Object-Oriented Software Development
Spring 2015

System Sequence Diagrams

by
Mauricio Monsalve
Design road

- We have described the Domain Model
- We now describe the System Sequence Diagram(s)
- Based on both of the above, we will create the Design Model
Design road

A more detailed design road for the Unified Process

Question: can you explain this layout?
System Sequence Diagrams

• We revisit the **Use Cases**
• The **Domain Model** did not model how Use Cases unfold with the system
• **System Sequence Diagrams (SSD)** model how interactions (incidents, events) unfold
• **Domain Model** used simplified **Class Diagram**
• **SSDs** use simplified **Sequence Diagrams**
System Sequence Diagrams

Elements of a System Sequence Diagram

- External actor to system
- A UML loop interaction frame, with a boolean guard expression
- Return value(s) associated with the previous message
- An abstraction that ignores presentation and medium
- The return line is optional if nothing is returned

System as black box

The name could be "NextGenPOS" but "System" keeps it simple

The ".:." and underline imply an instance, and are explained in a later chapter on sequence diagram notation in the UML

A message with parameters

It is an abstraction representing the system event of entering the payment data by some mechanism
Simple cash-only Process Sale scenario:
1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
...
**System Sequence Diagrams**

**Naming:** wording should be consistent with the domain!

- better name
- worse name

Also, being specific (exact?) is better

```
better name
```

: Cashier

```
enterItem(itemID, quantity)
```

: System

```
scan(itemID, quantity)
```

**enterItem** is what is done; **scan** serves the purpose of entering an item, and is system related (bad)
More on the purpose of SSDs

- SSDs are part of the **Use Case analysis**

- Draw an SSD for a main success scenario of each use case, and frequent or complex alternative scenarios

- SSDs allow for **black box** description of the behavior of the system

- SSDs can also be used to illustrate collaborations between **systems** *(other actors, no?)*
More on the purpose of SSDs

- SSDs are seldom used in the inception
  - If used, it is for early cost estimation
    - I mentioned that this is very hard and unreliable!

- SSDs are used in the elaboration for
  - Clarify major operations
    - Think about priority requirements, value
  - Write operation contracts
    - More on this later
    - Think about testing as well
  - Support ongoing estimation
    - Again, a hard task
Credits

Notes and figures adapted from