

## 4. Applying UML

Subject/Topic/Focus:

- Case Study of a Point of Sale System

Summary:

- Use Cases
- Class Diagrams
- Interaction Diagrams

Literature:

- Craig Larman; Applying UML and Patterns, Prentice Hall, 1997

## Case Study: POST

### Point-of-Sale Terminal

- A point-of-sale terminal (POST) system is a computerized system used to record sales and handle payments.
- It is typically used in a retail store.
- It includes hardware components, i.e., a computer, a bar code scanner and software to run the system.
- The POST application is representative for many information systems and touches upon common problems, developers may encounter.
- This case study demonstrates an iterative incremental process, covering the analysis and design phases.



## POST: Inception Phase

During the inception phase the basic requirements of the development process are defined.

- **Overview statement:**  
The purpose of this project is to create a POST system to be used in retail sales.
- **Customer:**  
ObjectStore, Inc., a multinational object retailer.
- **Goal:**  
The general goal is to increase checkout automation, to support faster, better and cheaper services and business processes.
  - Quick checkout for the customer.
  - Fast and accurate sales analysis.
  - Automatic inventory control.

## POST: Elaboration, Use Cases(1)

- To improve the understanding of requirements, use cases are created to receive narrative descriptions of the requested software system.
- There are two approaches to identify use cases that are intertwined:
  - Actor-based method:
    - Identify the actors related to a system or organization.
    - For each actor, identify the processes they initiate or participate in.
  - Event-based method:
    - Identify the external events that a system must respond to.
    - Relate the events to actors and use cases.

## POST: Elaboration, Use Cases(2)

Applying the actor-based method for identifying the appropriate use cases in the POST system example, the following actors and use cases were found:

Actor	Event
Cashier	Log in Cash out
Customer	Buy items Refund items

## POST: Elaboration, Use Cases (3)

The description below is a textual presentation (i.e., derived from a customer interview) of the high level use case *buy items*.

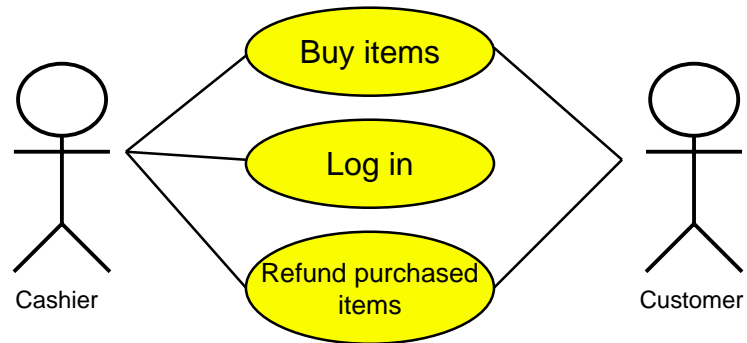
Use case:	Buy items.
Actors:	Customers, cashier.
Type:	Primary.
Description:	A customer arrives at a checkout with items to purchase. The cashier records the purchase items and collects payment. On completion the customer leaves with items.

Type is basis for ranking:

- primary: central, often needed
- secondary: sometimes needed
- optional: nice to have

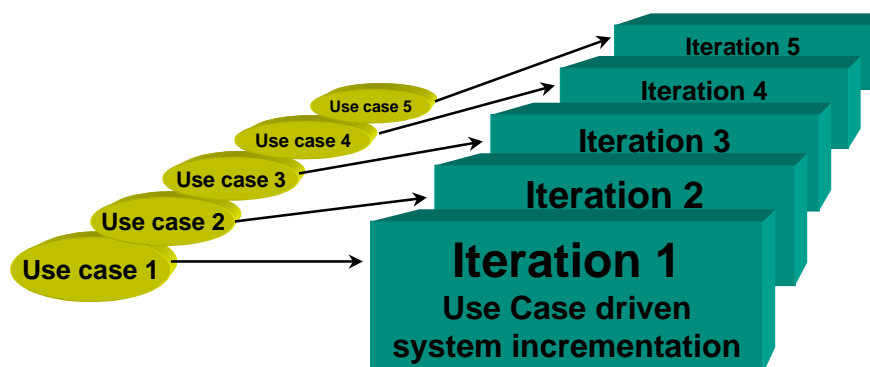
## Use Case Diagram

A partial use case diagram:



## Use Cases and Iterations

The use cases are assigned to proceeding iterations.



The order in which use cases are assigned is determined by scheduling and ranking of the individual use case.

## Scheduling Use Cases

- During the iterative process of software engineering, use cases are assigned to development cycles.
- **Ranking** can be done by weighting several qualities:
  - a: significant impact on the architectural design,
  - b: significant information and insight regarding the design is obtained with relatively little effort
  - c: risky, time-critical or complex functionality
  - d: significant research or new technology
  - e: primary line-of-business processes

Use Case	a	b	c	d	e	sum
Buy Items	5	3	2	0	5	15
:						

## Ranking in the POST Application

Based on the prior ranking criteria, a fuzzy and informal ranking of the sample point-of-sale application use cases.

Rank	Use Case	Justification
High	Buy items	High scores on most ranking criteria
Medium	Add new user Log in Refund Items	Affects security subdomain Affects security subdomain Important process; affects accounting
Low	Cash out Start up Shut down	Minimal effect on architecture Definition depends on other use cases Minimal effect on architecture

## Conceptual Perspective: Class Diagram

- After the use cases are scheduled and ranked, the next step within an iteration is to design classes, their qualities and the relationships among them.
- To identify classes and create a class diagram:
  - list the candidate classes using the noun phrase identification related to the current requirements under consideration,
  - list the identified classes and draw them into the class diagram,
  - add the associations necessary to record relationships among classes,
  - add the attributes to fulfill the information requirements.

## Class Diagram: Noun Phrase List

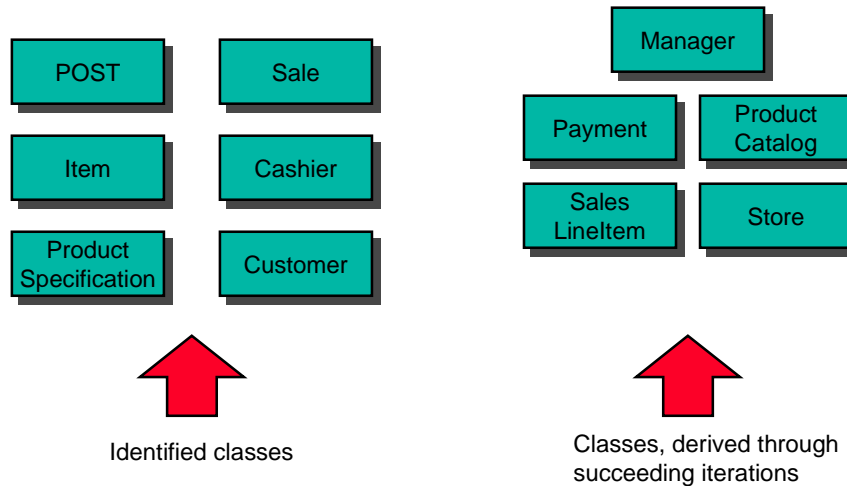
Noun phrase analysis on the *buy item* use case:

### Use Case: Buy item

1. This use case begins when a **customer** arrives at a **POST checkout** with **items** to purchase.
2. The **cashier** records the **universal product code** (UPC) from each **item**.
3. The System determines the **item price** and adds the item to the running **sales transaction**.
4. If there is more than one of the same **item**, the **cashier** can enter the **quantity** as well.
5. The **description** and the **price** of the current item are displayed.

## Identified Classes

Classes, identified by the **noun phrase** method:



OOA&D © J.W. Schmidt, F. Matthes, TU Hamburg-Harburg

4.13

## Class Diagram: Associations

- To satisfy the information requirement of the use cases under development, it is necessary to identify the associations between classes.
- The technique of an **association list** is helpful for identifying relationships:

Category	Example
A is a physical part of B	Drawer - POST
A is a logical part of B	SalesLineltem - Sale
A is physically contained in B	POST - Store
A is a logically contained in B	ItemDescription - Catalog
A is a description for B	POST - Store
A uses or manages B	Cashier - POST
A communicates with B	Customer - Cashier

OOA&D © J.W. Schmidt, F. Matthes, TU Hamburg-Harburg

4.14

## A Basic Class Diagram

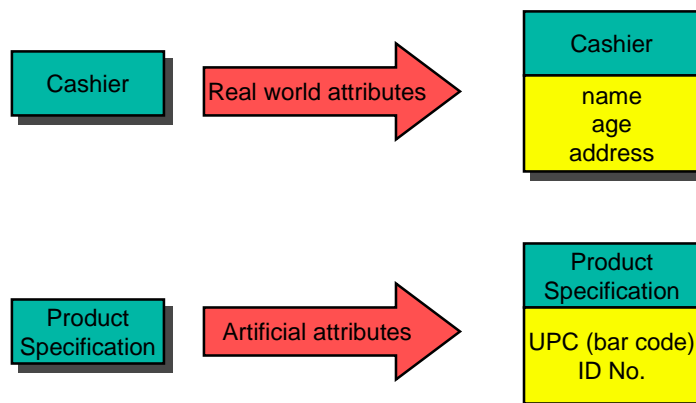
A basic, coarse grain class diagram is derived by taking the association list into account.

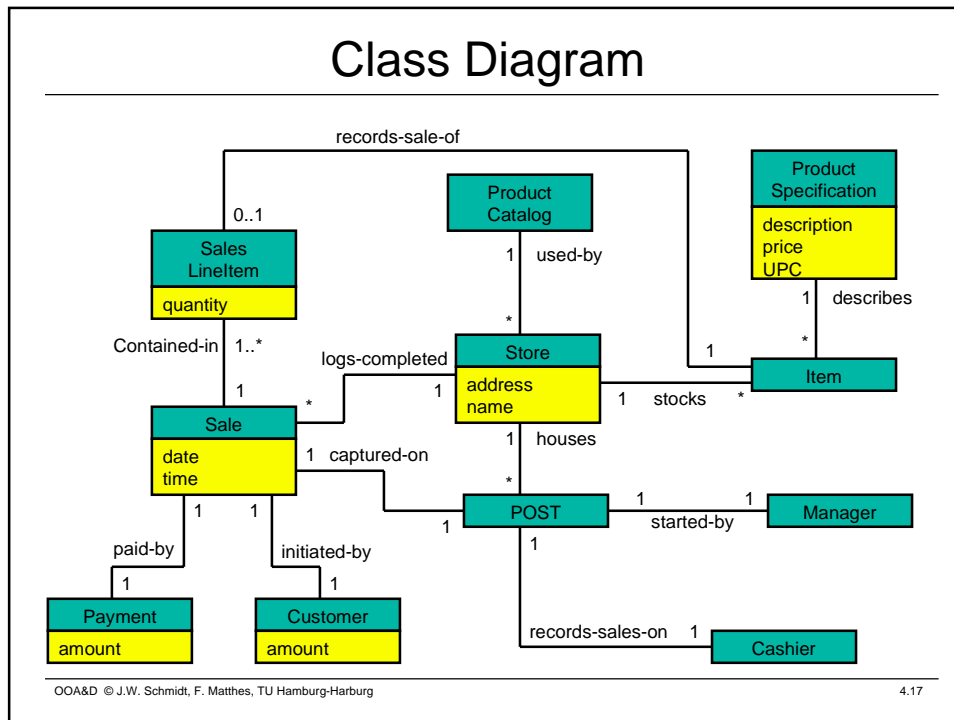


Not every association mentioned in the association list is required!

## Adding Attributes

To find attributes, the real-world attributes of objects can be used as well as the attributes necessary to fulfill the requirements defined by the use case under consideration.





## Development of Dynamic Behavior

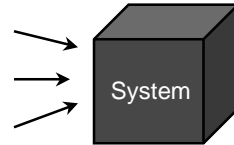
- Use cases suggest the system events which are explicitly shown in system interaction diagrams.
- An initial, textual description of the system events and their goal is presented by operation contracts.
- The system events represent messages that initiate interaction diagrams, which illustrate how objects interact to fulfill the required tasks.

The assignment of responsibilities and development of interaction diagrams is the most significant **creative** step during the design phase!

## Dynamic Behavior: Perspective

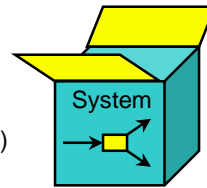
○ The dynamic behavior of a use case viewed from the **conceptual perspective** shows:

- actors of a use case,
- interactions of the actors with the system
- the system as a **black box**.



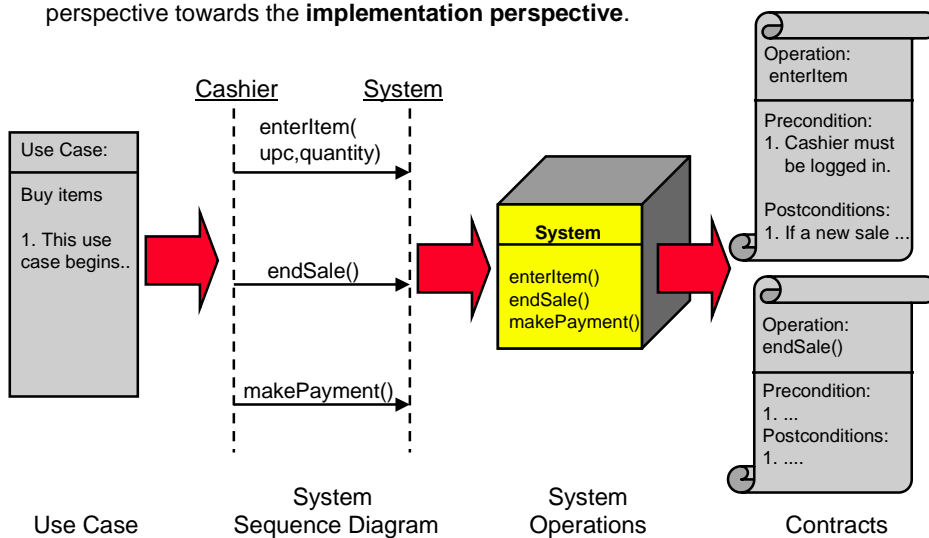
○ The **implementation perspective** of a use case shows

- the requests from outside (i.e., system operations),
- the internal messages and system events,
- the system as **structured components** (e.g., classes) involved in the process.



## Use Cases and Interaction Diagrams: Overview

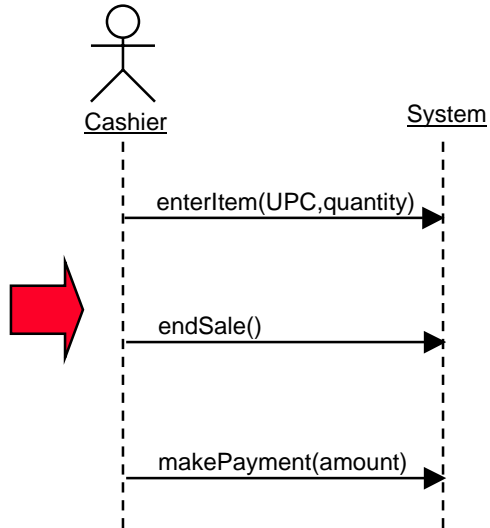
The dynamic behavior of a use case and the transition from conceptual perspective towards the **implementation perspective**.



## Use Case and System Interaction

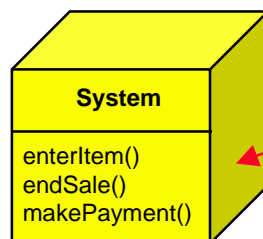
**USE CASE: BUY ITEMS**  
Typical course of events:

1. This use case begins when a Customer arrives at the POST checkout with items to purchase.
2. The cashier records the universal product code (UPC) from each item. If there is more than one of the same item, the cashier can enter the quantity as well.
3. System determines the item price and adds the item information to the running sales transaction. The description and the price of the current item are displayed



## Contracts

- The system sequence diagram shows the system events that an external actor generates.
- It does not elaborate on the details of the functionality associated with the system operations invoked.
- In general a contract is a document that describes what an operation commits to achieve.
- A system operation contract describes changes in the state of the overall system when a system operation is invoked.



Contracts are written for each system operation to describe its behavior.

## Contracts: Example

**Name:** enterItem(upc : number , quantity : integer )

**Responsibilities:** Enter (record) sale of an item and add it to the sale. Display the item description and price.

**Type:** System

**Cross References:** System Functions: ...  
Use Cases: ...

**Notes:**

**Exceptions:** If the UPC is not valid, indicate that it was an error.

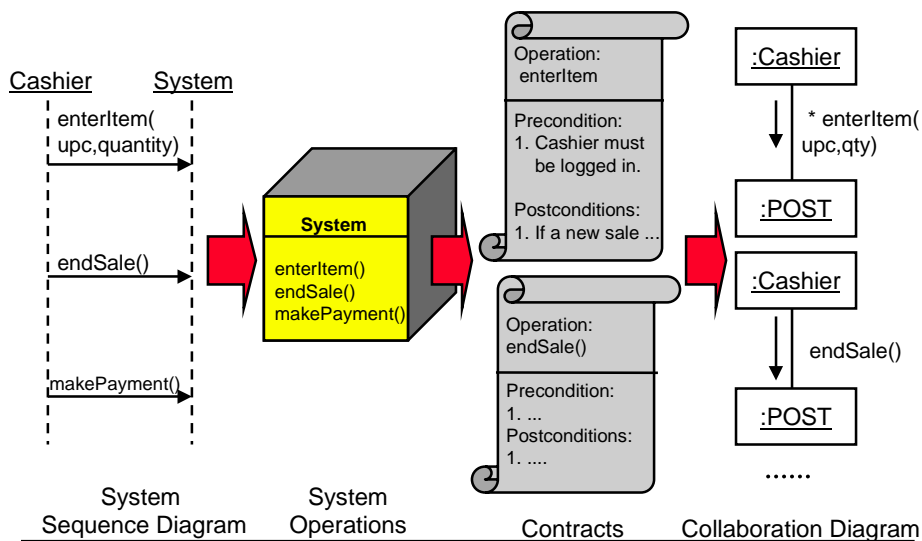
**Output:**

**Pre-conditions:** UPC is known to the system.

**Post-conditions:** Product and quantity are added to the sale transaction or an error is displayed.

## Contracts and Collaboration Diagrams

Transition from contracts to collaboration diagrams:



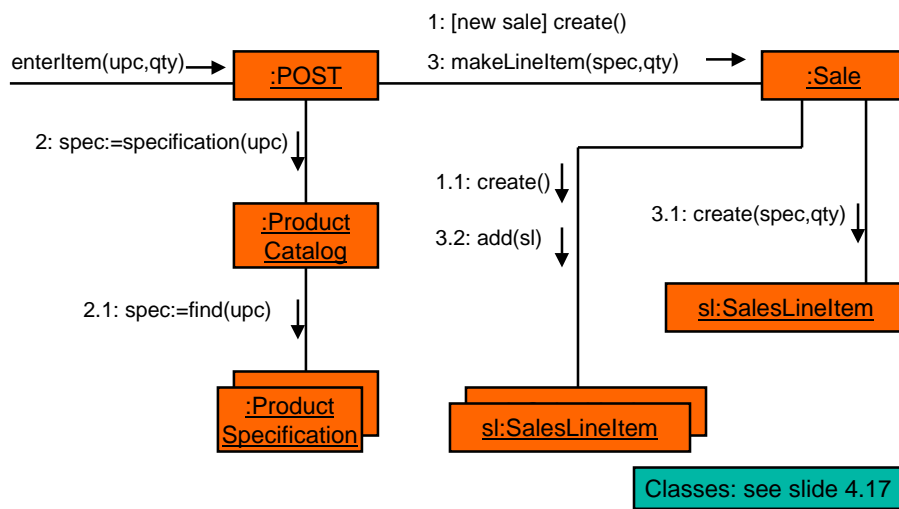
## Assignment of Responsibility

- An important yet difficult task is to choose the appropriate class out of the set of classes which are involved in the realization of the system operation.



## Collaboration Diagram: Example

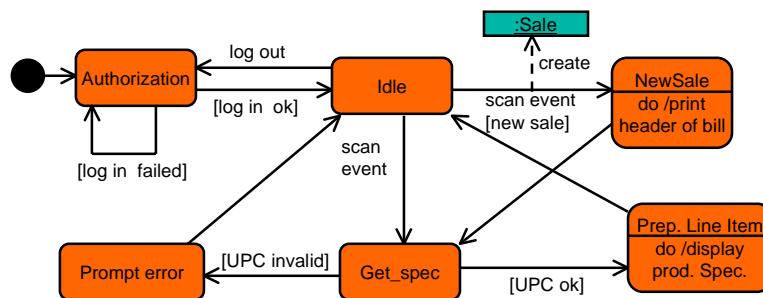
The collaboration diagram for the system operation: enterItem(upc,qty)



## State Diagrams

- State diagrams show states and the corresponding transitions of objects.
- State diagrams are used to present details on internal behavior of objects.
- State diagrams can be applied to software classes as well as to use cases.

State diagram for classes, example: *:POST* instance

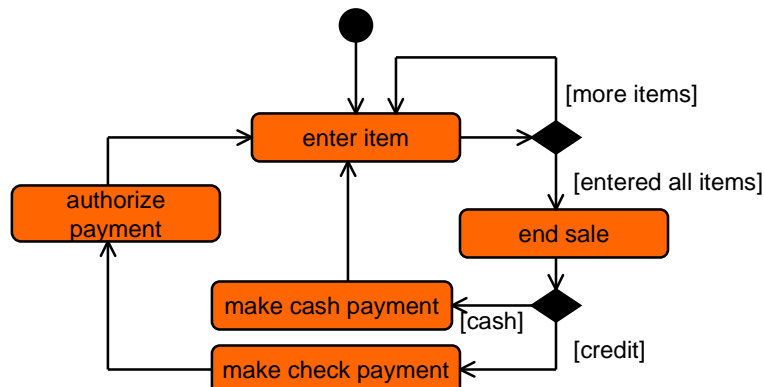


OOA&D © J.W. Schmidt, F. Matthes, TU Hamburg-Harburg

4.27

## Activity Diagrams

Activity diagrams for use cases, example: *Buy items*

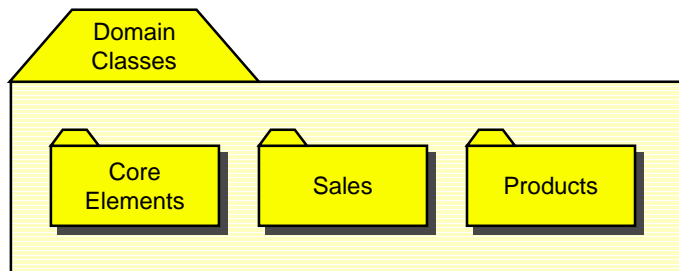


OOA&D © J.W. Schmidt, F. Matthes, TU Hamburg-Harburg

4.28

## Package Diagrams (1)

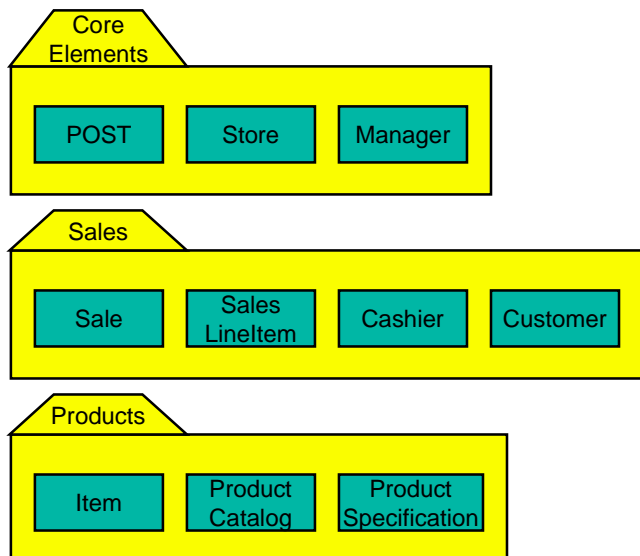
- Package diagrams are a valuable tool for grouping logically depending elements together.
- For packaging, place those objects together which
  - are in the same subject area - closely related by concept or purpose,
  - are in a type hierarchy,
  - participate in the same use case,
  - are strongly associated.



OOA&D © J.W. Schmidt, F. Matthes, TU Hamburg-Harburg

4.29

## Package Diagrams (2)



OOA&D © J.W. Schmidt, F. Matthes, TU Hamburg-Harburg

4.30