Object-Oriented Analysis and Design

Isara Anantavrasilp

Lecture 9: Dynamic Model

Recap

- We have discussed about:
  - System analysis and design
  - Abstraction and model
  - Requirement elicitation
    - Functional / Non-functional requirements
    - Scenario-based design
    - User stories
    - Work-Breakdown Structure
  - System structure design
    - Use case diagram
    - Package diagram
    - Class diagram

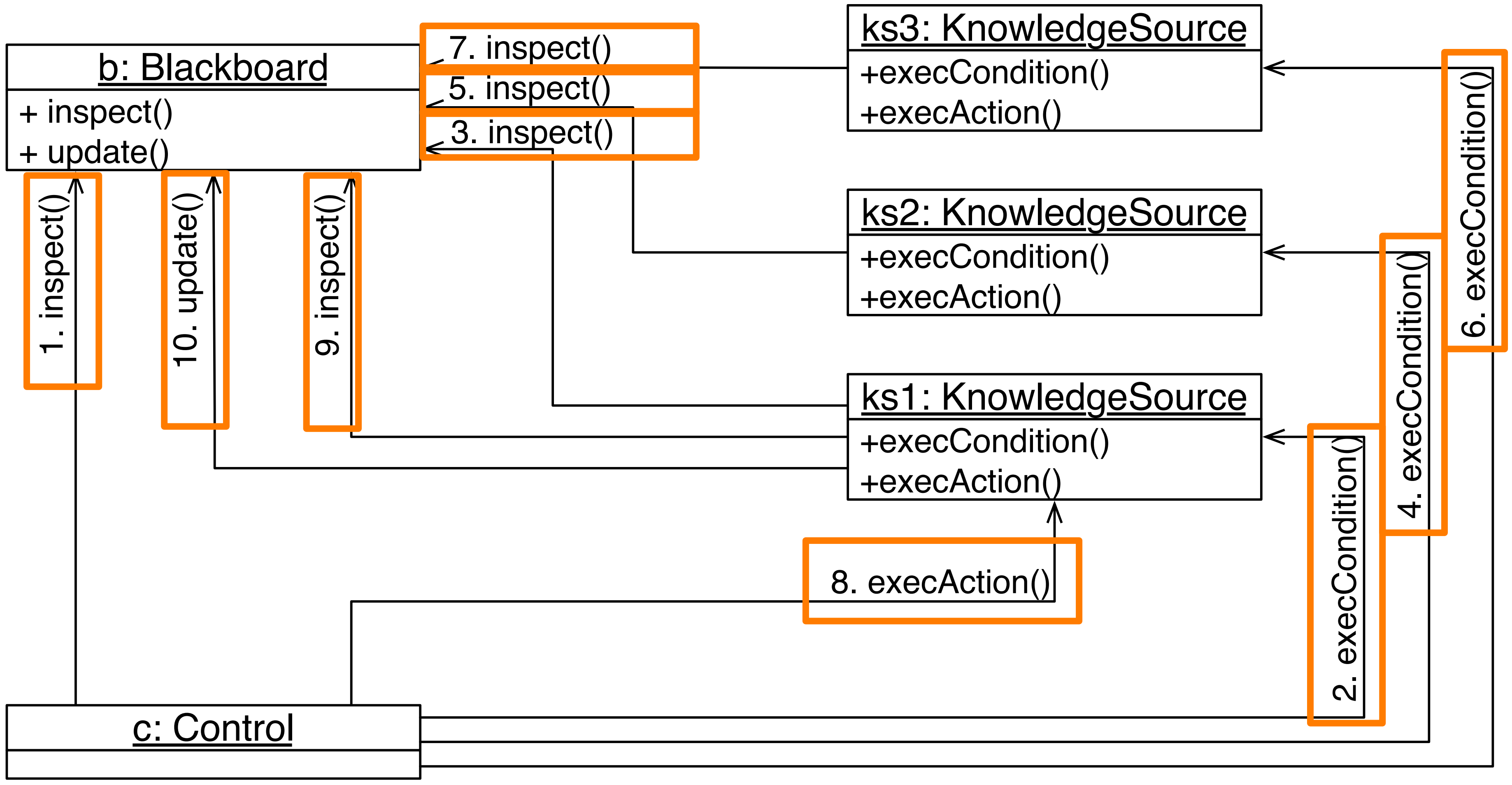
Static vs. Dynamic Model

- Package or class diagram describe the **structure** of a system
  - How the system is constructed
- They do **not** describe **behavior** of the system
  - How the system works
- Dynamic models describe system behavior
  - In UML, they are called **interaction diagrams**

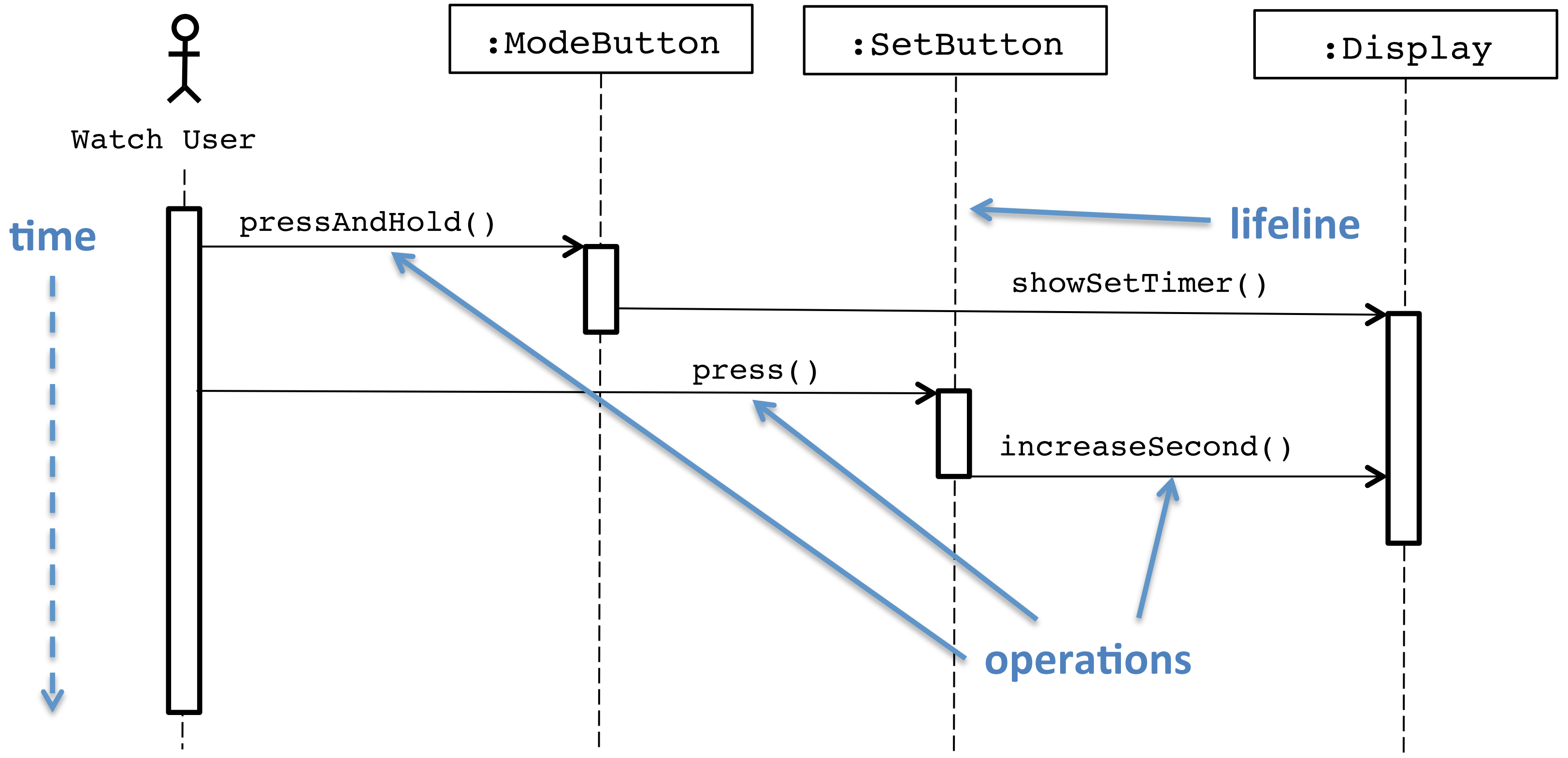
Interaction Diagrams

- Communication Diagram:
  - Describes interactions between objects in term of message exchanges
  - Messages are labeled numbers
- Sequence Diagram
  - Represents the flow of object interactions (exchanging messages)
  - The messages are shown in time sequence

Communication Diagram Example

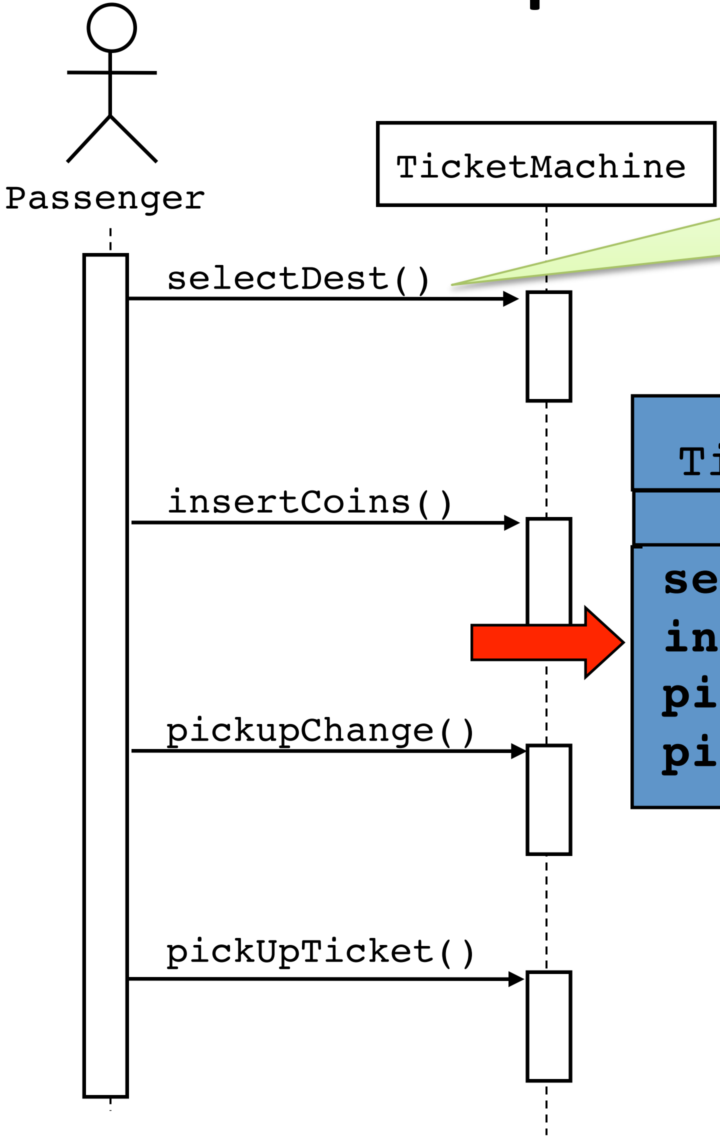


Sequence Diagram Example



© Isara Anantavrasilp

Sequence Diagrams



Focus on Controlflow

during analysis

refine use case descriptions

find additional objects ("participating objects")

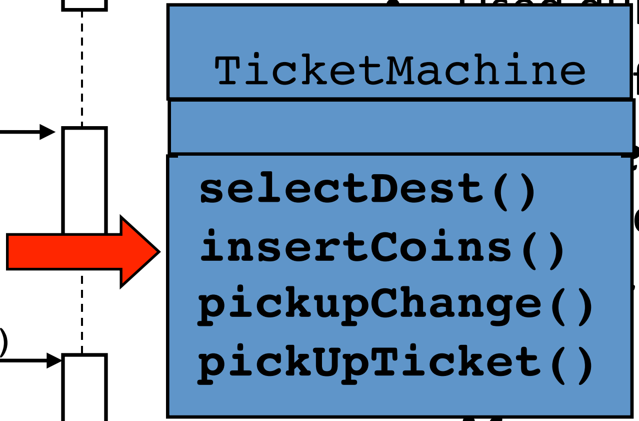
- ring system design

fine subsyst

s are repr

es. *Actors*

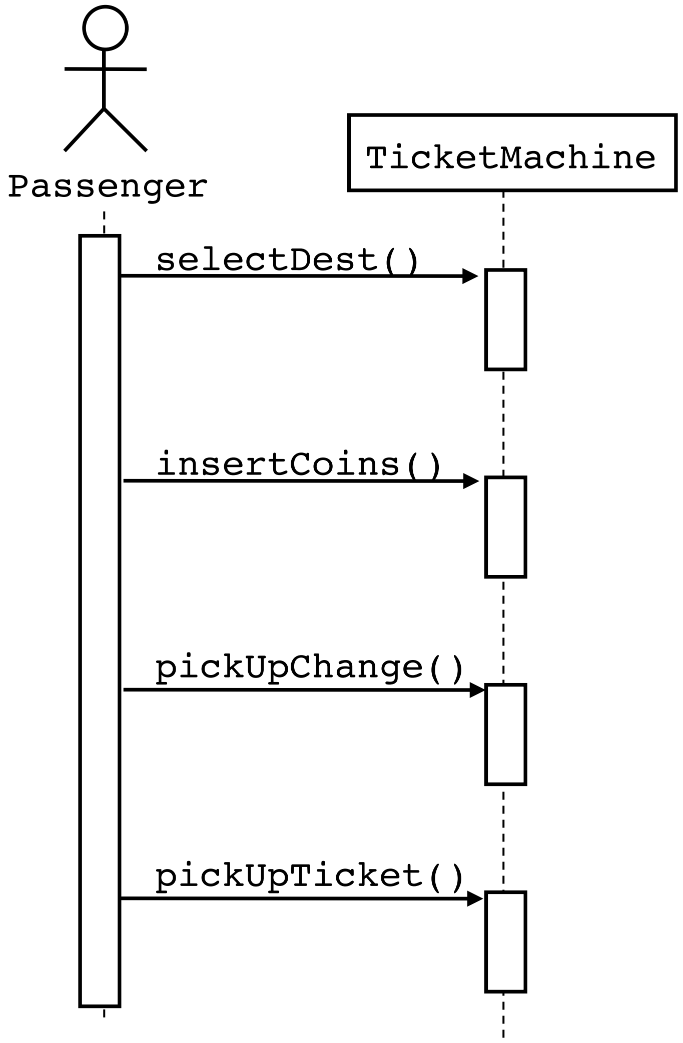
are repres



Messages -> Operations on participating Object

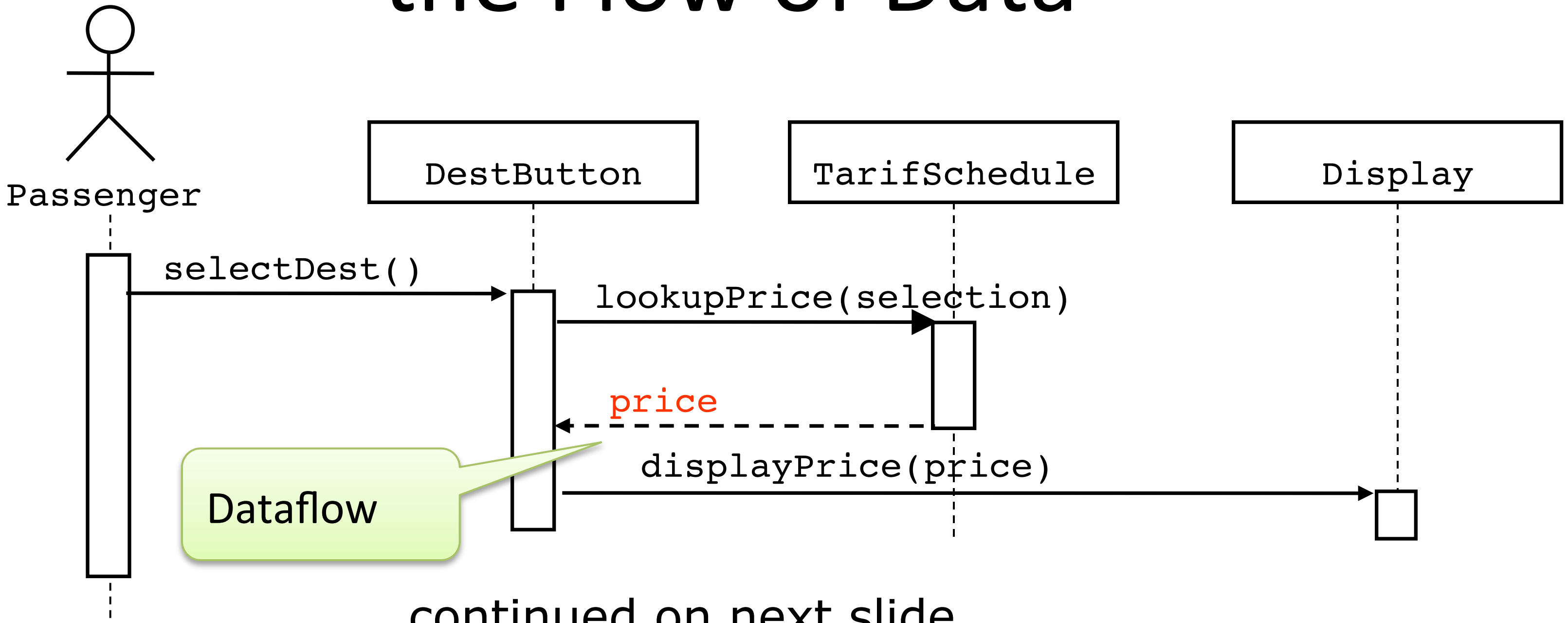
- *Messages* are represented by arrows
- *Activations* are represented by narrow rectangles.

System Sequence Diagram



- Sequence diagram can be used to describe communications between actor or external system with our system
- We call such diagram **System Sequence Diagram** (**SSD**)
- SSD has only two objects, the actor and the system

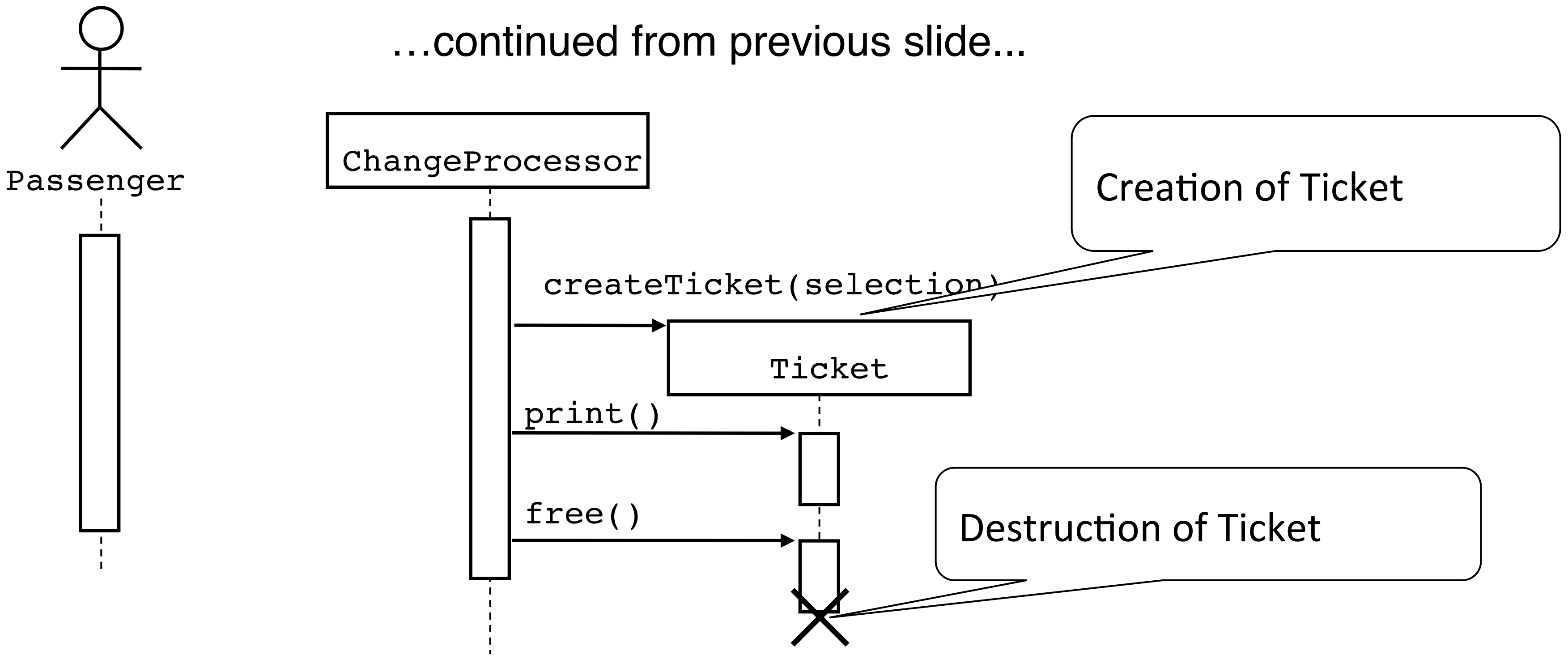
Sequence Diagrams can also model the Flow of Data



…continued on next slide…

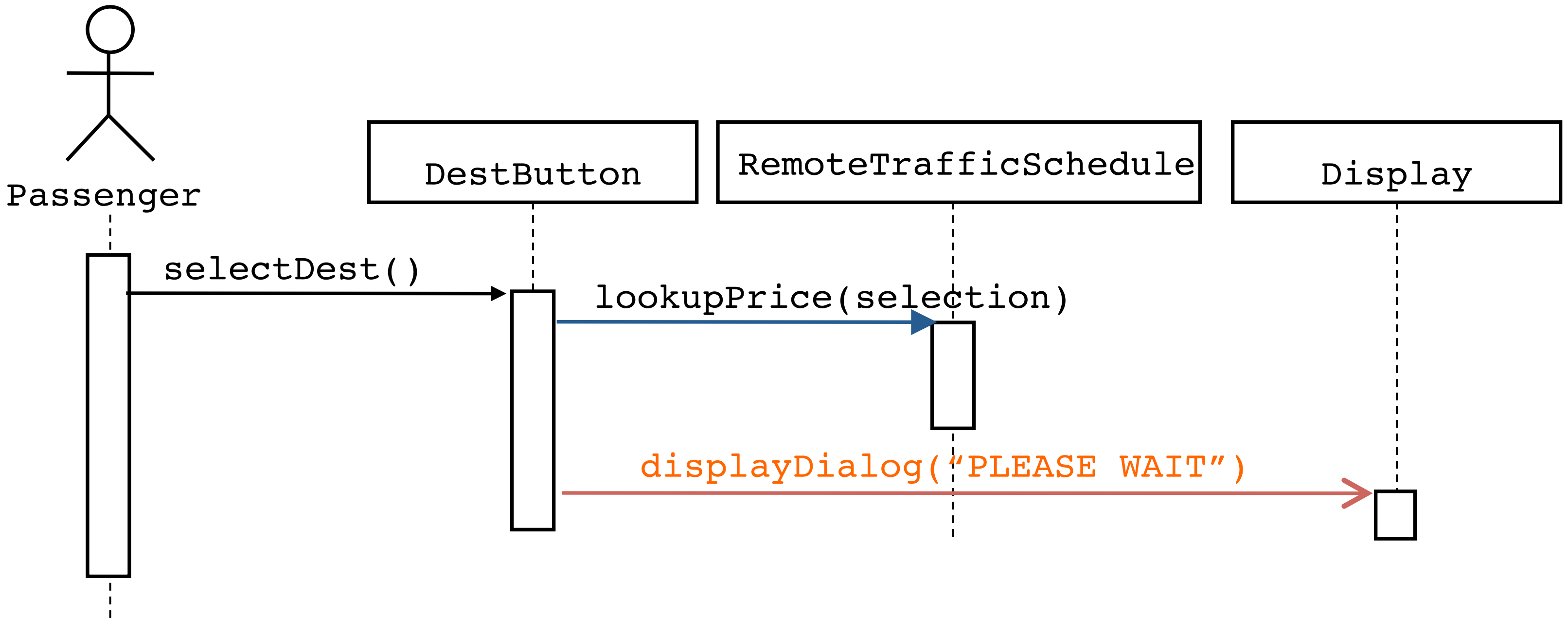
- The source of an arrow indicates the activation which sent the message
- Horizontal dashed arrows indicate data flow, for example return results from a message

Creation and Destruction



- Creation is denoted by a message arrow pointing to the object
- Destruction is denoted by an X mark at the end of the destruction activation
  - In garbage collection environments, destruction can be used to denote the end of the useful life of an object.

Asynchronous Message

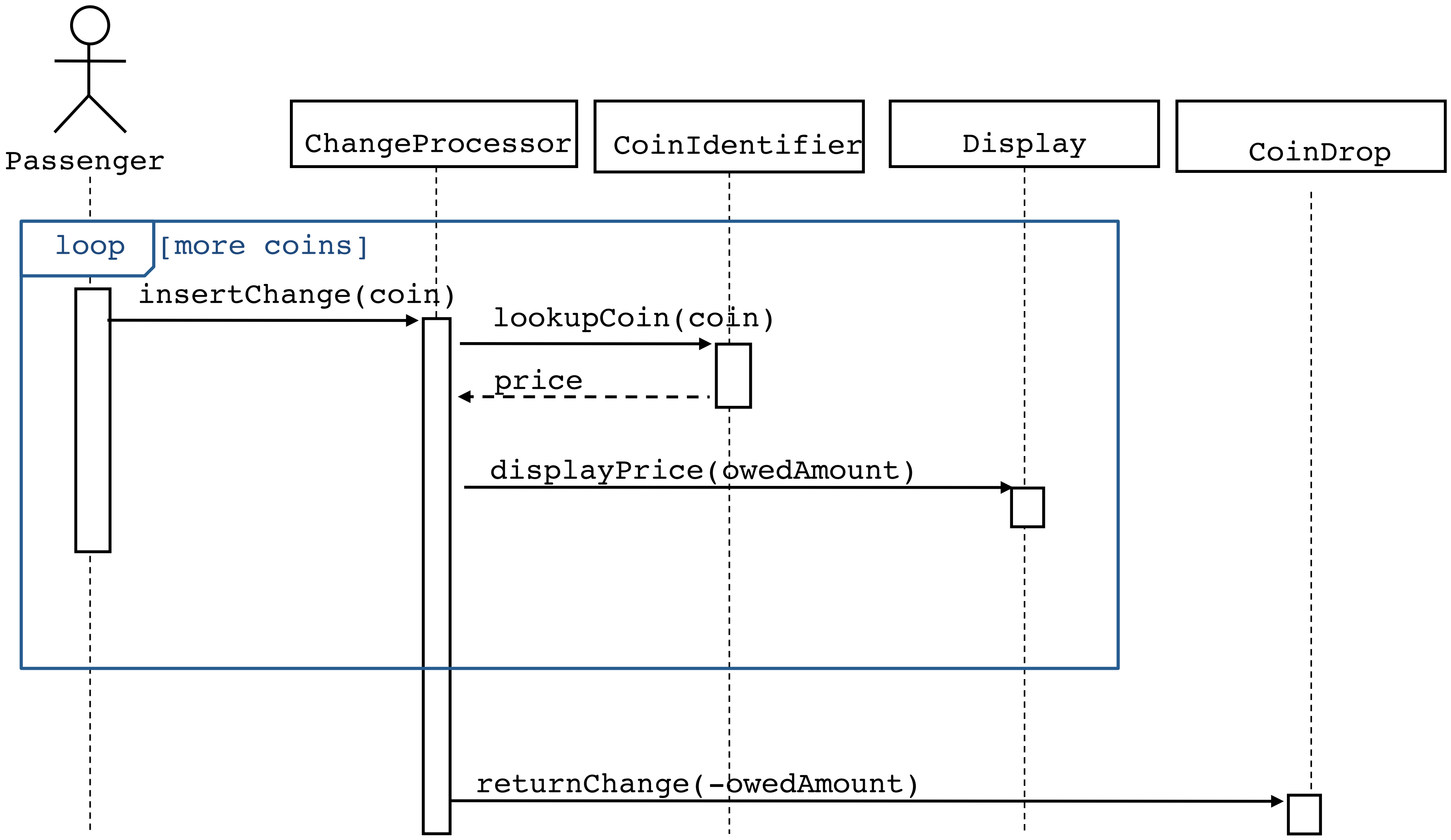


- Messages are normally sent in sequence. We have to wait until the operation of called class is done and return a response before moving on to the next step.
- However, sometimes **asynchronous** process is preferred.

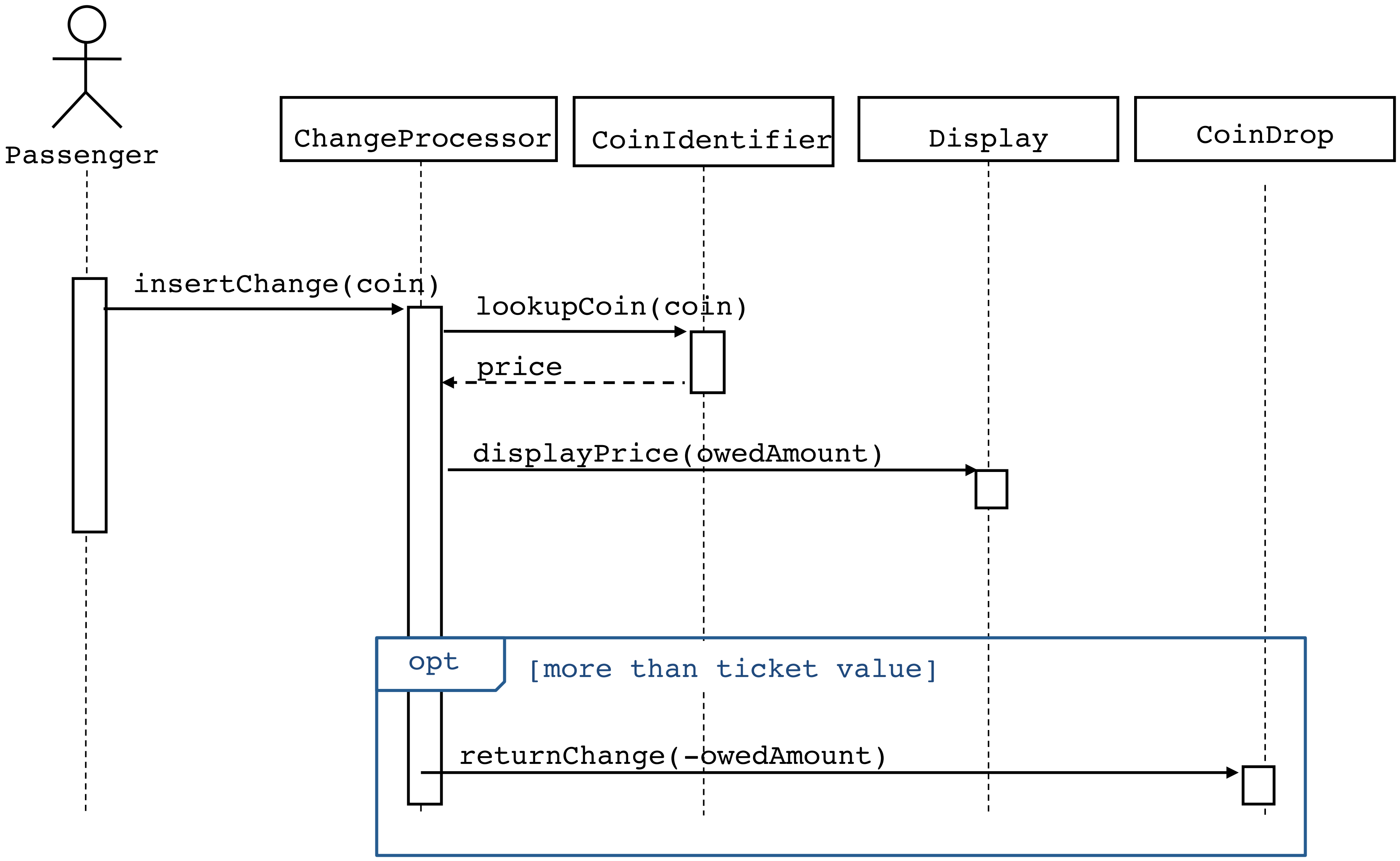
Program Logic with Sequence Diagram

- Sequence diagram also supports common program logics
  - Iteration or loop (for / while)
  - Condition (if-else): Option and Alternative
- Both uses a box with guard condition
- You can also refer to another diagram

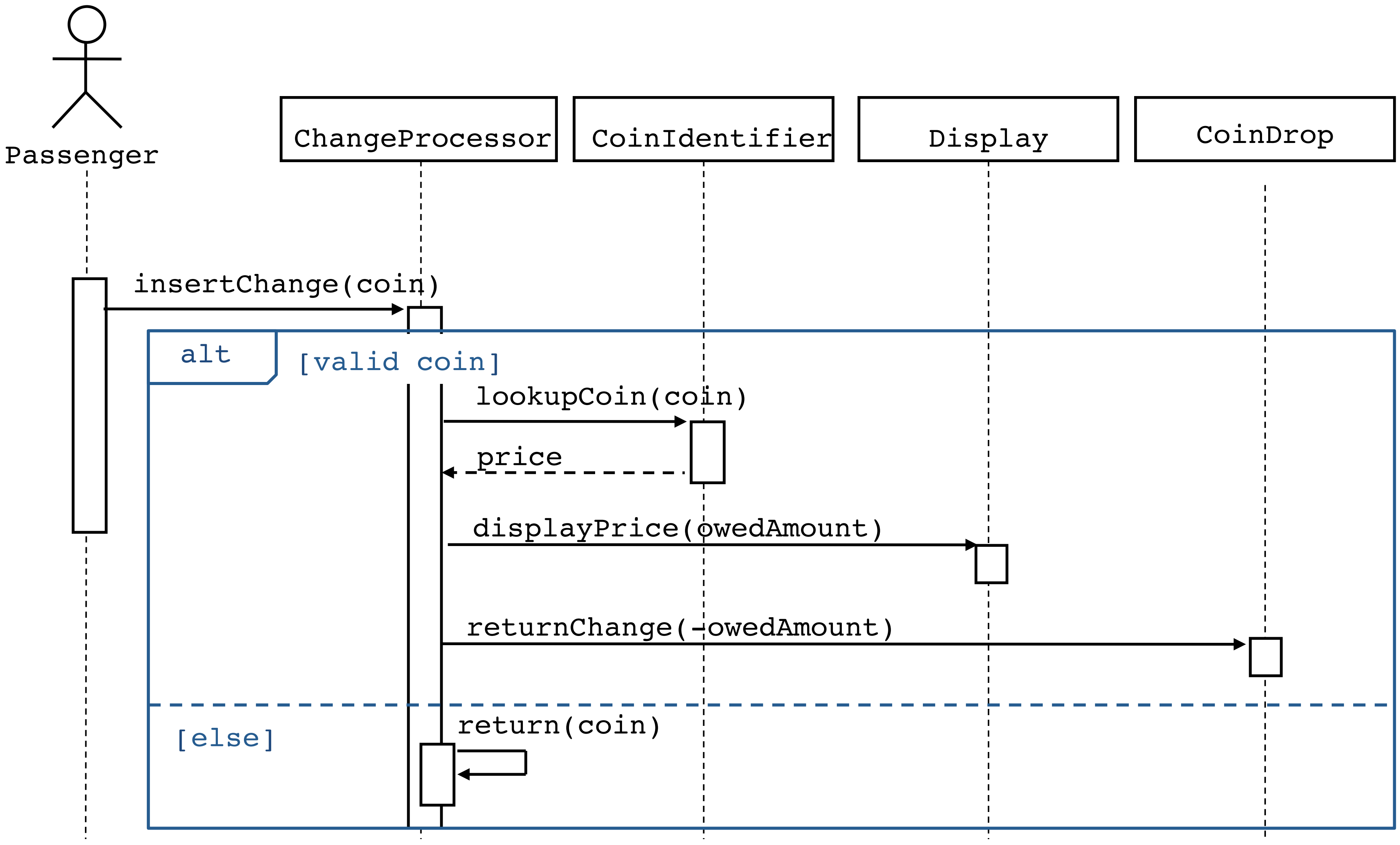
Sequence Diagrams: Iteration



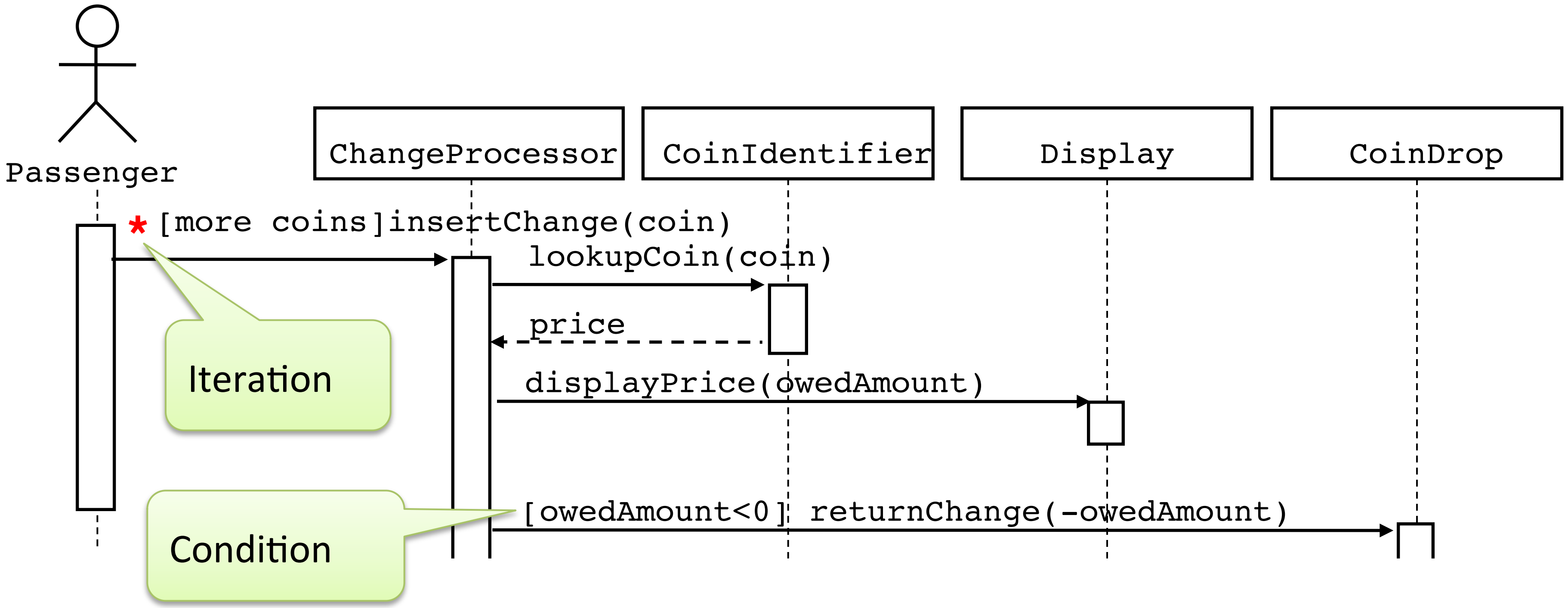
Sequence Diagrams: Option



Sequence Diagrams: Alternative



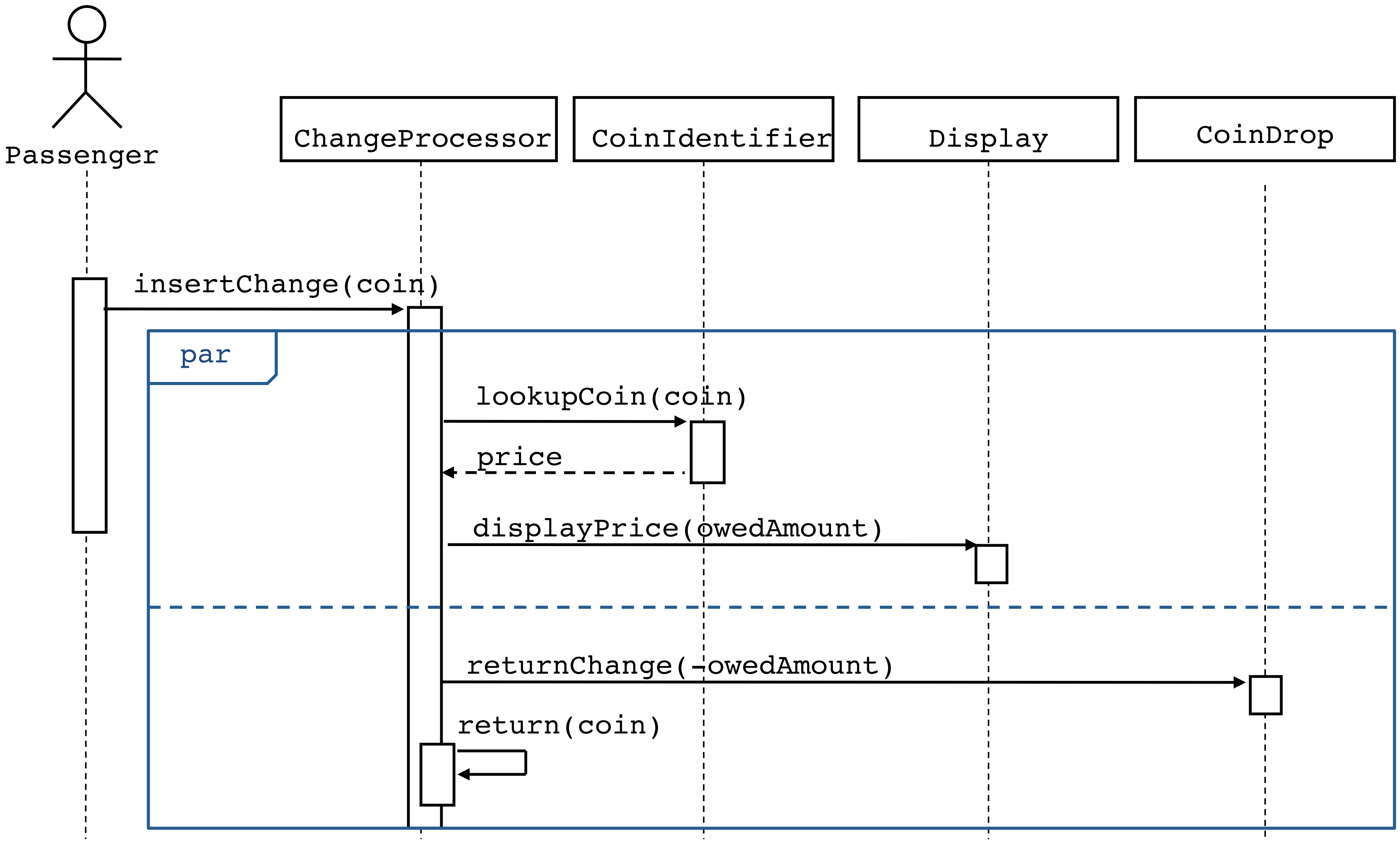
Iteration and Condition in Short Form



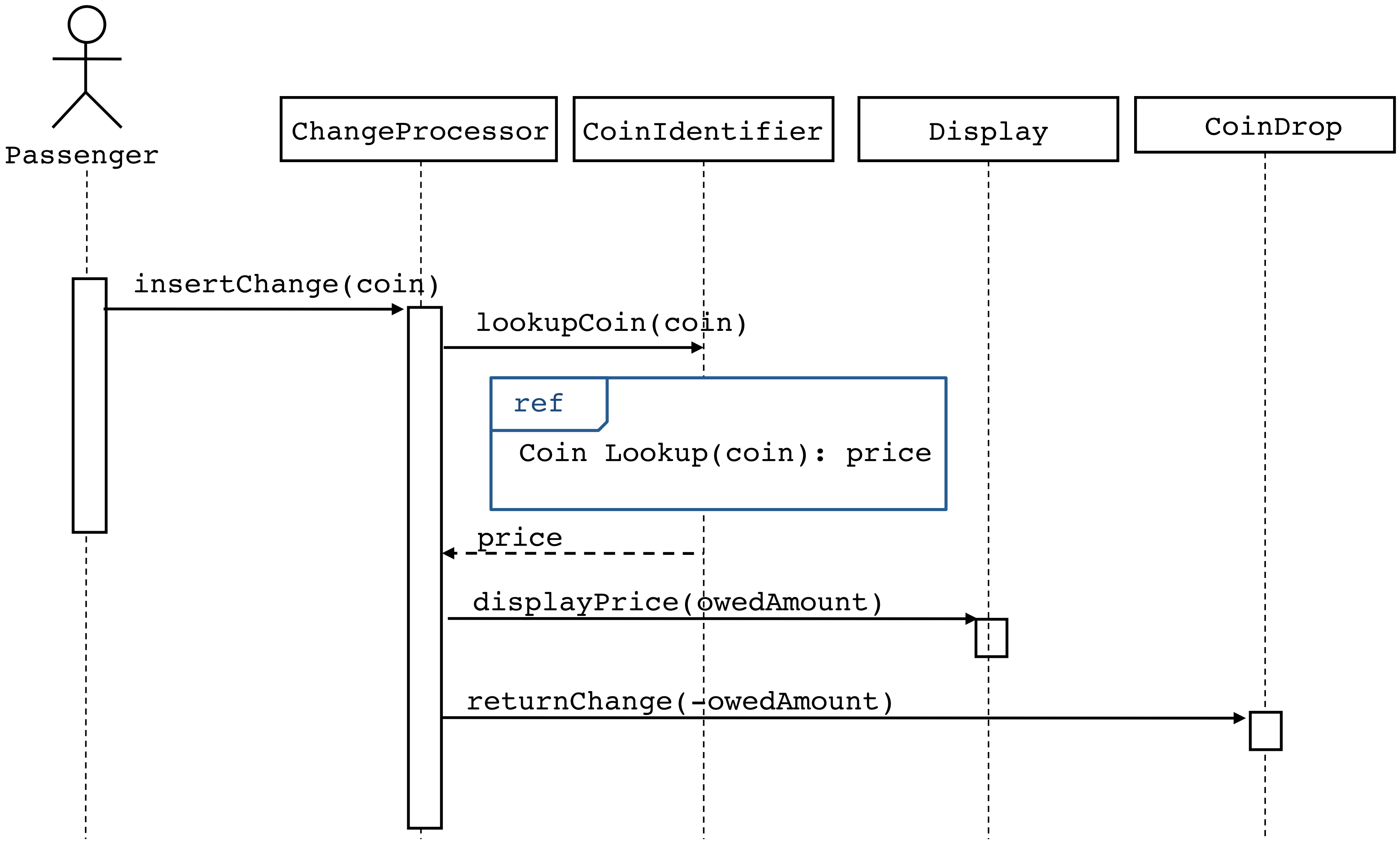
…continued on next slide…

- Iteration (loop) is denoted by a * preceding the message name with guard condition denoting when to stop
- Condition is denoted by Boolean expression in [ ] before the message name

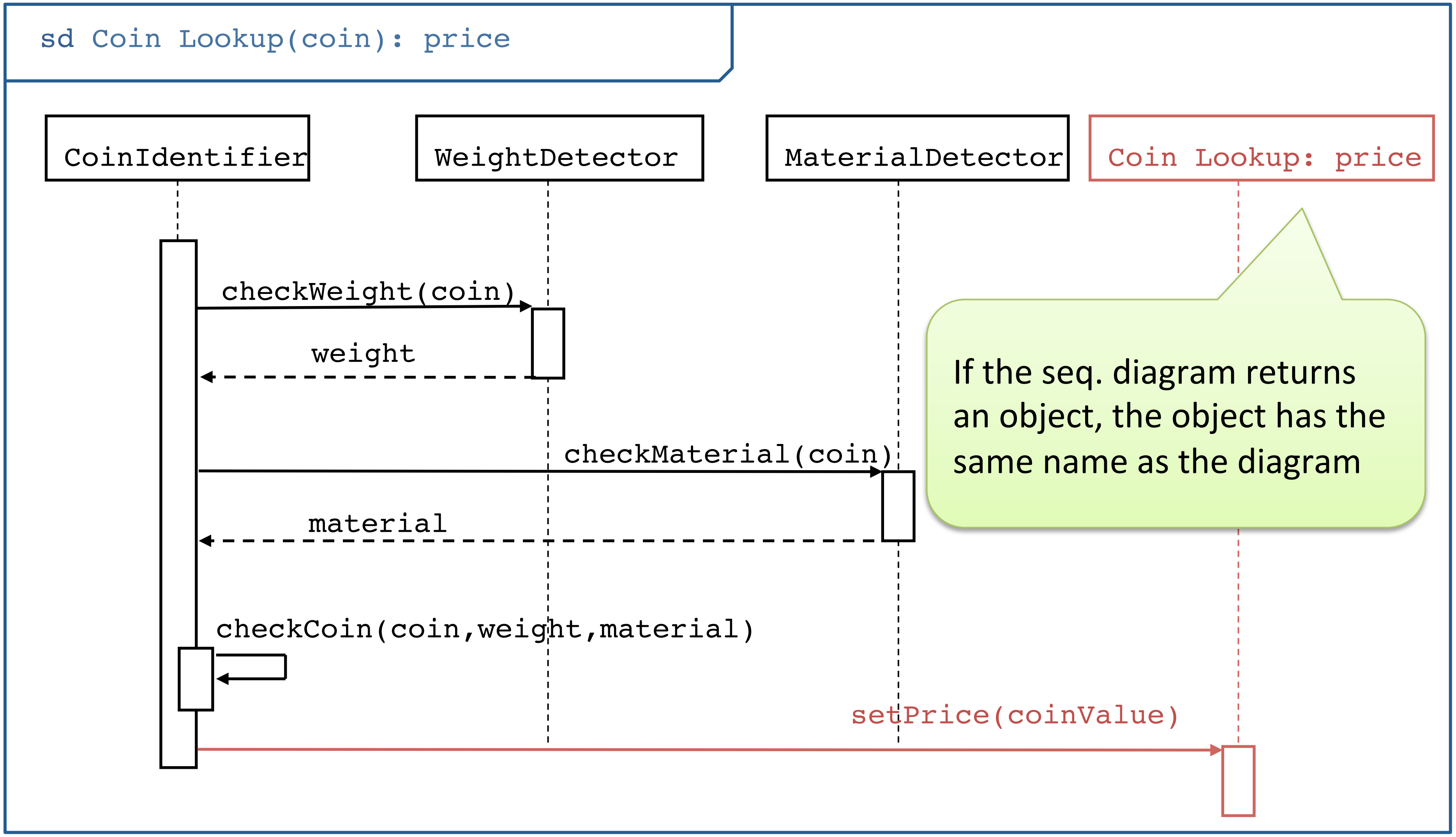
Sequence Diagrams: Parallel



Referring to Another Diagram



Referring to Another Diagram (2)



If the seq. diagram returns an object, the object has the same name as the diagram

Sequence Diagram Properties

- UML sequence diagram represent *behavior in terms of interactions*
- Useful to identify or find missing objects
- Time consuming to build, but *may* worth the investment
- Complement the class diagrams (which represent structure).