Use Case

Table 15.6 Use Case KBO_UC_01: Record Transaction
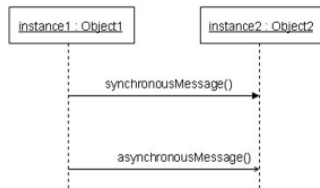
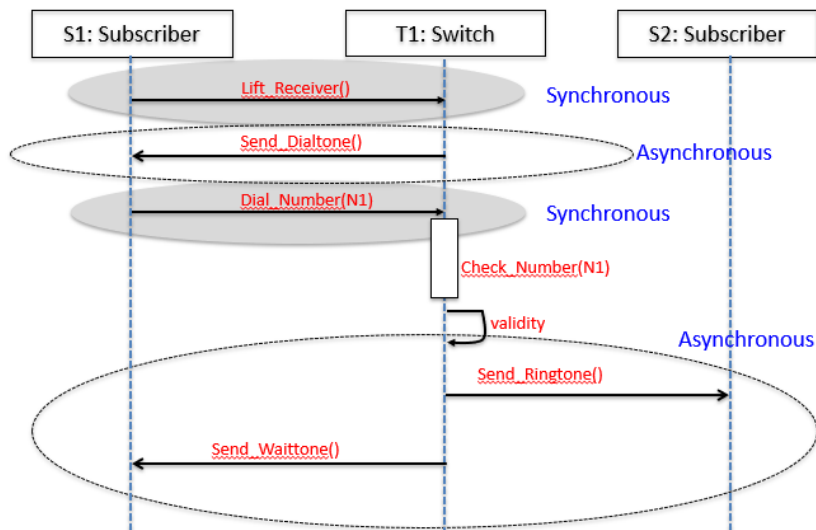| ID	KBO_UC_01
Name	Record Transaction
Objective	To record transaction information for a particular account in KBO.
Pre-conditions	Administrator must be logged into KBO and must be able to access transaction information from Master Ledger.
Post-conditions	• Success 1. Transaction information is successfully recorded in KBO. • Failure 1. Transaction has failed. 2. Transaction has succeeded but transaction information cannot be recorded in KBO.
Actors	• Primary 1. Administrator • Secondary 1. Master Ledger 2. User Notifier
Trigger	Periodic reminder to Administrator
Normal flow	1. Administrator requests transaction information for a particular account for a particular period of time from Master Ledger via the Get Transaction Information use case. 2. Administrator accepts information supplied by Master Ledger. 3. Administrator checks status of each particular transaction, whether success or failure via the Check Transaction Status use case. Administrator sends failed transaction information to failed transactions queue. System processes failed transaction information via the Notify User use case. 4. Administrator records succeeded transactions in the system. System confirms successful recording of transaction.
Alternative flow	1. a Master Ledger is not able to supply transaction information; use case concludes with error notification. 3. a. No successful transactions to record; use case concludes with error notification. 4. a. Successful transactions can not be recorded in KBO; use case concludes with error notification.
Interacts with	Get Transaction Information, Check Transaction Status, Record Transaction, Login, Notify User use cases.
Open issues	1. How quickly must Master Ledger respond to request for transaction information? 2. How will the system identify Administrator from other users?

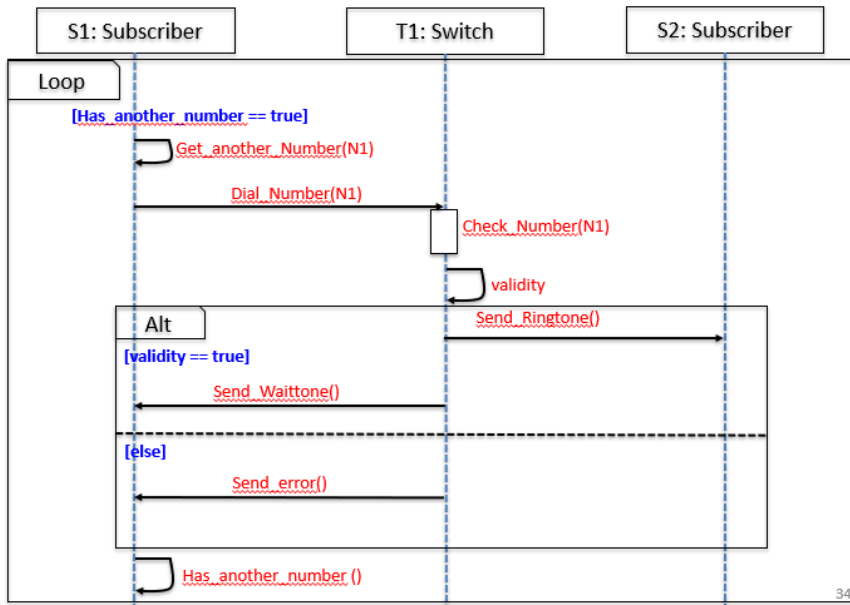# Misuse Case : Online Purchase
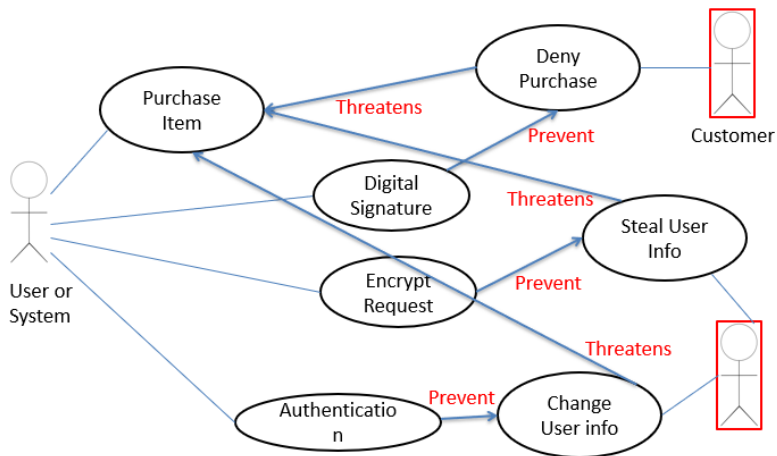
Arrows and Orders

- A solid arrowhead if a synchronous call operation
- A stick arrowhead if an asynchronous signal
- The order of the messages is defined two rules:
  - On the same lifeline, a higher message precedes a lower message
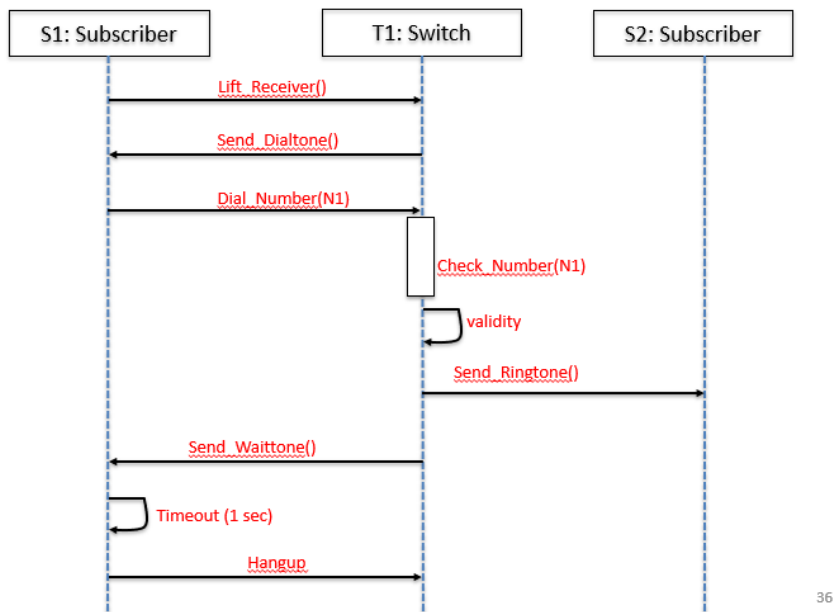 - Message sending precedes message receiving
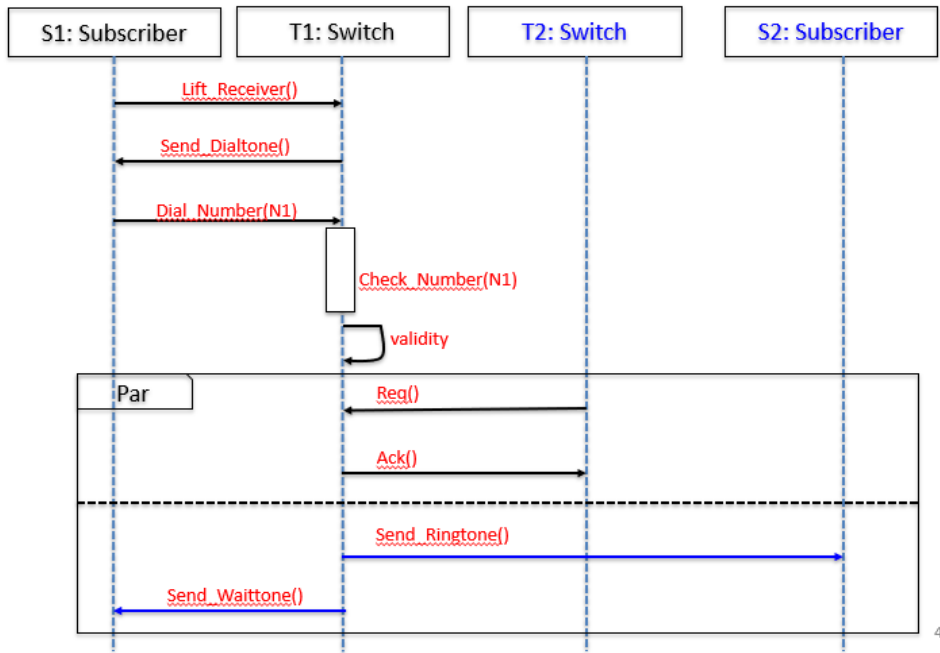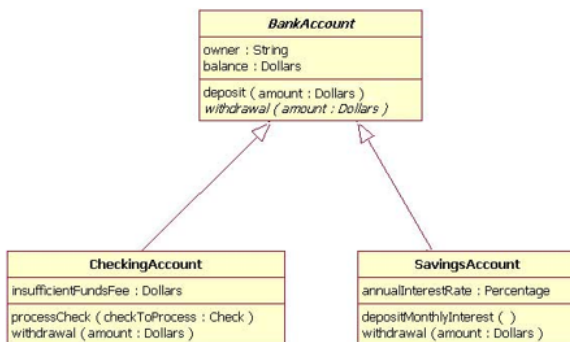

Synch and Asynch Messages

Loops

```
        S1: Subscriber          T1: Switch          S2: Subscriber

    ┌─ Loop ─────────────────────────────────────────────────────────┐
    │  [Has_another_number == true]                                   │
    │         ┌─┐  Get_another_Number(N1)                             │
    │         └─┘                                                     │
    │              Dial_Number(N1)                                    │
    │         ──────────────────────►┌─┐                             │
    │                                │ │ Check_Number(N1)            │
    │                                └─┘                             │
    │                                ┌─┐ validity                    │
    │                                └─┘                             │
    │   ┌─ Alt ──────────────────────── Send_Ringtone() ───────────►│
    │   │ [validity == true]                                         │
    │   │       Send_Waittone()                                      │
    │   │  ◄──────────────────────────                               │
    │   ├ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -│
    │   │ [else]                                                     │
    │   │       Send_error()                                         │
    │   │  ◄──────────────────────────                               │
    │   └────────────────────────────────────────────────────────────┘
    │         ┌─┐ Has_another_number ()                              │
    │         └─┘                                                     │
    └─────────────────────────────────────────────────────────────────┘
                                                                    34
```

Timeout

```
        S1: Subscriber          T1: Switch          S2: Subscriber

              Lift_Receiver()
         ──────────────────────►
              Send_Dialtone()
         ◄──────────────────────
              Dial_Number(N1)
         ──────────────────────►┌─┐
                                │ │
                                │ │ Check_Number(N1)
                                └─┘
                                ┌─┐ validity
                                └─┘
                                     Send_Ringtone()
                                ──────────────────────►
              Send_Waittone()
         ◄──────────────────────
         ┌─┐ Timeout (1 sec)
         └─┘
              Hangup
         ──────────────────────►
                                                                    36
```

Parallel (Combined)

```
S1: Subscriber          T1: Switch          T2: Switch          S2: Subscriber

        Lift_Receiver()
       ───────────────►
        Send_Dialtone()
       ◄───────────────
        Dial_Number(N1)
       ───────────────►
                         Check_Number(N1)
                         validity

┌─Par────────────────────────────────────────────────────────────────────┐
│                        Req()                                            │
│                       ◄───────────────                                  │
│                        Ack()                                            │
│                       ───────────────►                                 │
│─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ │
│                        Send_Ringtone()                                  │
│                       ──────────────────────────────────────►          │
│        Send_Waittone()                                                  │
│       ◄───────────────                                                  │
└────────────────────────────────────────────────────────────────────────┘
```

40

Inheritance

BankAccount

owner : String
balance : Dollars

deposit (amount : Dollars)
withdrawal (amount : Dollars)

CheckingAccount

insufficientFundsFee : Dollars

processCheck (checkToProcess : Check)
withdrawal (amount : Dollars)

SavingsAccount

annualInterestRate : Percentage

depositMonthlyInterest ( )
withdrawal (amount : Dollars)

Interfaces

Dotted line indicates it's NOT inheritance.
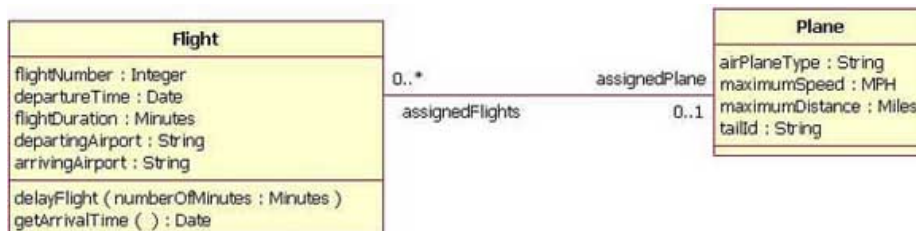
Multiplicity
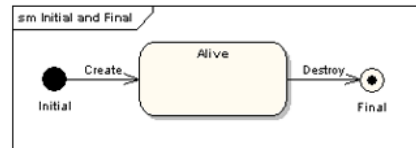
Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more
1..*	One or more
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen

State Machine Diagram

- States

sm State

Idle
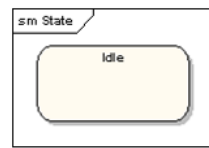
- Initial and final states

sm Initial and Final

Initial → Create → Alive → Destroy → Final

- Transitions

sm Transition

Source State → Trigger [Guard] /Effect → Target State

State Machine Example

Enrollment

Proposed → Open For Enrollment

Scheduled → Full → Closed to Enrollment

term started → Being Taught

student dropped [seminar size > 0]

classes end → Final Exams

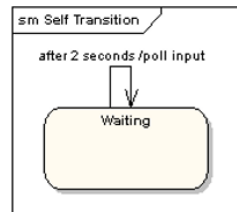student dropped [seminar size = 0]

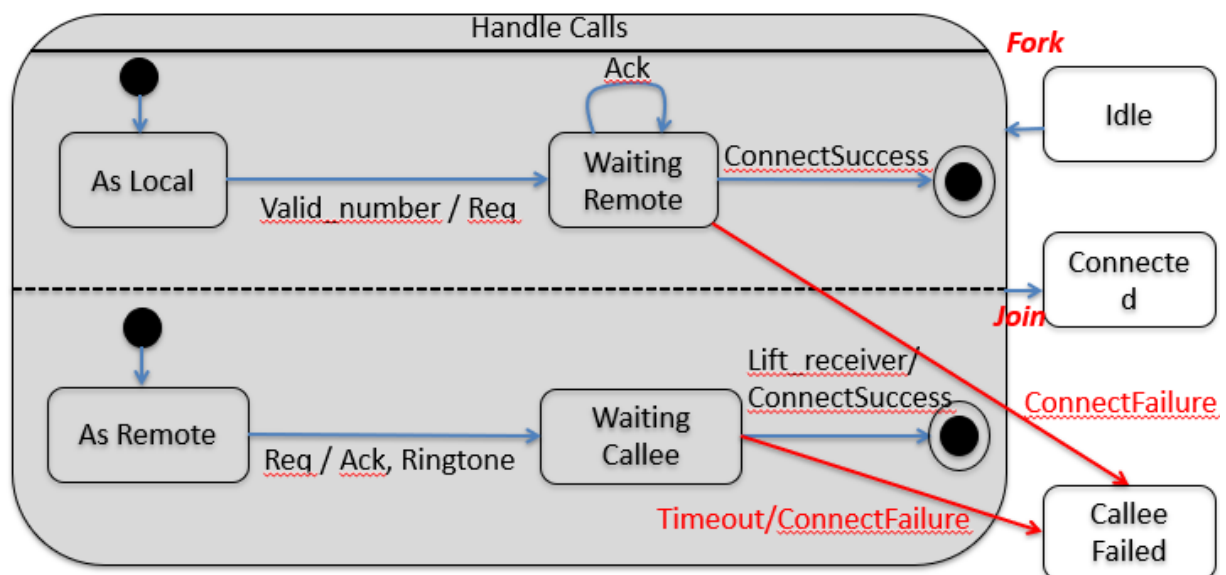cancelled

closed
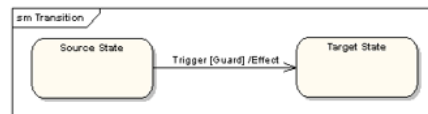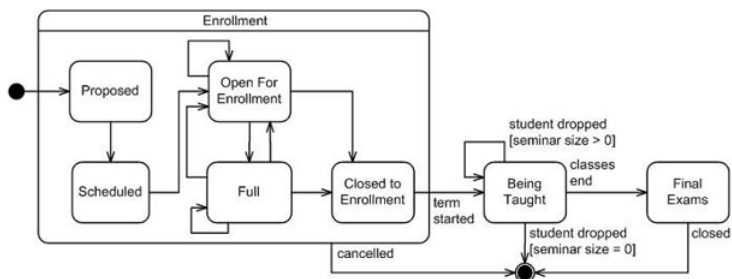
State Machine Diagrams: More

- State actions
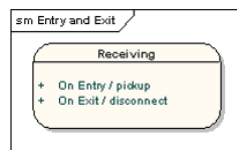
- Self-looping actions

Concurrent states

A switch may act as a local and remote switch at the same time

59