Business-Driven Software Engineering
Lecture 9 – Business Objects and Object Life Cycles

Jochen Küster jku@zurich.ibm.com
Agenda

- Introduction to Business Objects and Object Life Cycles
- Overview of Use Cases for Object Life Cycles
- Consistency of Business Process Models and Object Life Cycles
- Generation of a Business Process Model from Object Life Cycles
- Extraction of Object Life Cycles
- Resolution of Inconsistencies
- Summary and References
Business Process Models and Object Life Cycles
Business Objects

- A *business object* is a discrete entity that plays a role in business processes of an organization
- Examples: Claim, Bill, Contract

- Objects can be associated with a number of distinct business object states

- Modeling of business objects
  - UML class diagrams for capturing structure of business objects
  - UML state diagrams for capturing dynamics of business objects
Object life cycles model allowed state transitions.

No events/conditions/actions as in conventional UML State Diagrams.

Instead event is replaced by the activity which performs the transition on the object.

Object life cycle for Claim object type:
- register
- registered
- grant
- rejected
- close
- settle
- settled
- closed

(UML State Diagram showing transitions and states for Claim object type)
Business process models capture coordination of activities performed to achieve a business goal.

Activities work on data, they consume data and produce new data.

In some occasions it is necessary to model data more explicitly.

Some process modeling languages support business objects:
- BPMN, UML 2.0
Business Objects in BPMN

- BPMN supports the modeling of object flow
- Business objects define input and output of activities
- Business objects can be given states
- Decoupled and connected option
- Too much object flow introduces cluttering into the diagram
Business Objects in UML 2.0

- UML 2.0 supports the modeling of object flow

- Explicit form of object flow:
  - Object flow connections
  - Object pins are used for outputting or reading a business object

- Implicit form of object flow (shorthand for this lecture):
  - Datastores are used
  - Connections between pins and datastores are omitted
Business Process Models and Business Object Flow

- Object pins and data stores support the modeling of business object flow
- Decision conditions can be specified in terms of the state of business objects (see Data-based Routing Pattern)
Overview of Use Cases for Object Life Cycles
Use Cases of Object Life Cycle Modeling (1)

1. Model both Business Process Models as well as Object Life Cycles
   – To get an overview of how business objects are manipulated
   – To complete the picture of the business process (another view)

2. Establish consistency between Business Process Models and Object Life Cycles
   – To improve the quality of the overall model
   – To ensure consistency with reference object life cycles
Use Cases of Object Life Cycle Modeling (2)

3. Generate a process model from existing object life cycles
   – To understand better the hidden process behind object life cycles
   – To follow a process-driven approach for implementation
   – To ensure consistency of object life cycles with generated process model

4. Extract object life cycles from business process models
   – To get an overview of how business objects are manipulated
   – To ensure consistency of process model with extracted life cycles
   – To follow an object-based approach for implementing processes
Techniques for Process Models and Object Life Cycles

- A consistency concept for the two views (for checking consistency)
- Model transformations for generation and extraction
- Resolution of inconsistencies by predefined transformations
Consistency of Business Process Models and Object Life Cycles
Relationship between process models and object life cycles

- Process models and object life cycles represent **overlapping behavior**
Checking Consistency - Example

Object life cycle for Claim object type

Process model for Claim Handling process

- Creates Claim objects in state registered
- Induces transitions from registered to granted and from registered to rejected for Claim objects
- Creates Payment objects in state paid in full
- Claim can be in state closed or rejected and Payment can be in state paid in full
- Does not induce transition from cancelled to rejected for Claim objects
Checking Consistency – Forms of Consistency

- Conformance: all induced transitions, first states and last states in process model have corresponding elements in object life cycle

- Coverage: all transitions, target states of initial transitions and final states in object life cycle must have corresponding elements in process model
Examples of possible inconsistencies

- **Non-conformant transitions:** (1) offer benefit task induces transition from state granted to rejected and (2) close claim task induces transition from state granted to state closed, which are not defined in the object life cycle.

- **Non-conformant last state:** (3) rejected is a last state in the process model, but it is not a final state in the object life cycle.

- **Non-covered transitions:** (4) transitions from state granted to settled and (5) from state settled to closed are not induced in the process model.
Approaches to achieving consistency

1. Consistency checking
   - Object life cycles
   - Existing business process model
   - Inconsistencies
   - Consistency ensured

2. Resolution of inconsistencies
   - Object life cycles
   - Inconsistencies
   - Consistency ensured

3. Consistency Checking
   - Initial business process model
   - Customized business process model

4. Resolution of inconsistencies
   - Object life cycles
   - Inconsistencies
   - Consistency ensured

1. Process model generation
2. Customization
3. Consistency Checking
4. Resolution of inconsistencies
Generation of a Business Process Model from Object Life Cycles
Generation from one object life cycle

1. For each event labeling a state transition in object life cycle, an activity is generated with appropriate input/output pins and object states.
Generation from one object life cycle

1. For each event labeling a state transition in object life cycle, an activity is generated with appropriate input/output pins and object states

2. Order of activities is based on matching input/output object states
Generation from one object life cycle

1. For each event labeling a state transition in object life cycle, an activity is generated with appropriate input/output pins and object states.

2. Order of activities is based on matching input/output object states.

3. Control nodes are added for correct control flow.
Generation from one object life cycle

1. For each event labeling a state transition in object life cycle, an activity is generated with appropriate input/output pins and object states
2. Order of activities is based on matching input/output object states
3. Control nodes are added for correct control flow
4. Combination of selected activities
Generation from a set of object life cycles

1. Identification of synchronization events (manual step)

2. Composition of object life cycles

3. Process model generation:
   - Task generation
   - Object state relation for tasks
   - Process fragment generation
   - Connection of process fragments
Identification of synchronization events

- A synchronization event is an event that triggers state transitions in more than one object life cycle.

- Identifying synchronization events is necessary given several object life cycles, to ensure that invalid composite states cannot be reached.
Composition of object life cycles
Transition and first state conformance with respect to both object life cycles are satisfied, but last state conformance is not.

All coverage conditions are satisfied here, but this is not guaranteed.
Extraction of Object Life Cycles
Example for Extraction of Life Cycles

- **Register new claim**
- **Check for fraud**
- **Initiate fraud investigation**
- **Prepare settlement**
- **Carry out payment**
- **Close**

- **Evaluate**
- **Prepare settlement**
- **Settlement [Requested]**
- **Settlement [Settled]**

- **Notify rejection**
- **Claim [Rejected]**
- **Claim [PreRejected]**
- **Claim [PreRejected]**
- **Claim [Requested]**

- **Claim [Granted]**
- **Claim [Rejected]**
- **Claim [Settled]**

- **Settlement [PreRejected]**
- **Settlement [Settled]**

- **Claim [Registered]**
- **Claim [NotFraudulent]**
- **Claim [Fraudulent]**
- **Claim [NotFraudulent]**
- **Claim [NotFraudulent, NeedsReevaluation]**

- **Prepare for reevaluation**
- **Close [Closed]**
- **Claim [Settled]**
- **Claim [Rejected]**
- **Claim [PreRejected]**

- **Claim [Fraudulent]**
- **Claim [NotFraudulent]**
- **Claim [NotFraudulent, NeedsReevaluation]**
- **Claim [NeedsReevaluation]**
Transformation Rules for Generation

<table>
<thead>
<tr>
<th>Process model $P$</th>
<th>Object life cycle $OLC_P$ for $O$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rule 1 (objectCreation)</strong></td>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Rule 2 (stateChange)</strong></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Rule 3 (finalConsumption)</strong></td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Rule 4 (finalNode)</strong></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Rule 5 (processInput)</strong></td>
<td><img src="image5.png" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Rule 6 (processOutput)</strong></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Generating object life cycles from Claims handling process

Rule 1 (objectCreation)

- Register new claim
- Check for fraud
  - Claim [Fraudulent, NotFraudulent]
- Prepare settlement
  - Settlement [Requested]
  - Claim [Granted]
- Carry out payment
- Initiate fraud investigation
Generating object life cycles from Claims handling process

Rule 2 (stateChange)
Generating object life cycles from Claims handling process

Rule 3 (finalConsumption)
Generated Object Lifecycles

Settlement

START$_P$

- Requested
  - Prepare settlement
- Authorized
  - Carry out payment
- Settled

Claim

- Register new claim
- Check for fraud
- Registered
  - Check for fraud
  - Fraudulent
    - Initiate fraud investigation
    - Evaluate
  - NotFraudulent
    - Evaluate
  - PreRejected
    - Notify rejection
    - Evaluate
  - Rejected
    - Evaluate
    - Closed
  - NeedsReevaluation
    - Evaluate
    - Notify rejection
    - Evaluate
    - Close
  - Settled
    - Evaluate
    - Closed
  - Granted
    - Evaluate
    - Notify rejection
    - Evaluate
    - Close
  - Settled
    - Evaluate
    - Close
  - Closed

Resolution of Inconsistencies
Examples of Inconsistencies

Non-conformant transitions:
- transition from “granted” to “needs review” \((\text{ncnf\_tran(settle, granted, needs review)})\)
- transition from “needs review” to “reviewed” \((\text{ncnf\_tran(review, needs review, reviewed)})\)

Non-covered transitions:
- transition from “rejected” to “closed” \((\text{ncov\_tran(rejected, closed)})\)
Inconsistency resolutions may be captured as resolution rules.

- Here: inconsistency resolutions for non-conformant transition.
- Application of the resolution rule leads to resolution of the inconsistency.
- But: resolutions may introduce new inconsistencies as side effects and resolve other inconsistencies as side effects.
Inconsistency Resolution Rules and Side Effects

Potential side effect of resolution r1:
- introduction of a new non-covered transition (if the transition in the business process provided coverage for a transition in the object life cycle)

Concrete side effect of resolution r1:
- ncov_tran(granted, settled)

side effects can only be determined precisely for a given concrete model
Object Life Cycles and Business Process Models - Tools

- Object life cycles introduce a complementary view
- Consistency management with business process models is required

- Ideal solution:
  - Process Modeling Environment supports both views and offers support for consistency management

- Suboptimal solution:
  - Models are created in different Modeling Environments, no consistency management, unclear how they are related
Summary of Lecture and References

- Introduction to Object Life Cycles and their relationship to Process Models
- Consistency Checking of Process Model and Object Life Cycle
- Generation of Object Life Cycles from Process Models
- Generation of Process Model from a set of Object Life Cycles

Further Reading:
