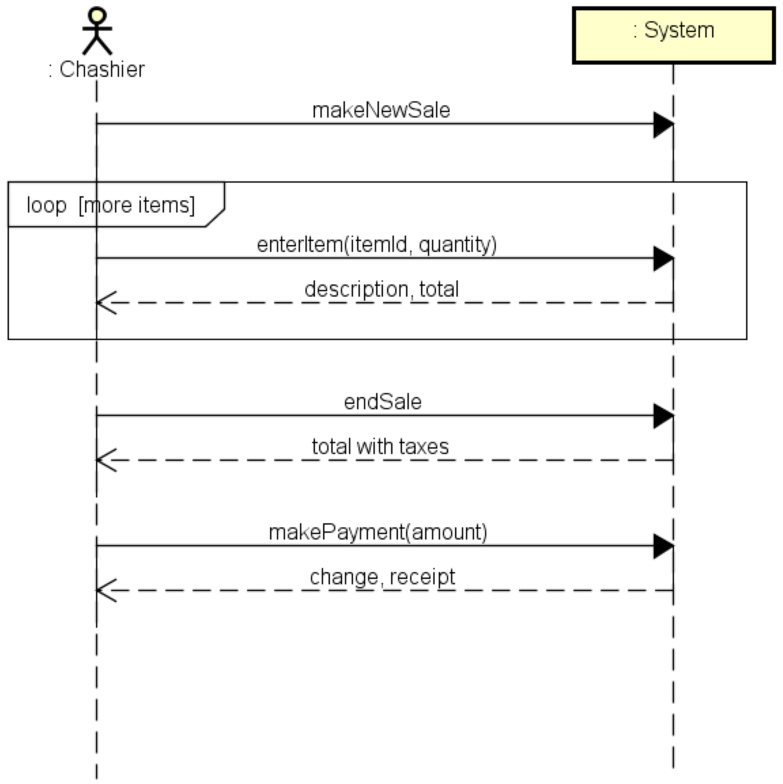
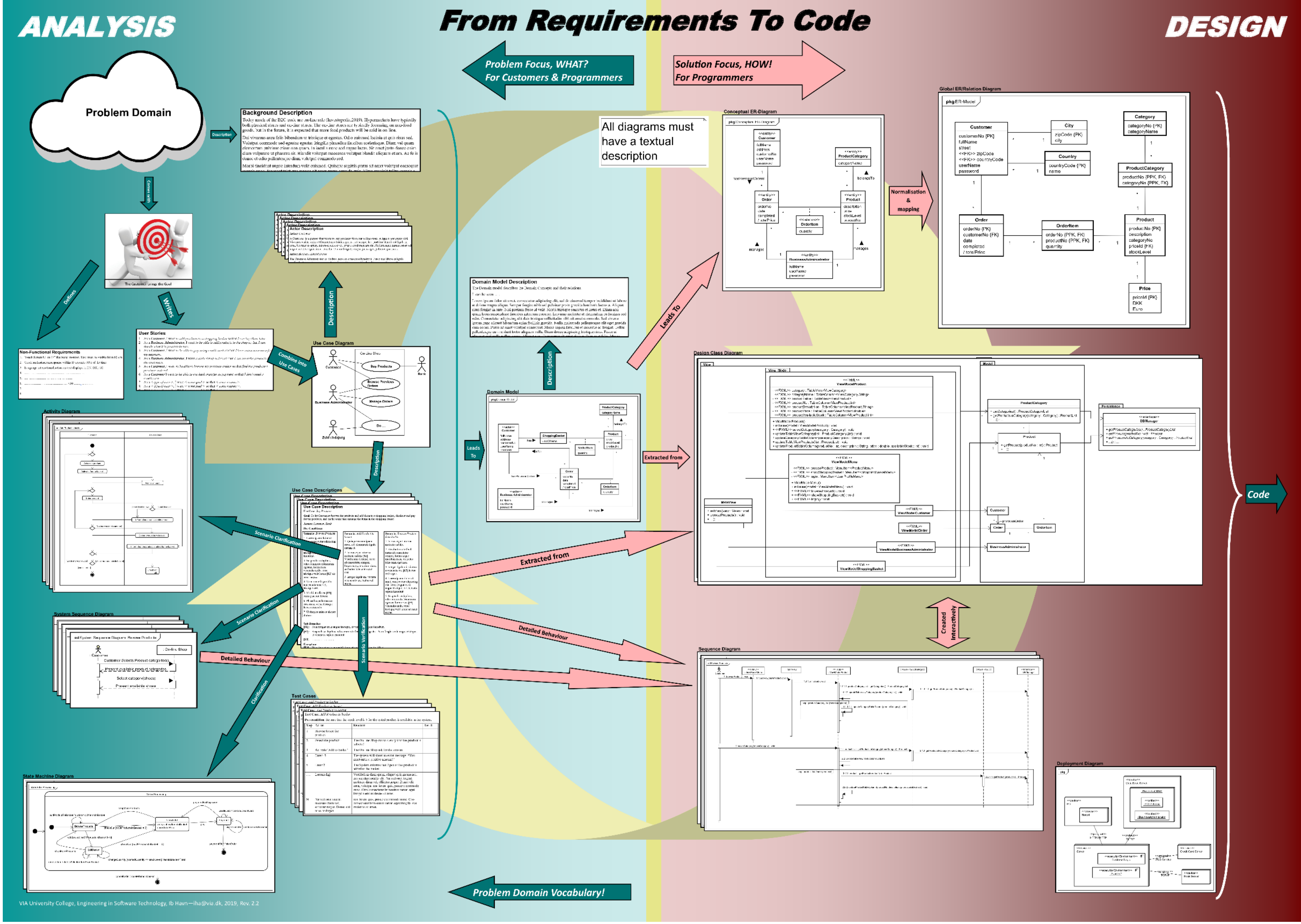
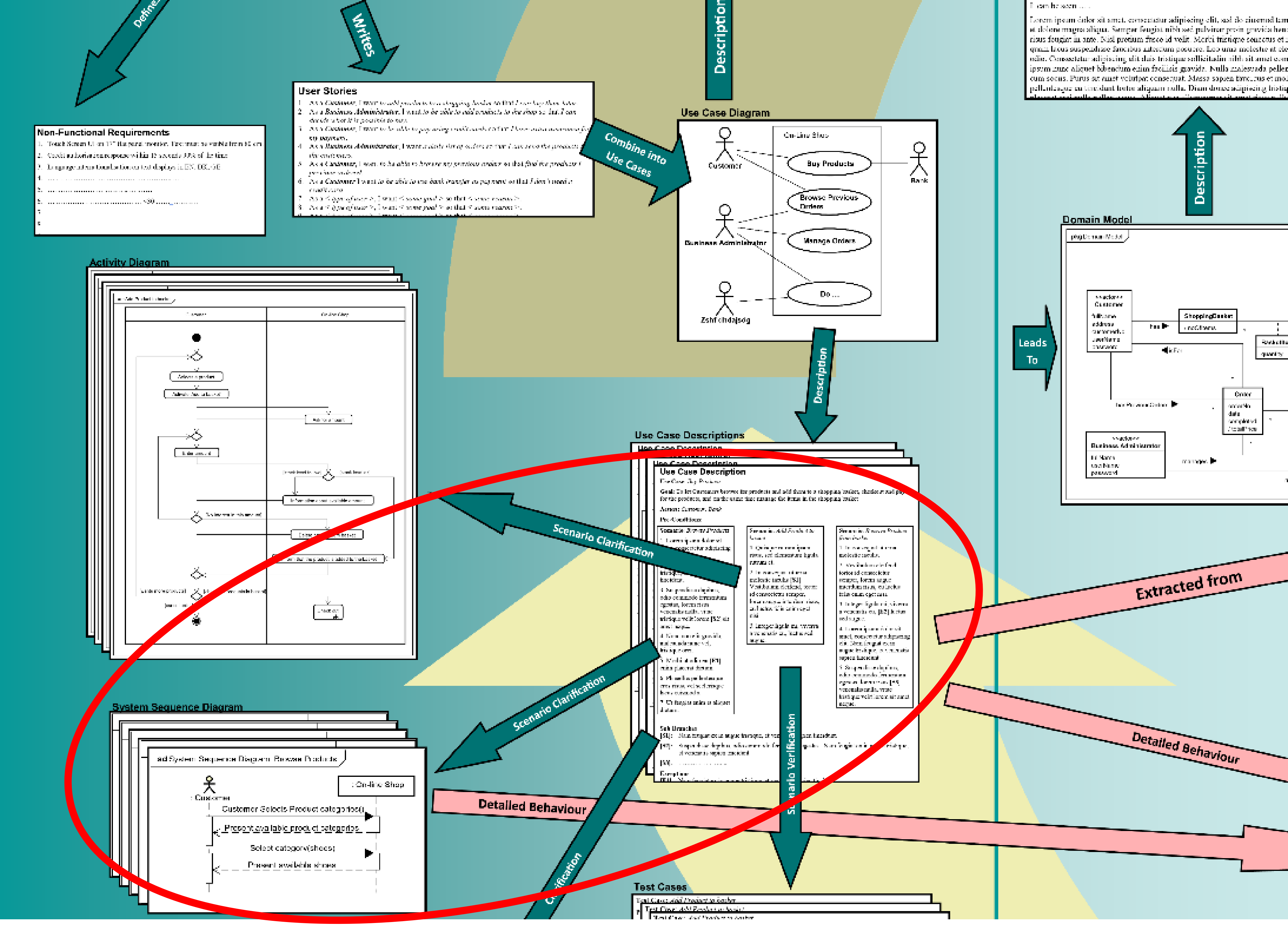
VIA University College
Software Engineering

System sequence diagram

[SWE1]

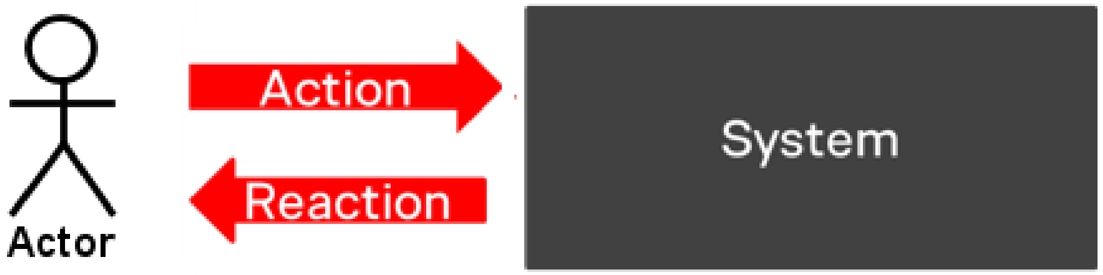



ANALYSIS
From Requirements To Code
DESIGN
Problem Focus, WHAT?
For Customers & Programmers
Solution Focus, HOW!
For Programmers
Problem Domain
Description
Background Description
Comes with
The Customer Brings the Goal
Defines
Writes
Non-Functional Requirements
User Stories
Actor Description
Description
Combine into Use Cases
Use Case Diagram
Customer
Buy Products
Browse Previous Orders
Business Administrator
Manage Orders
Bank
All diagrams must have a textual description
Domain Model Description
Description
Domain Model
Leads To
Extracted from
Conceptual ER-Diagram
Normalisation & mapping
Global ER/Relation Diagram
Customer
City
Country
Category
ProductCategory
Order
OrderItem
Product
Price
Leads To
Activity Diagram
Use Case Descriptions
Description
Scenario Clarification
System Sequence Diagram
Scenario Clarification
Detailed Behaviour
Scenario Verification
Clarification
Test Cases
State Machine Diagram
Extracted from
Design Class Diagram
Detailed Behaviour
Created Interactively
Sequence Diagram
Deployment Diagram
Code
Problem Domain Vocabulary!
VIA University College, Engineering in Software Technology, Ib Havn—iha@via.dk, 2019, Rev. 2.2


Defines
Writes
Description
User Stories
Non-Functional Requirements
Use Case Diagram
Combine into Use Cases
On-Line Shop
Customer
Buy Products
Bank
Browse Previous Orders
Business Administrator
Manage Orders
Do ...
Description
Domain Model
Leads To
Activity Diagram
Use Case Descriptions
Use Case Description
Scenario Clarification
Extracted from
Scenario Clarification
System Sequence Diagram
sd System Sequence Diagram Browse Products
:On-line Shop
:Customer
Customer Selects Product categories()
Present available product categories
Select categories(shoes)
Present available shoes
Detailed Behaviour
Detailed Behaviour
Scenario Verification
Clarification
Test Cases

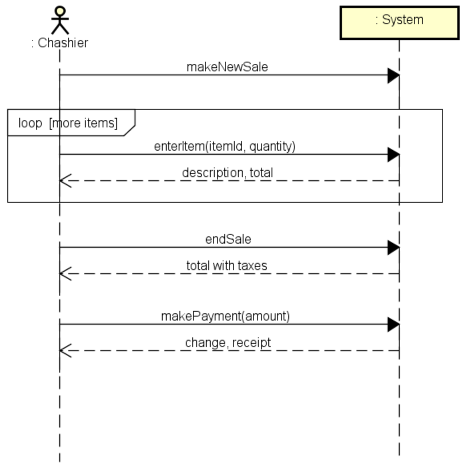
System sequence diagram (SSD)

SSD presents events in/out between actor and system
– The system is treated as a **black-box**



SSD show **one** scenario of a Use Case, typically

1. Main success scenario
2. Alternative/exception scenarios

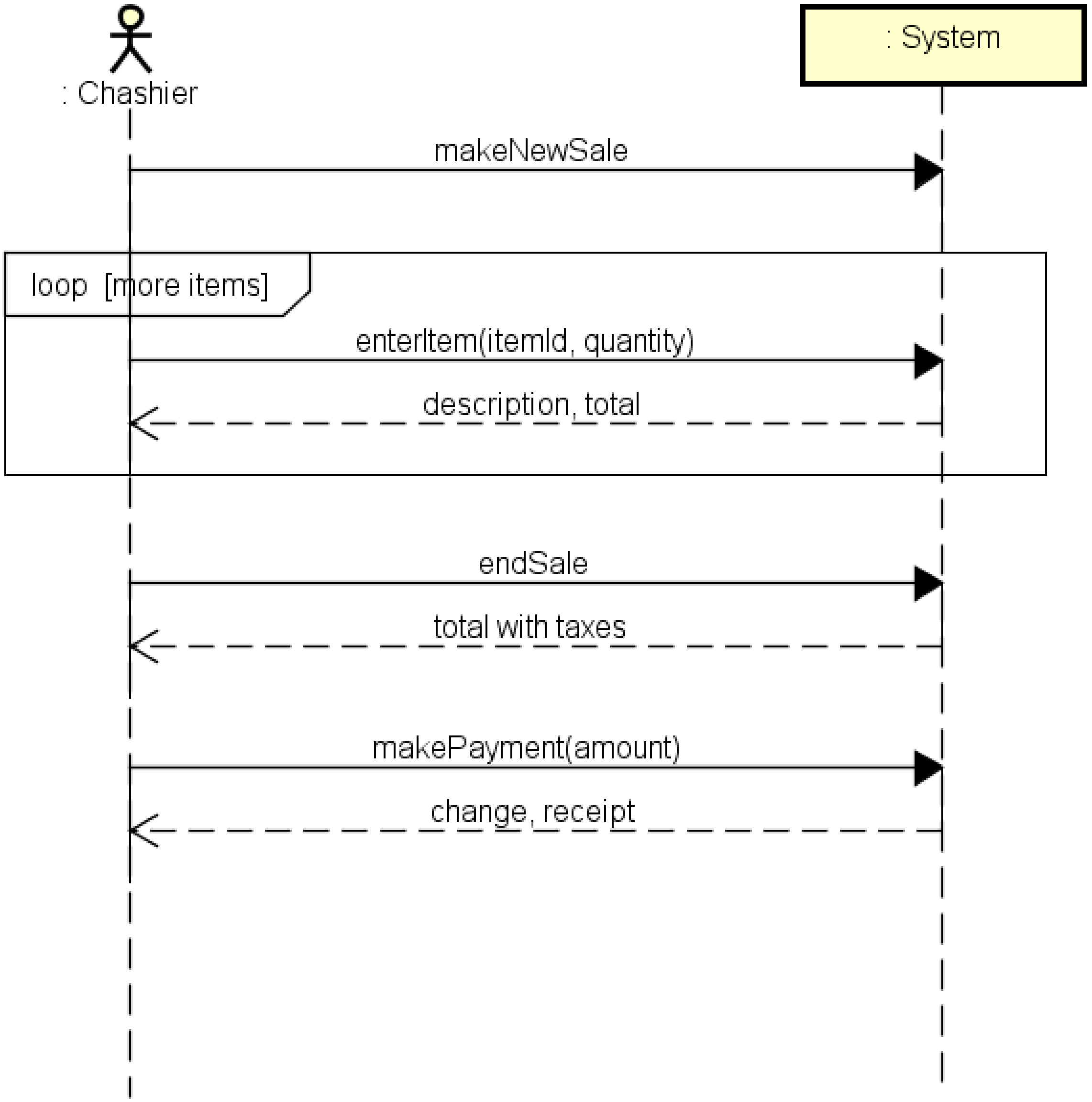


System sequence diagram (SSD)

An **Analysis** artefact

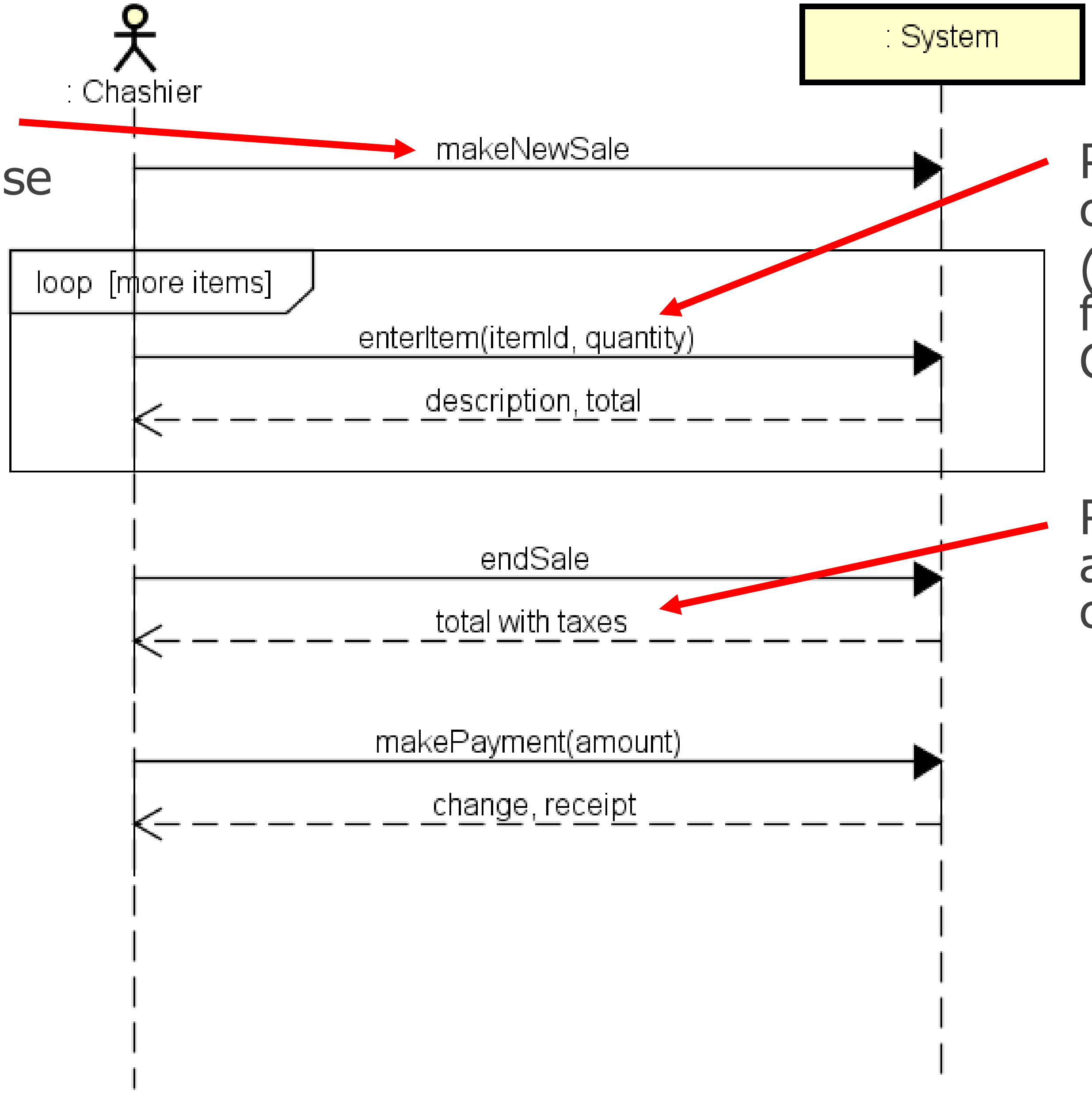
SSD meant for

1. Our costumer - to make sure we have understood the scenario correctly
2. Developers - as input for later design



SSD Example

Events comes from a Use Case scenario



Parameters are conceptual

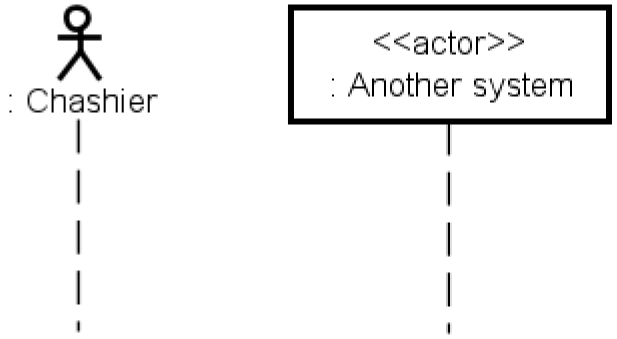
(and are coming from the Use Case scenarios)

Return values are also conceptual

Type of events

The System must be ready to react on three types of events

1. External events to/from Actors
   - Humans or other systems



2. Time events

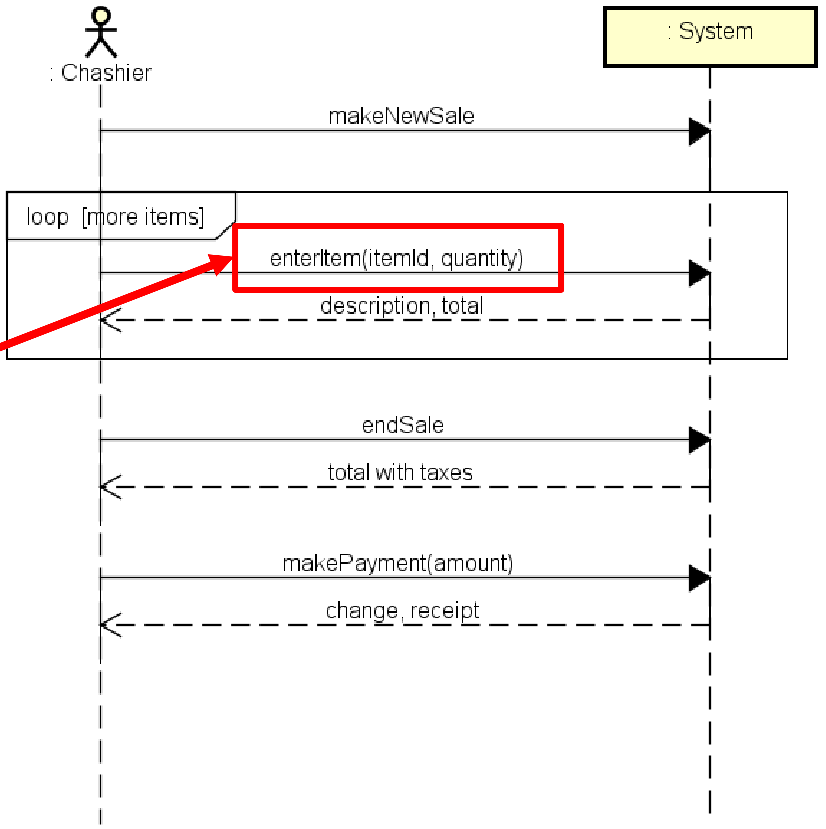
3. Faults or exceptions
   - E.g. from external systems

SSD Details

Naming Events/Operation

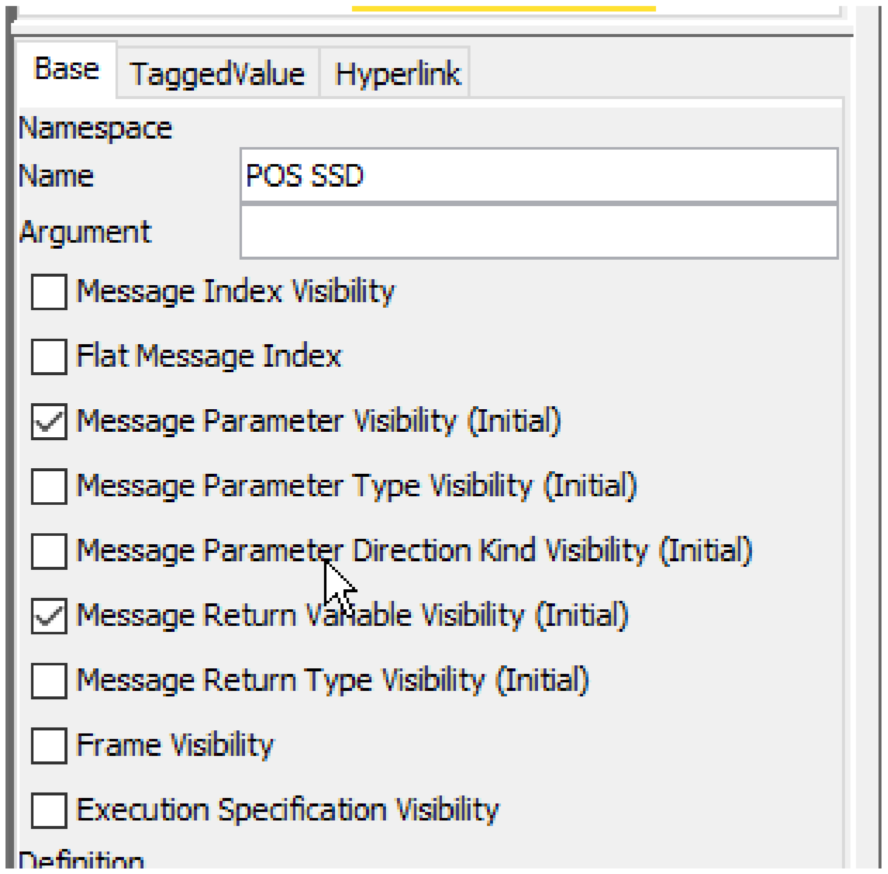
- System events should be on an abstract level of intentions – not how or from what they are generated

enterItem is better than *scanItem* (covers keyboard, scanner etc.)



SSD in Astah

- Diagram type: Sequence diagram
- Don't show indices, return types, etc

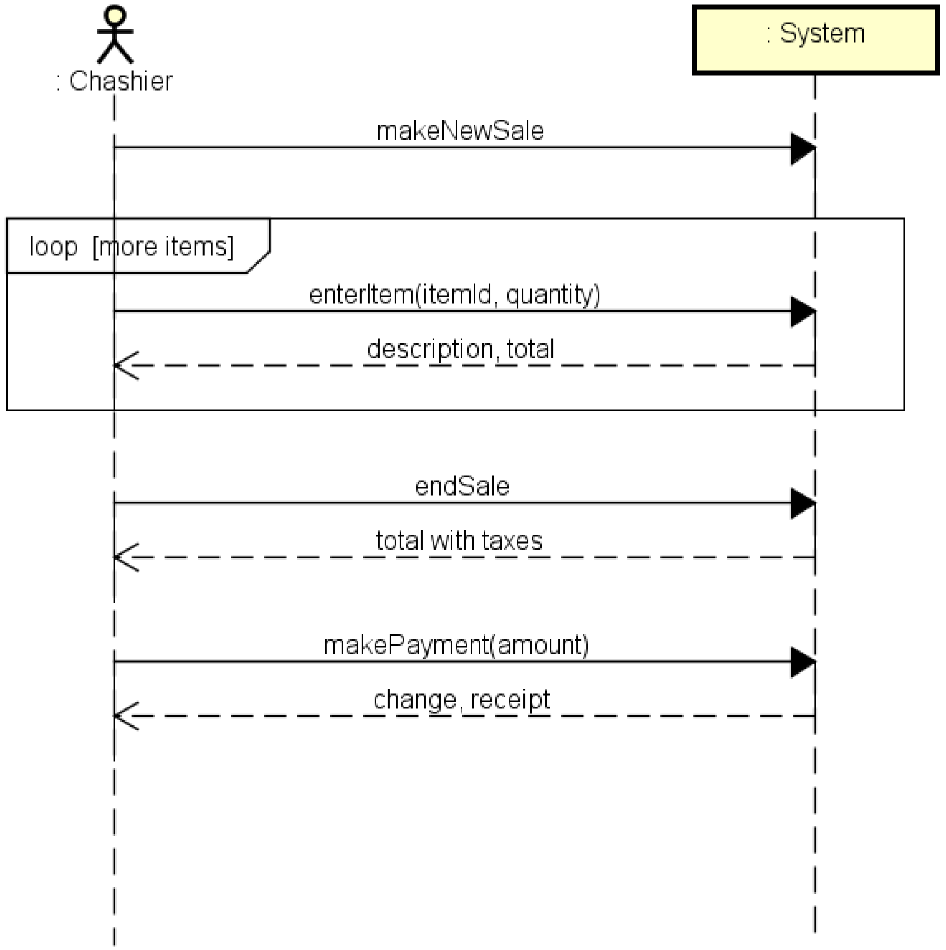


Base | TaggedValue | Hyperlink

Namespace	
Name	POS SSD
Argument	

- [ ] Message Index Visibility
- [ ] Flat Message Index
- [✓] Message Parameter Visibility (Initial)
- [ ] Message Parameter Type Visibility (Initial)
- [ ] Message Parameter Direction Kind Visibility (Initial)
- [✓] Message Return Variable Visibility (Initial)
- [ ] Message Return Type Visibility (Initial)
- [ ] Frame Visibility
- [ ] Execution Specification Visibility

Definition

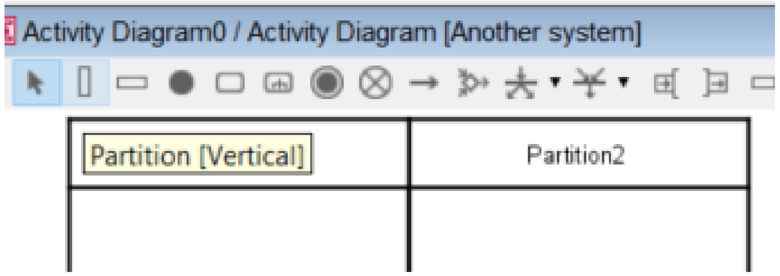


Activity Diagrams

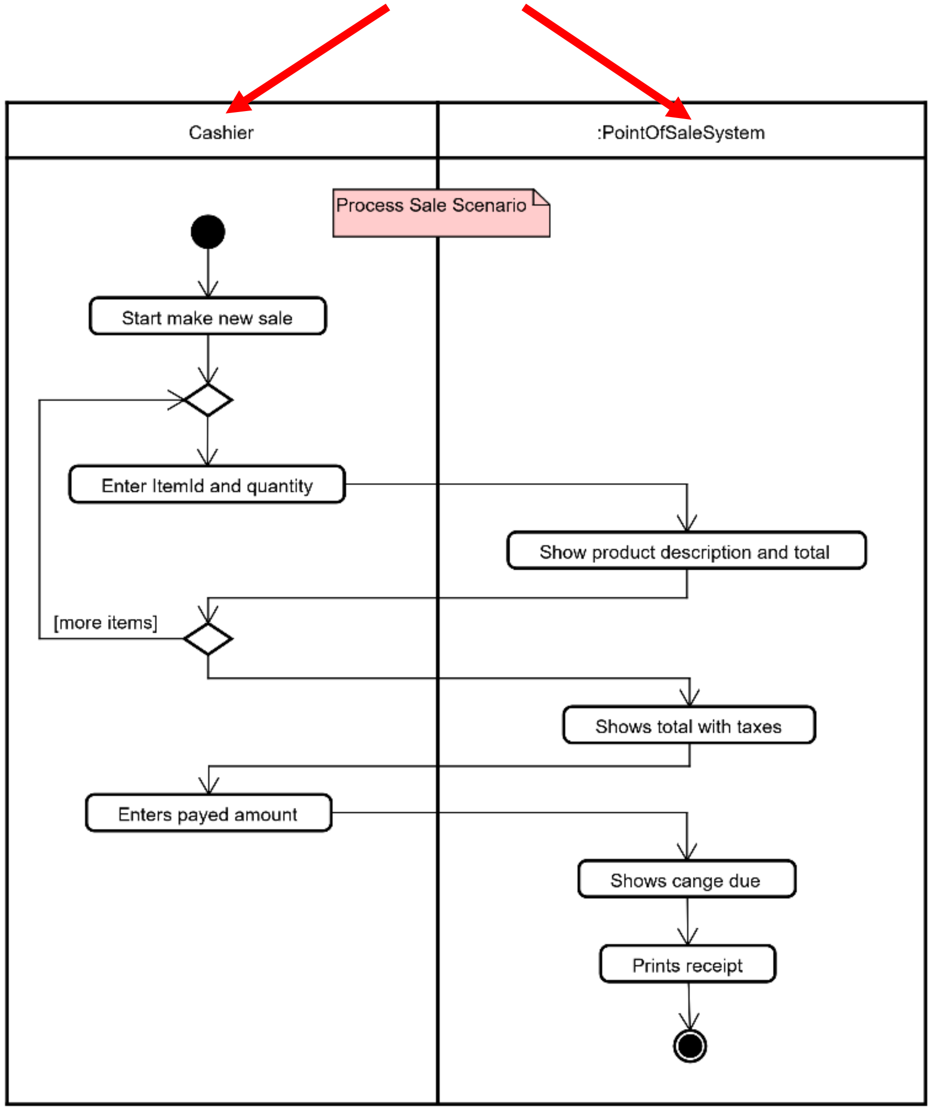
Activity Diagrams can also be used as supplement to Use Case Scenarios

- Only show System activities directly involved in the interaction with the Actor
- Do NOT show what is going on in the System

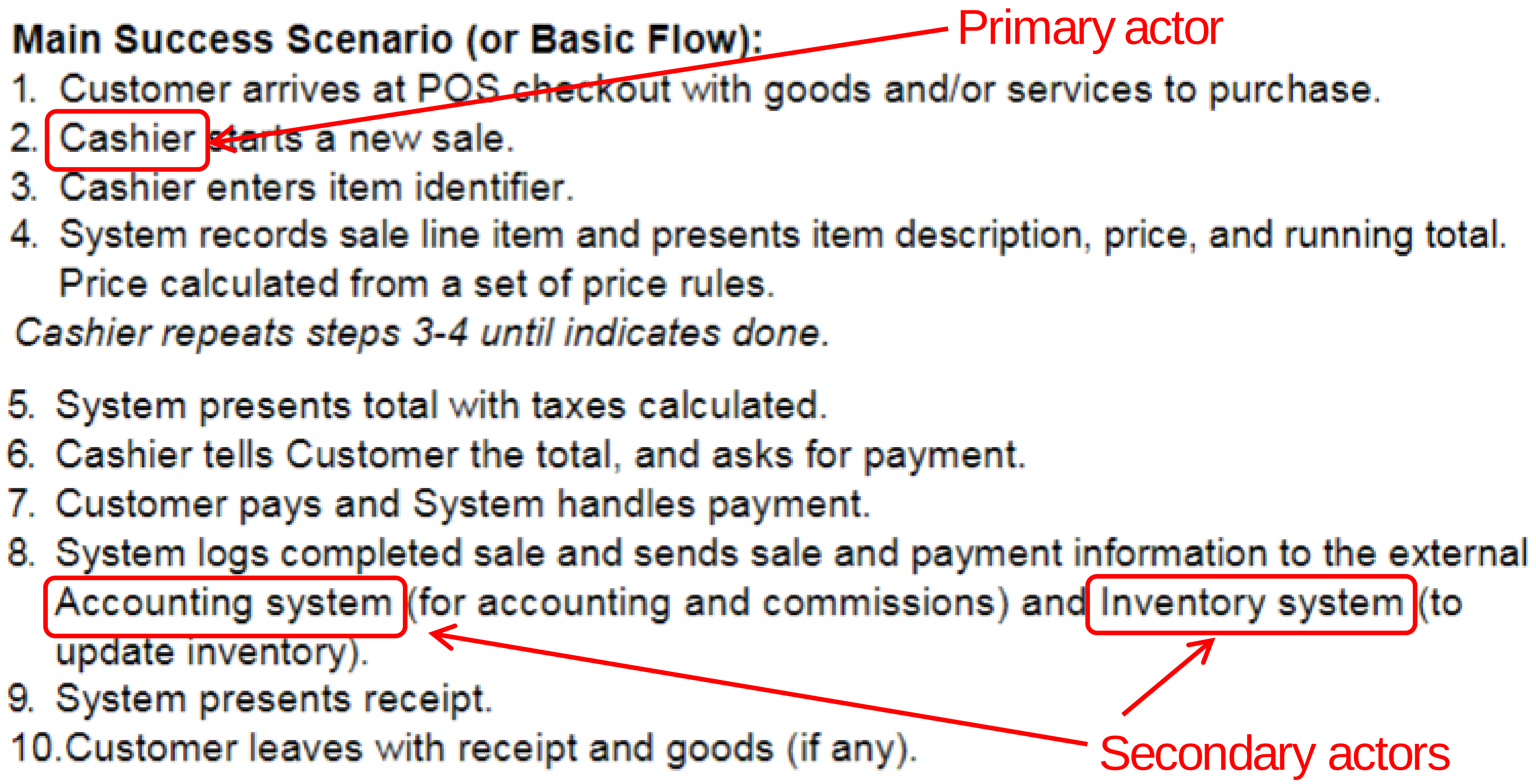
Astah: Vertical Partition



Swim-lanes – shows who are responsible for doing the activities



Example: Process Sale (Larman, section 6.6)



Main Success Scenario (or Basic Flow):

1. Customer arrives at 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. Price calculated from a set of price rules.

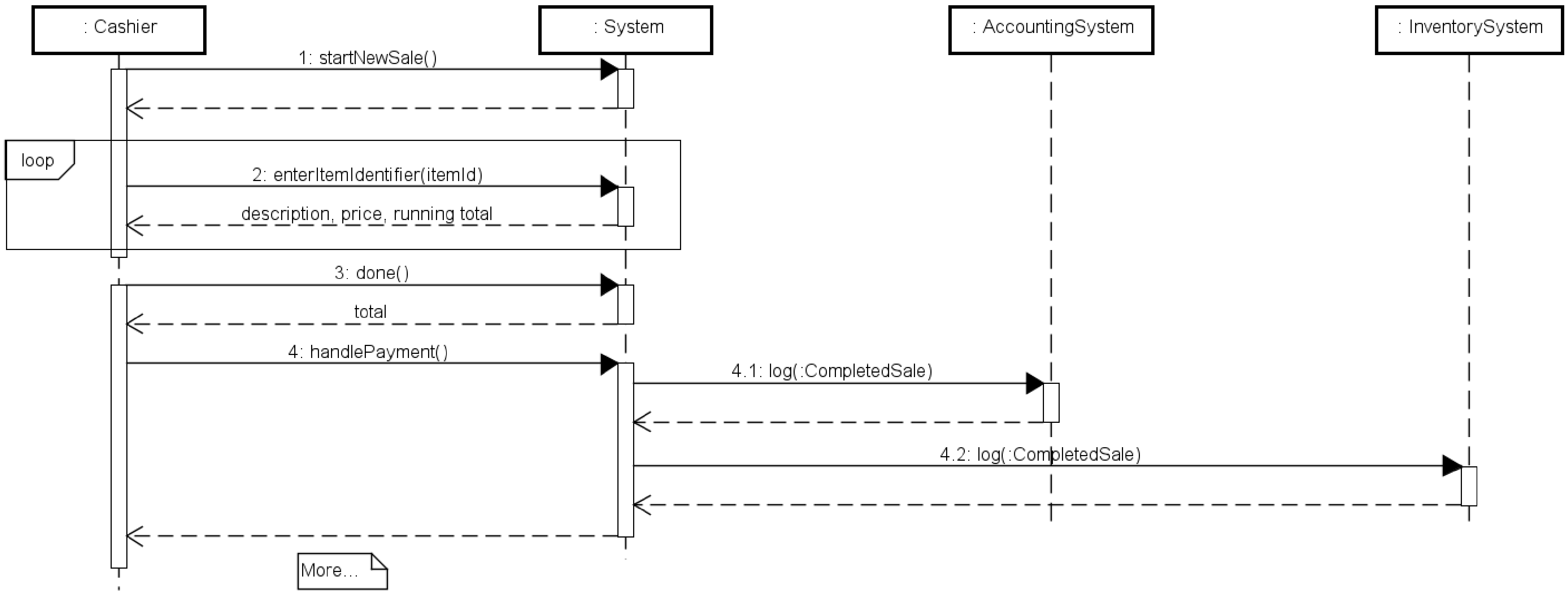
Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

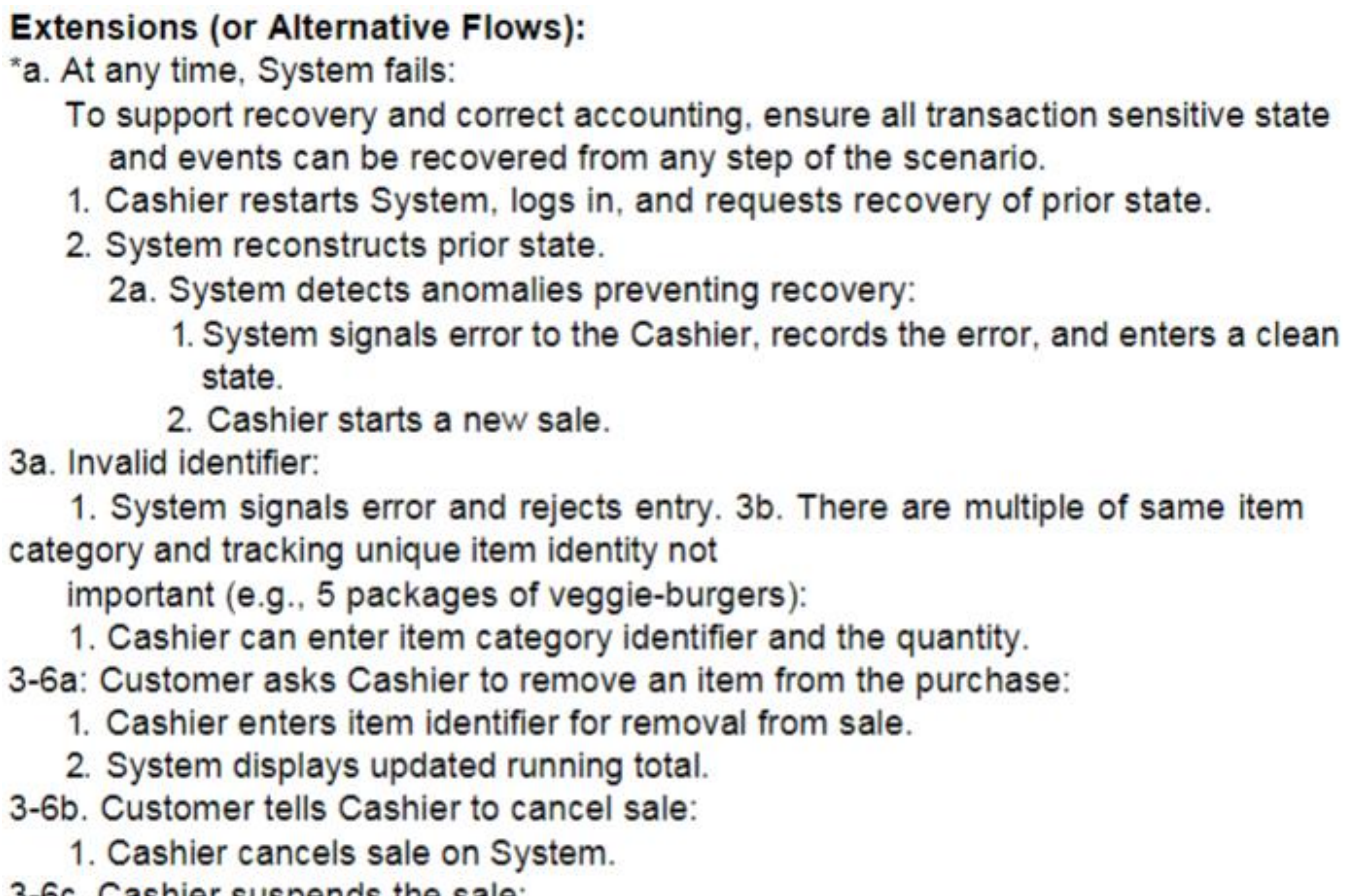
Primary actor

Secondary actors

Example: Process Sale



Example: Process Sale (Larman, section 6.6)



Extensions (or Alternative Flows):

*a. At any time, System fails:

To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.

1. Cashier restarts System, logs in, and requests recovery of prior state.
2. System reconstructs prior state.

    2a. System detects anomalies preventing recovery:

    1. System signals error to the Cashier, records the error, and enters a clean state.
    2. Cashier starts a new sale.

3a. Invalid identifier:

1. System signals error and rejects entry. 3b. There are multiple of same item category and tracking unique item identity not
important (e.g., 5 packages of veggie-burgers):
1. Cashier can enter item category identifier and the quantity.

3-6a: Customer asks Cashier to remove an item from the purchase:

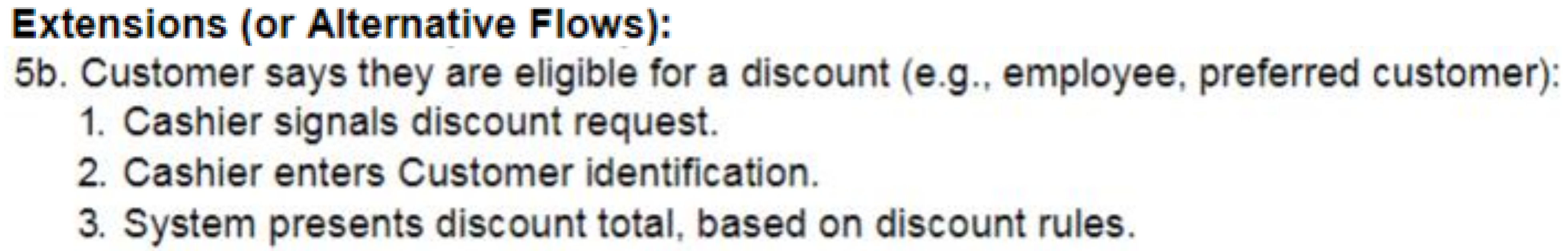
1. Cashier enters item identifier for removal from sale.
2. System displays updated running total.

3-6b. Customer tells Cashier to cancel sale:

1. Cashier cancels sale on System.

3-6c. Cashier suspends the sale:

Example: Process Sale (Larman, section 6.6)

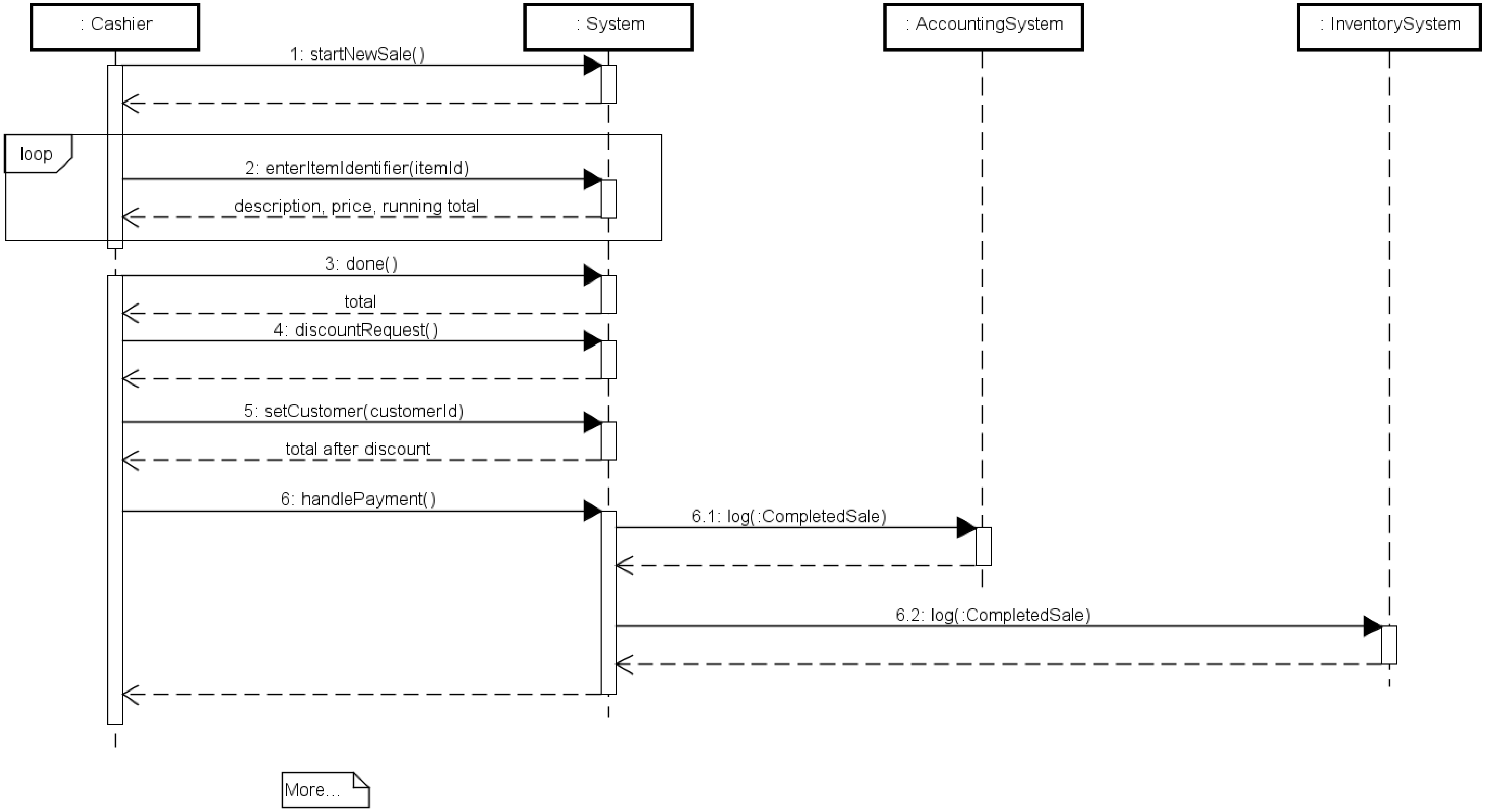


Extensions (or Alternative Flows):

5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):

1. Cashier signals discount request.
2. Cashier enters Customer identification.
3. System presents discount total, based on discount rules.

Example: Process Sale


: Cashier
: System
: AccountingSystem
: InventorySystem
1: startNewSale()
loop
2: enterItemIdentifier(itemId)
description, price, running total
3: done()
total
4: discountRequest()
5: setCustomer(customerId)
total after discount
6: handlePayment()
6.1: log(:CompletedSale)
6.2: log(:CompletedSale)
More…