Operation Contracts

by

Mauricio Monsalve
Design road

- We have described:
  - Use Cases
  - Domain Model
  - System Sequence Diagrams

- We now describe Operation Contracts

- Afterwards, we go into the Design Model
This is where we stand right now
Operation Contracts

- **Use Cases** often fully describe the behavior of a system
- But they may not be enough
- **Operation Contracts** describe how the internal state of the concepts in the **Domain Model** may change
- Operation Contracts are described in terms of preconditions and postconditions
# Operation Contracts

## Contract CO2: enterItem

<table>
<thead>
<tr>
<th>Operation:</th>
<th>enterItem(itemId: ItemID, quantity: integer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross References:</td>
<td>Use Cases: Process Sale</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>There is a sale underway.</td>
</tr>
<tr>
<td>Postconditions:</td>
<td>– A SalesLineItem instance sli was created <em>(instance creation).</em></td>
</tr>
<tr>
<td></td>
<td>– sli was associated with the current Sale <em>(association formed).</em></td>
</tr>
<tr>
<td></td>
<td>– sli.quantity became quantity <em>(attribute modification).</em></td>
</tr>
<tr>
<td></td>
<td>– sli was associated with a ProductDescription, based on itemId match <em>(association formed).</em></td>
</tr>
</tbody>
</table>
Operation Contracts

- Operation Contracts are defined in terms of system operations
  - Operations (say, *methods*) that the system offers as a *whole*
    - The system is still a *black box* at this stage

- The System Sequence Diagrams show system events
  - I.e., the SSD's *messages*

- System operations *handle* system events
Writing Operation Contracts

Process Sale Scenario

: Cashier

makeNewSale()

loop

[ more items ]

enterItem(itemId, quantity)

description, total

endSale()

total with taxes

makePayment(amount)

change due, receipt

: System

these input system events invoke system operations

the system event enterItem invokes a system operation called enterItem and so forth

this is the same as in object-oriented programming when we say the message foo invokes the method (handling operation) foo
Writing Operation Contracts

Contract CO2: enterItem

<table>
<thead>
<tr>
<th>Operation:</th>
<th>enterItem(itemID: ItemID, quantity: integer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross References:</td>
<td>Use Cases: Process Sale</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>There is a sale underway.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Postconditions:</th>
<th>A SalesLineItem instance sli was created <em>(instance creation).</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sli was associated with the current Sale <em>(association formed).</em></td>
</tr>
<tr>
<td></td>
<td>sli.quantity became quantity <em>(attribute modification).</em></td>
</tr>
<tr>
<td></td>
<td>sli was associated with a ProductDescription, based on</td>
</tr>
<tr>
<td></td>
<td>itemID match <em>(association formed).</em></td>
</tr>
</tbody>
</table>
Writing Operation Contracts

- **Operation**—name of operation (parameters)
- **Cross Reference**—the Use Cases in which the OC occurs
- **Preconditions**—noteworthy assumptions about state of system or objects in DM before execution
- **Postconditions**—state of objects in DM after execution of operation
Postconditions

- Most important part of OCs!
- Include changes in state of DM
- Book uses categories (note that the names are for reference only):
  - Instance creation or deletion
  - Association formed or broken
  - Attribute modification
Postconditions

- Write in past (passive voice?)
  - A LineSaleItem was created

- Readability first

- Common mistake—forgetting that instance creation often implies association formation, and similarly, that instance deletion often implies association breaking
Practical summary

- Identify system operations from the SSDs
- Identify *subtle* or *complex* system operations
- Construct a contract for each of the above; for postconditions, use the following categories
  - *Instance created-deleted*
  - *Attribute modified*
  - *Association formed-broken*
Credits

Notes and figures adapted from