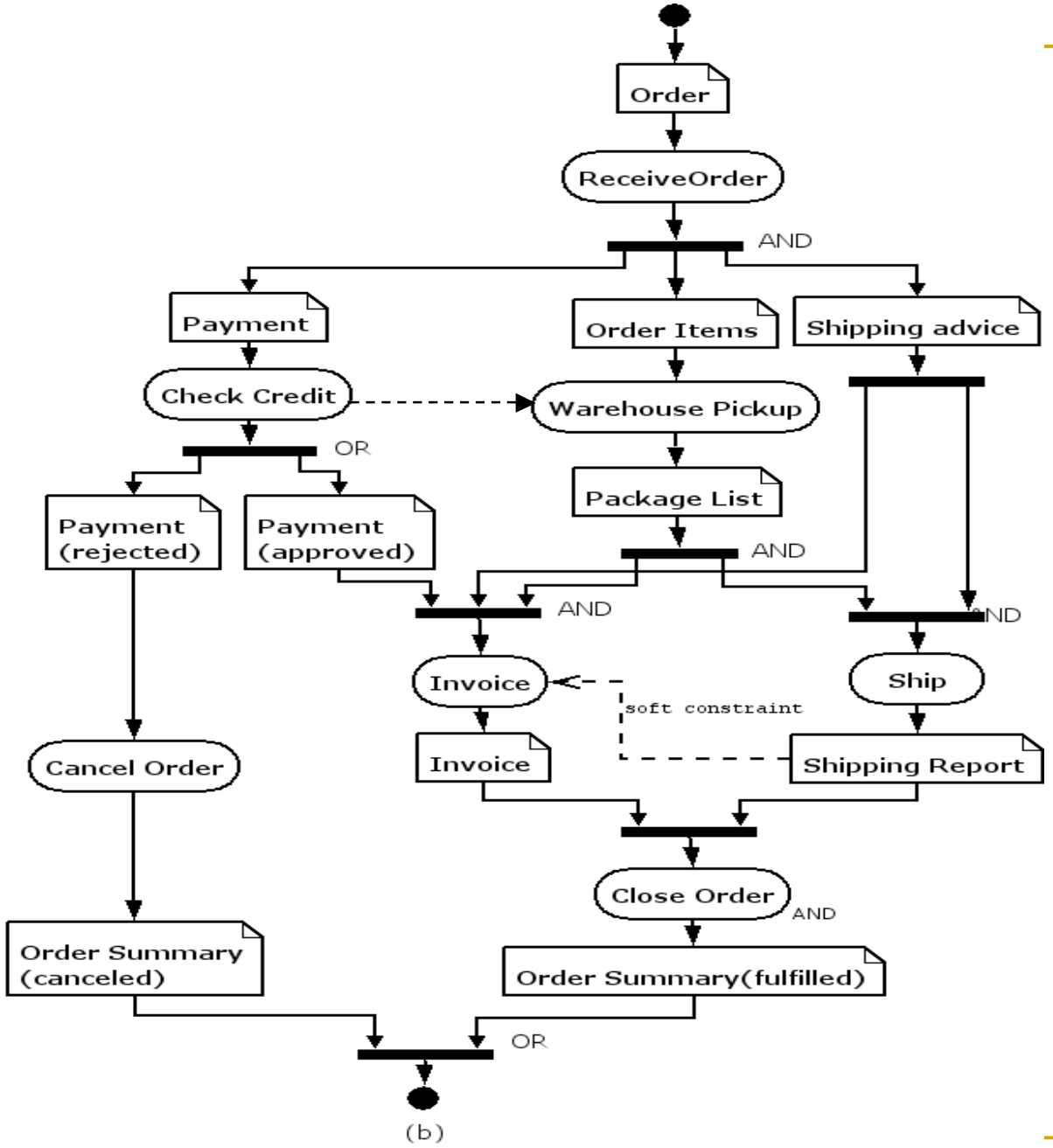
(a)

Issues:

- Why is *warehouse pickup* after *check credit*?
- Does *warehouse pickup* need any data produced by *check credit*?
- No – but there is a business rule that *warehouse pickup* must be after *check credit*.

➔ We are mixing soft and hard constraints!

Example workflow design based on data flow

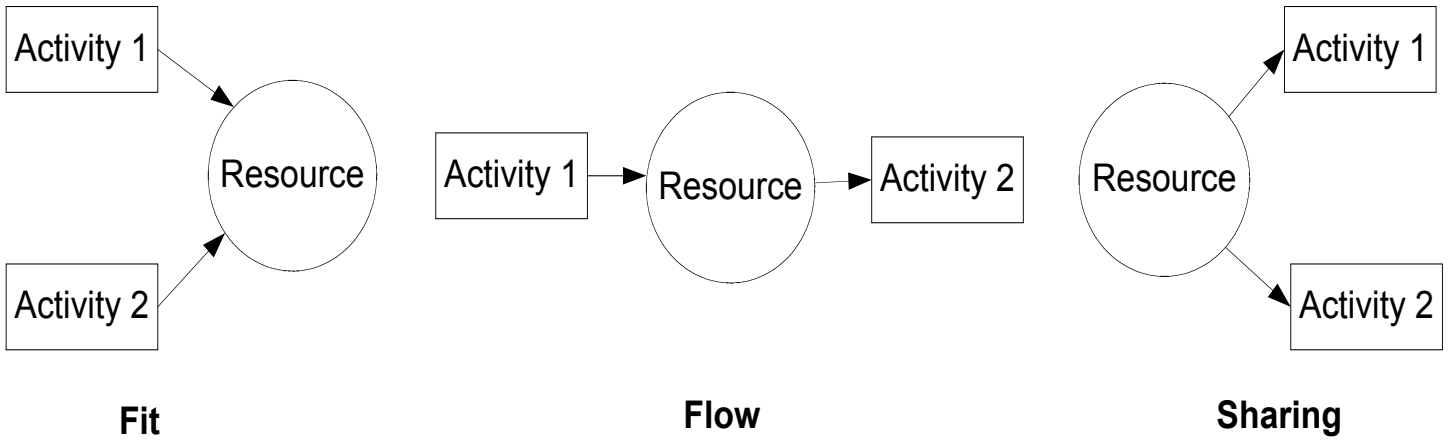


Objectives

- Integrate data flow into workflow modeling.
- Make a clear distinction between *hard* and *soft* constraints
- Develop a framework for document-driven workflow systems.
- Use document to model data flow.
- Exploit database support for workflow systems

Dependency Analysis

- Zlotkin's basic types of dependencies:
  - Fit
  - Flow
  - Sharing

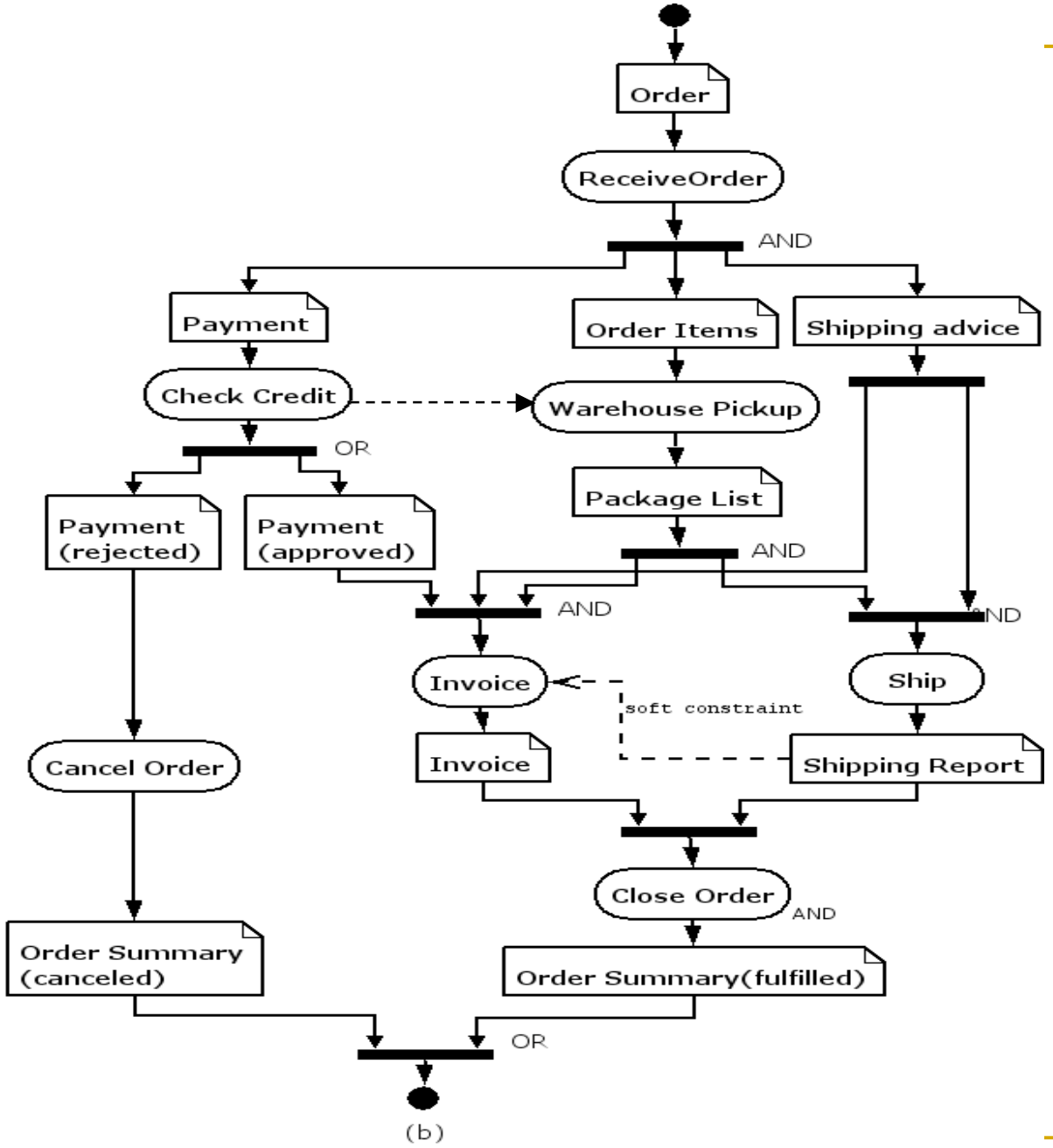


Data Dependency Analysis

- Data flow analysis

Task	Input Data	Output Data
Receive order	Order form	▪Payment info ▪Order items ▪Shipping advice
Check credit	Payment info	Credit (approved or rejected)
Warehouse pickup	Order items	Pickup list
Invoice	▪Payment info ▪Package list ▪Shipping advice	Invoice
Ship	▪Shipping advice ▪Pickup list	Proof of Shipment

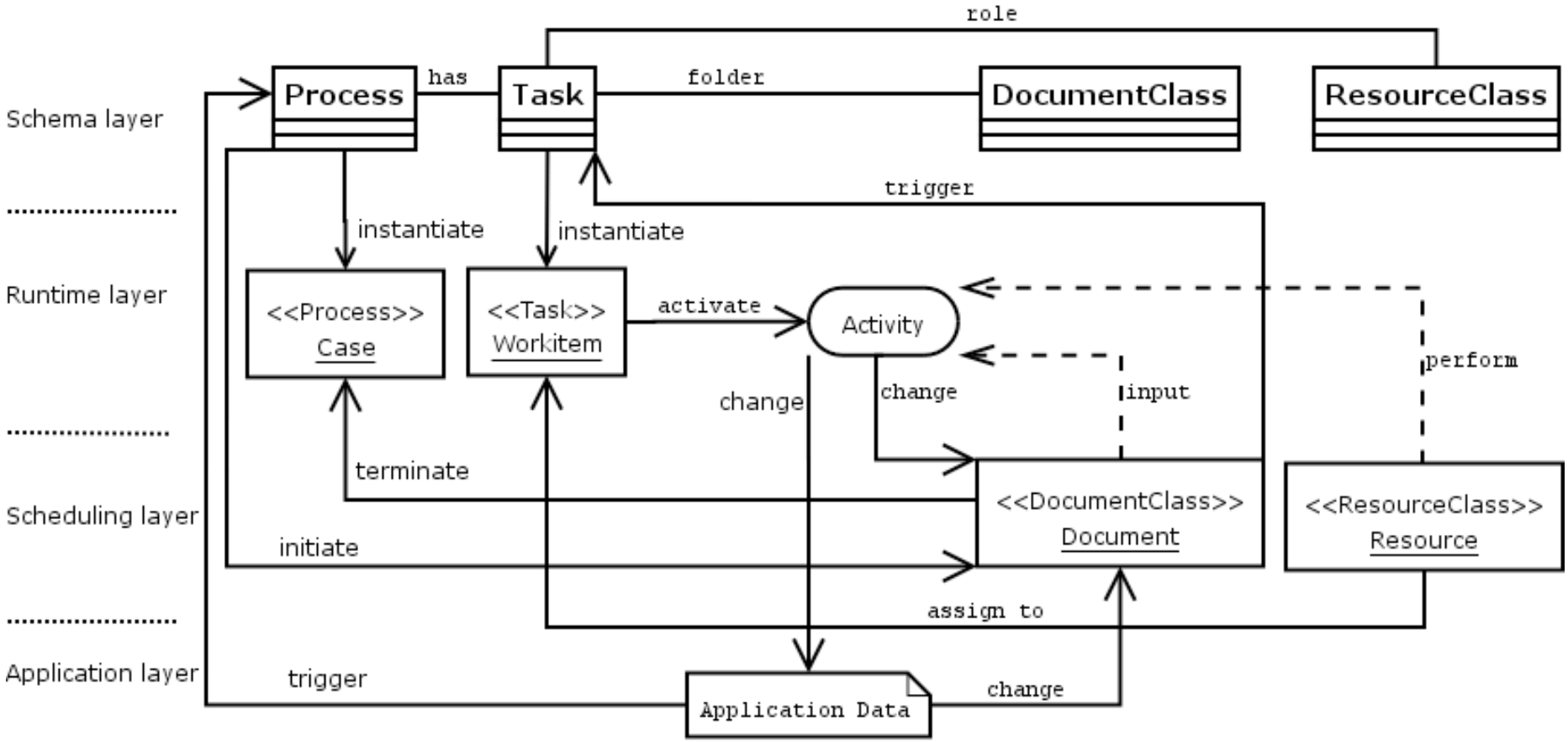
Example workflow design based on data flow



Architecture for Document-Driven Workflows

- Four-layer architecture
  - Schema Layer
    - Defines processes
  - Runtime Layer
    - How processes are started and ended
  - Scheduling Layer
    - Document and resource allocation
  - Application Layer
    - Links the workflow systems and the application data

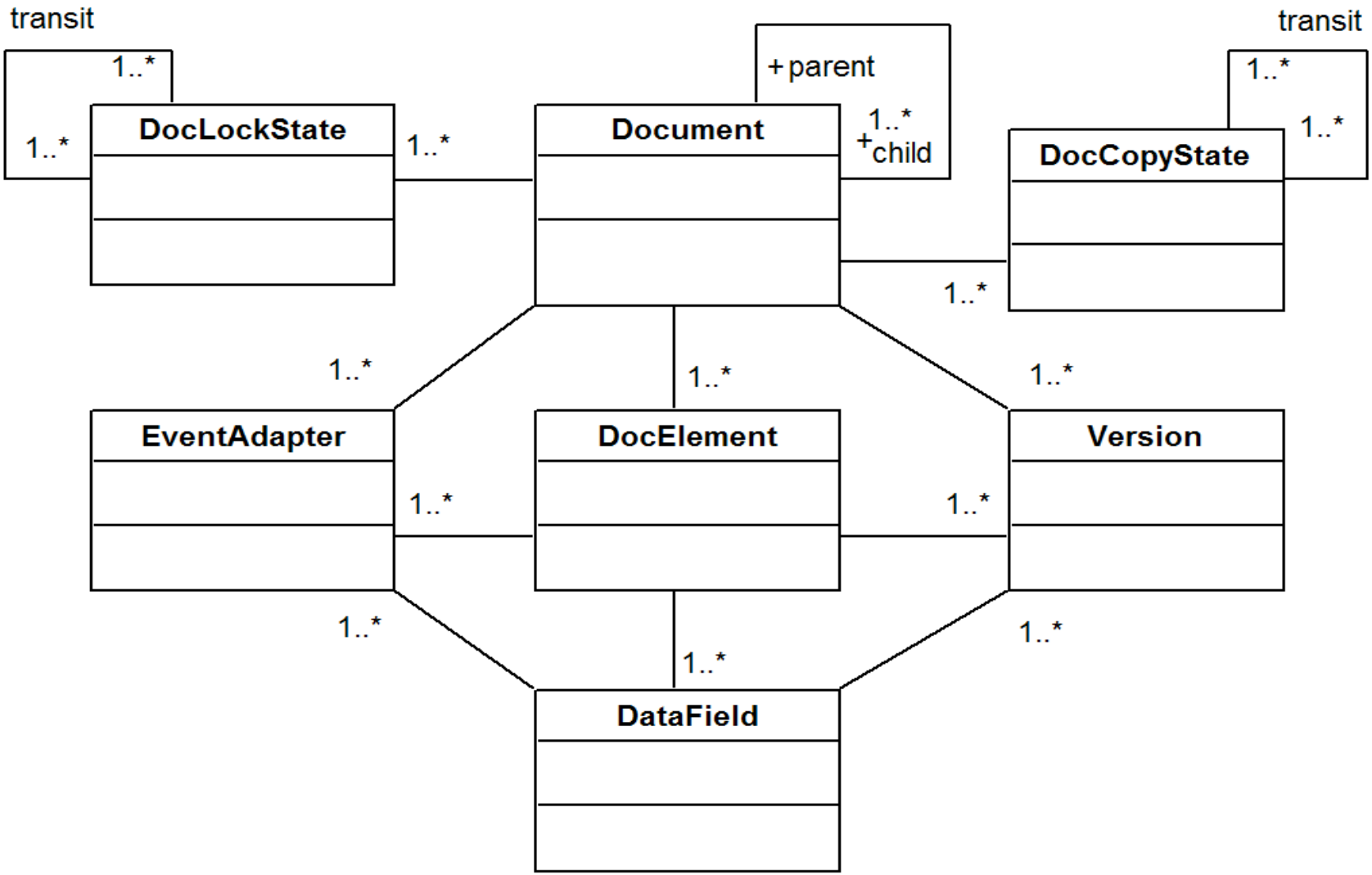
Architecture for Document-Driven Workflows



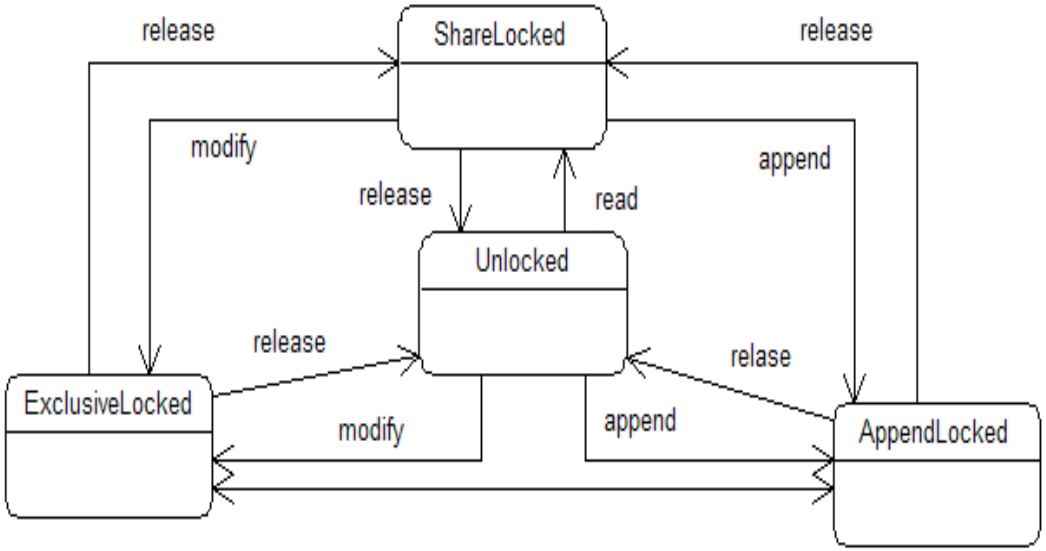
Document Meta-model

- A *Document* is a set of information *parts* which are composed together to serve a well defined business purpose.
  - A *Document* consists of several *DocElements*
  - A *DocElement* consists of several *DataFields*
- Documents have V*ersions.*
- Documents have a *DocLockState* and *DocCopyState.*
- Document changes are sent to the workflow runtime environment through *EventAdapters*.

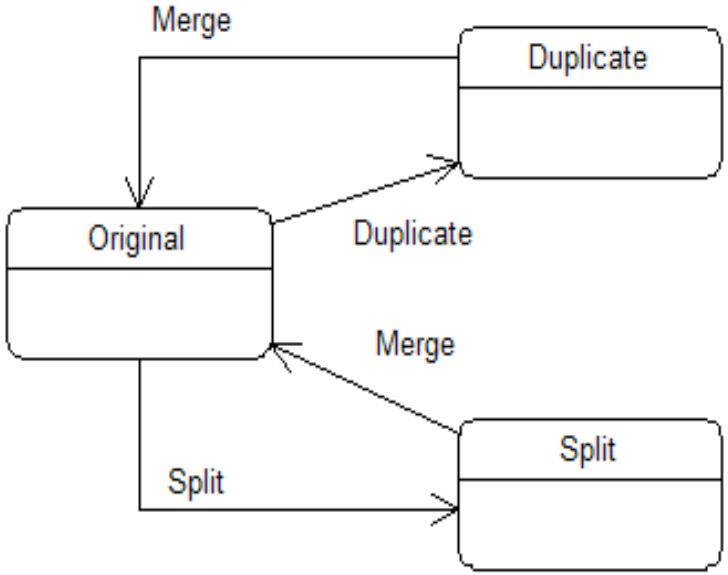
Document Meta-model (Continued)



Document Meta-model



DocLockState



DocCopyState

- DocLockState
  - suitable for *short term* locking
- DocCopyState
  - suitable for *long term* concurrent access

Document Meta-model

Example: Document state combinations that support parallel split

DocCopyState	DocLockState
Duplicate	ShareLocked
Duplicate	AppendLocked
Split	ShareLocked
Split	AppendLocked
Split	ExclusiveLocked

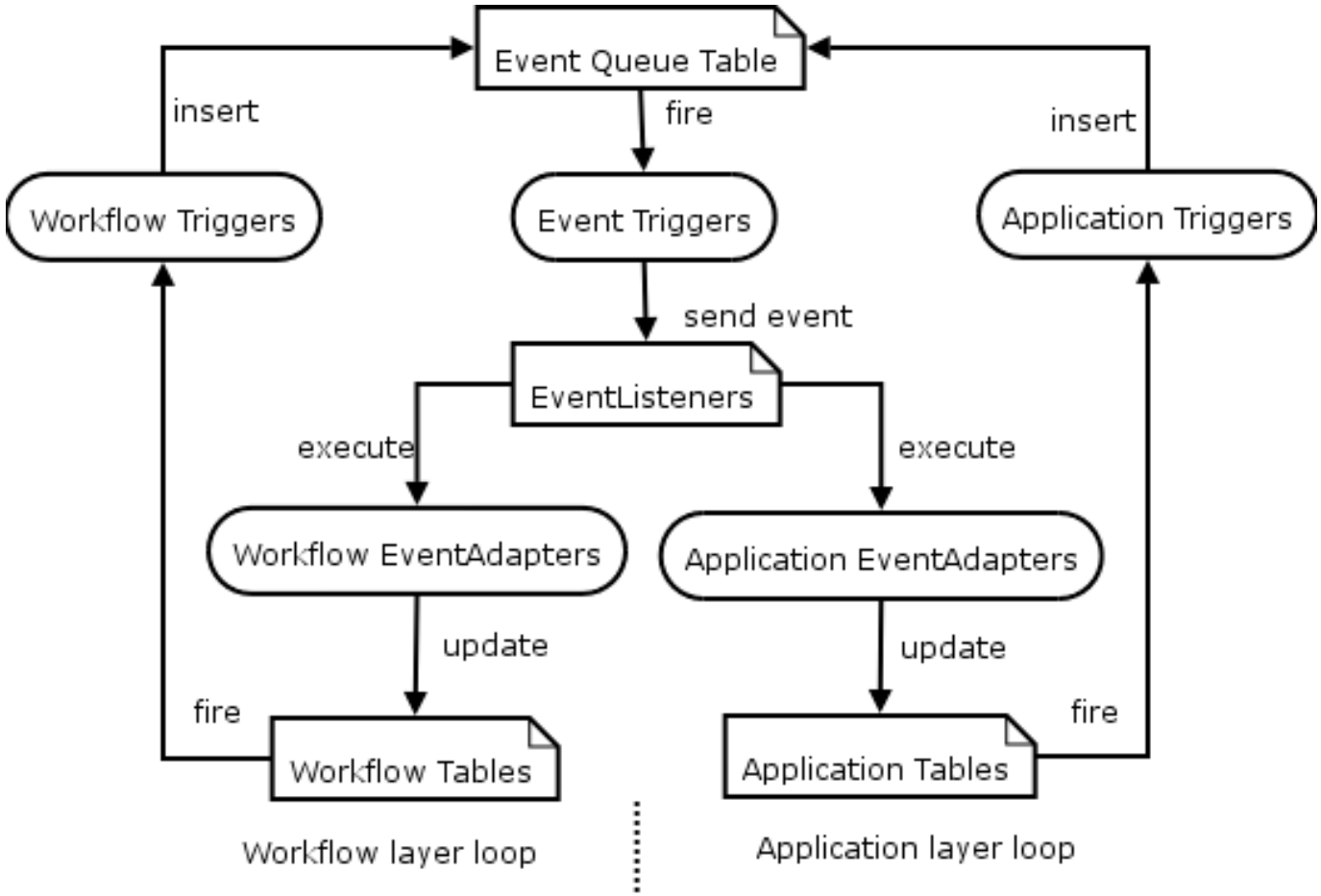
Process flow depends on document state and contents

Implementation

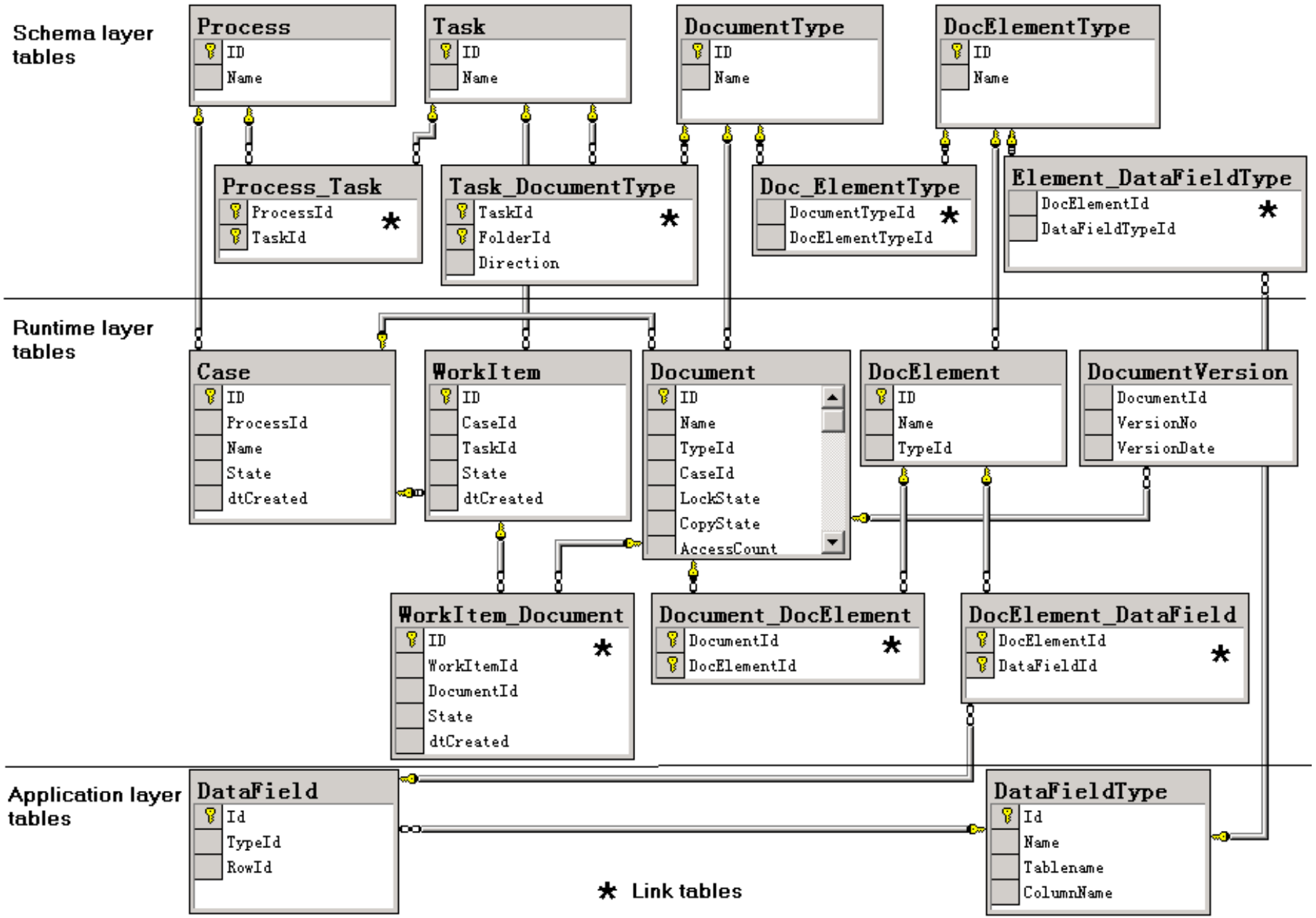
- Implementation issues – main features
  - Transact-SQL in Microsoft SQL Server
  - Driving force of workflow system:
    - Triggers
  - Communication by means of:
    - Event, EventListener, and EventAdapter.
  - Making changes to workflow system states:
    - EventAdapter
  - Mapping documents to database tables:
    - Lookup tables for linking workflow data and application data

Implementation

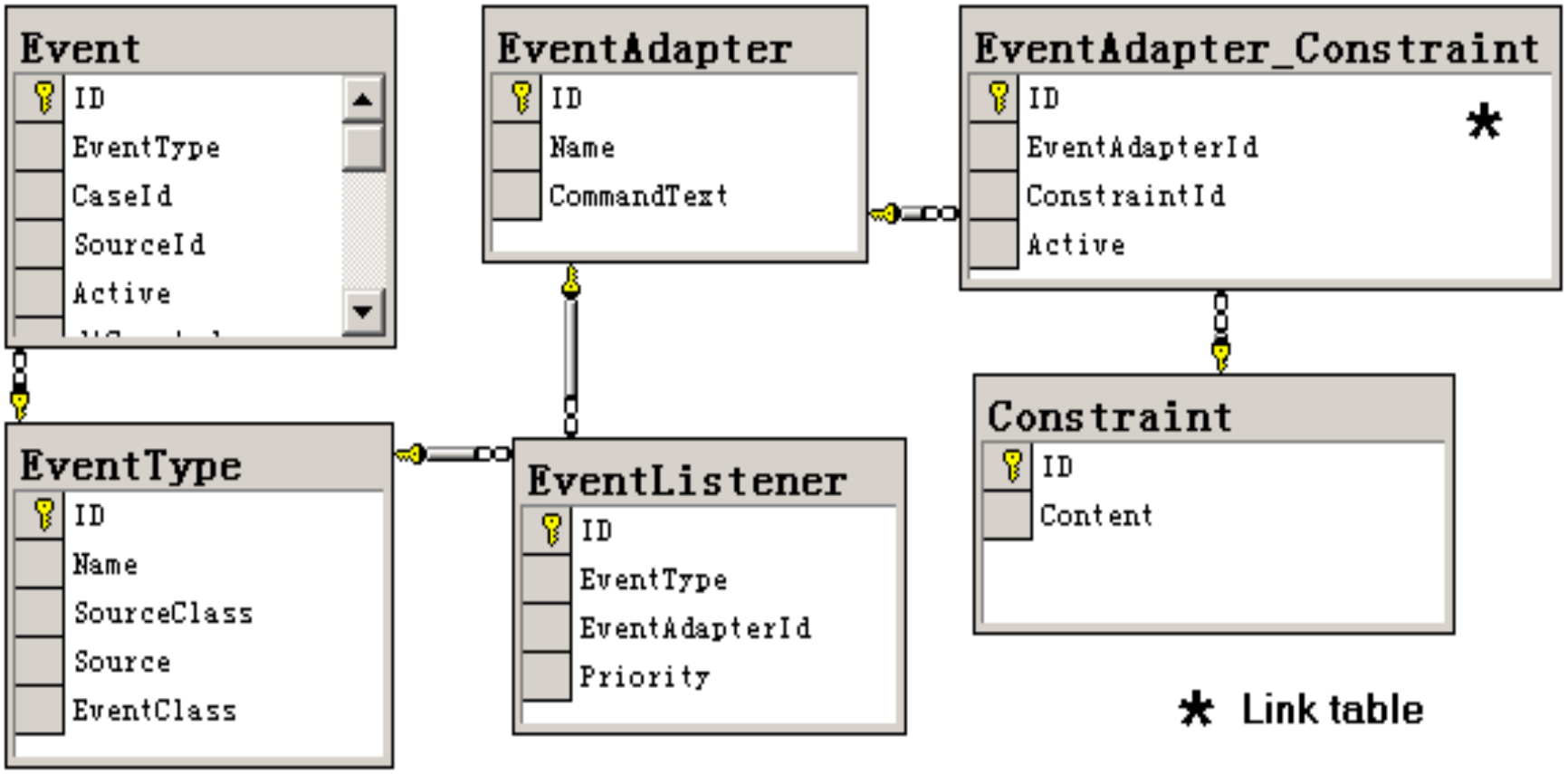
Workflow execution architecture in RDBMS



Entity Schema



Event Schema



Implementation (continued)

- Document arrival trigger

```
1  CREATE TRIGGER [onNewDocument] ON dbo.Document
2  FOR INSERT
3  AS
4  BEGIN
5      DECLARE @typeId int
6      DECLARE @docId int
7      DECLARE @caseId int
8      DECLARE @eventType int
9      /* retrieve document context */
10     SELECT @docId=[ID],@typeId=TypeId,@caseId=CaseId FROM inserted
11     /* lookup associated event type*/
12     SELECT @eventType=[ID] FROM dbo.EventType
13         WHERE SourceClass='Document' AND Source=@typeId AND EventClass='New'
14     /* generate event */
15     INSERT dbo.Event (EventType,CaseId,SourceId,dtCreated)
16         VALUES (@eventType,@caseId,@docId,getdate())
17 END
```

Implementation (continued)

- New Event Trigger

```
1  CREATE  TRIGGER [onNewEvent] ON dbo.Event
2  FOR INSERT
3  AS
4  BEGIN
5      DECLARE @eventType int
6      DECLARE @eventId int
7      DECLARE @commandText varchar(8000)
8      /* retrieve event context */
9      SELECT @eventId=ID, @eventType=eventType FROM inserted
10     /* retrieve all adapters listen to the event */
11     DECLARE cur_adapters CURSOR LOCAL For
12     SELECT dbo.EventAdapter.CommandText
13         FROM dbo.EventListener INNER JOIN
14             dbo.EventAdapter ON dbo.EventListener.EventAdapterId = dbo.EventAdapter.ID
15         WHERE (dbo.EventListener.EventType = @eventType)
16         ORDER BY dbo.EventListener.Priority
17     OPEN cur_adapters
18     FETCH NEXT FROM cur_adapters INTO @commandText
19     WHILE @@fetch_status=0
20     BEGIN
21         /* get SQL statement and attach input parameter */
22         SET @commandText=@commandText+ ' ' + cast(@eventId as varchar(50))
23         /* run SQL (event action)*/
24         EXEC (@commandText)
25         /* next adapter */
26         FETCH NEXT FROM cur_adapters INTO @commandText
27     END
28     CLOSE cur_adapters
29     DEALLOCATE cur_adapters
30 END
```

Sample Process Description File

```
<?xml version="1.0" encoding="utf-8" ?>
- <process name="Express Order Process">
  - <interfaces>
    - <interface>
      - <events>
          <event name="Order Arrives" sourceClass="External" source="Order" />
        </events>
      - <eventAdapters>
        - <eventAdapter sourceClass="External" sourceId="Order" mode="listen" action="dbo.wfEAOrderArrives">
          - <constraints>
              <constraint name="IsExpressOrder" constraintText="dbo.isExpressOrder" />
            </constraints>
          </eventAdapter>
        </eventAdapters>
      </interface>
    ...
  </interfaces>
```

Sample Process Description File (contd.)

```
- <documents>
  - <document name="Order Form">
    - <events>
        <event name="Document New" eventClass="New" />
        <event name="Document Updated" eventClass="Update" />
      </events>
    - <docElement>
        <dataField name="Cusomer ID" tableName="Customers" columnName="ID" />
        ...
      </docElement>
      ...
    </document>
    ...
  </documents>
```

Sample Process Description File (contd.)

```
- <tasks>
  - <task name="Receive Order">
    - <inputDocument>
        <item document="Order Form" />
      </inputDocument>
    - <outputDocument>
        <item document="Payment" />
        <item document="Pickup List" />
        <item document="Shipping Advice" />
      </outputDocument>
    - <eventAdapters>
        <eventAdapter sourceType="Document" sourceId="Order"
      </eventAdapters>
    </task>
    ...
  </tasks>
</process>
```

Document-driven Workflow vs. Control Flow Workflow

Document-Driven Workflow	**Control flow-Driven Workflow**
Process is driven by the documents.	Process is driven by the control flow.
The process is very flexible and can be changed instantly through changing constraints	It is difficult to change a control flow of an instance because all the instances share the same control flow pattern.
There are no fork/join design issues since there is control flow.	There are fork/join design issues to be addressed.
Application Data is separated from the control flow.	In most case, the application data are attached to the control flow.
Suited for ad hoc workflow.	Good for production workflow with mature processes and a large number of tasks.
Verification is relatively easy.	Verification could be hard
Have to deal with conflict resolution	No need for conflict resolution
Difficult to visualize the process	Process can be visualized easily

Related research

- ECA (McCarty, Dayal, 1989)
- WIDE project –triggers to handle exceptions (Grefen, 1999)
- Adding active properties to documents (Dourish, 2000)
- XDoc-WFMS (Krishnan, et al. 2002)
- Doc-centric approach to BPM (Mazumdar, Abusafiya, 2004)
- Zur Muehlen, 2003, Resource modeling

Conclusions

- Noted the importance of separating hard and soft constraints in workflow modeling.
- Document-driven workflow integrates dataflow into process model.
- Helps in designing more efficient processes by considering data dependencies.
- Illustrated an implementation inside RDBMS.
- Suitable for Ad hoc workflows.