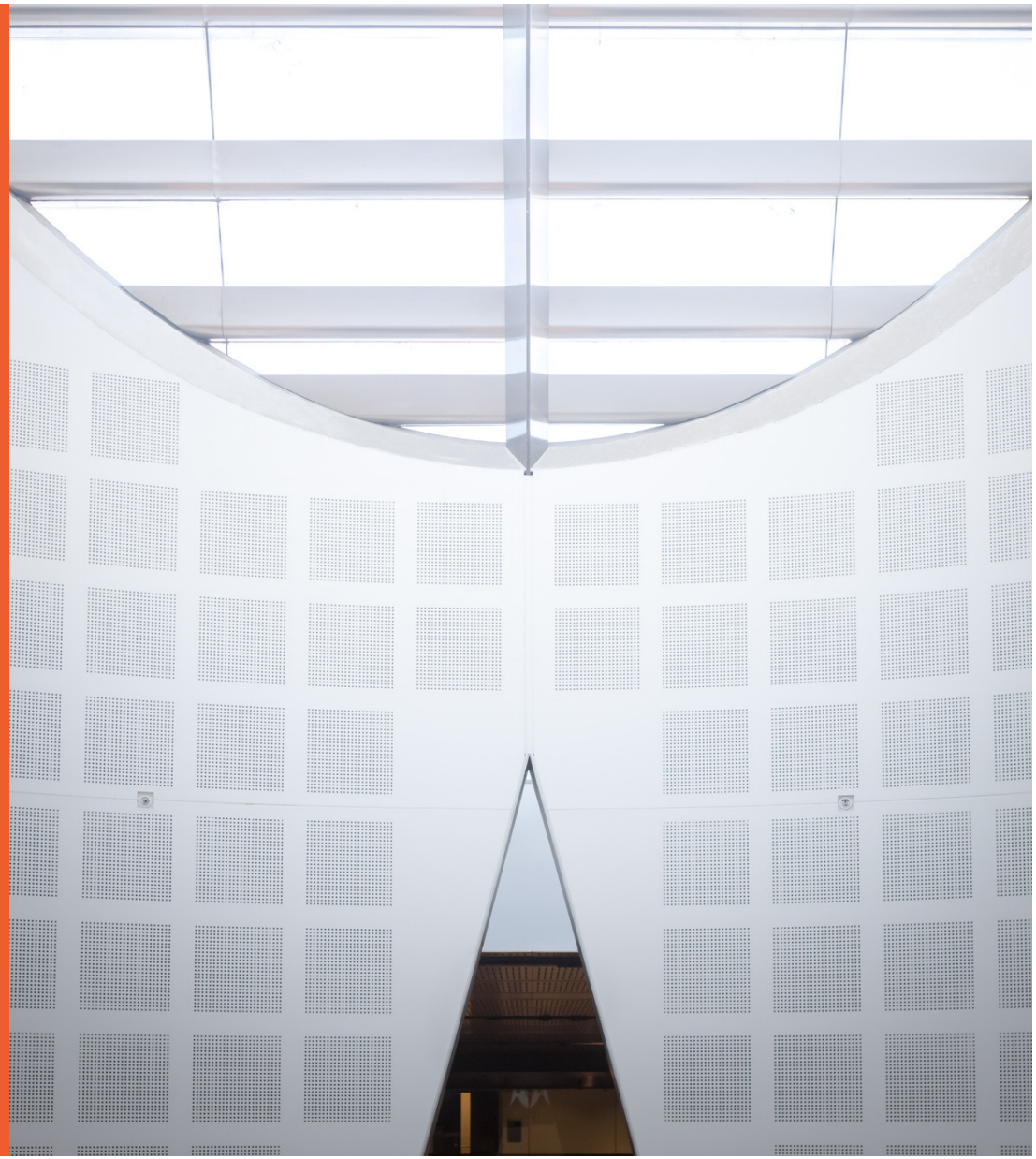
Software Design and Construction 1
SOFT2201 / COMP9201

Self Learning Case Study:
Next Gen Point-of-Sale (POS) System

School of Computer Science

THE UNIVERSITY OF SYDNEY



Copyright warning

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Software Modelling Case Study

NextGen POS software modeling

Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition).*

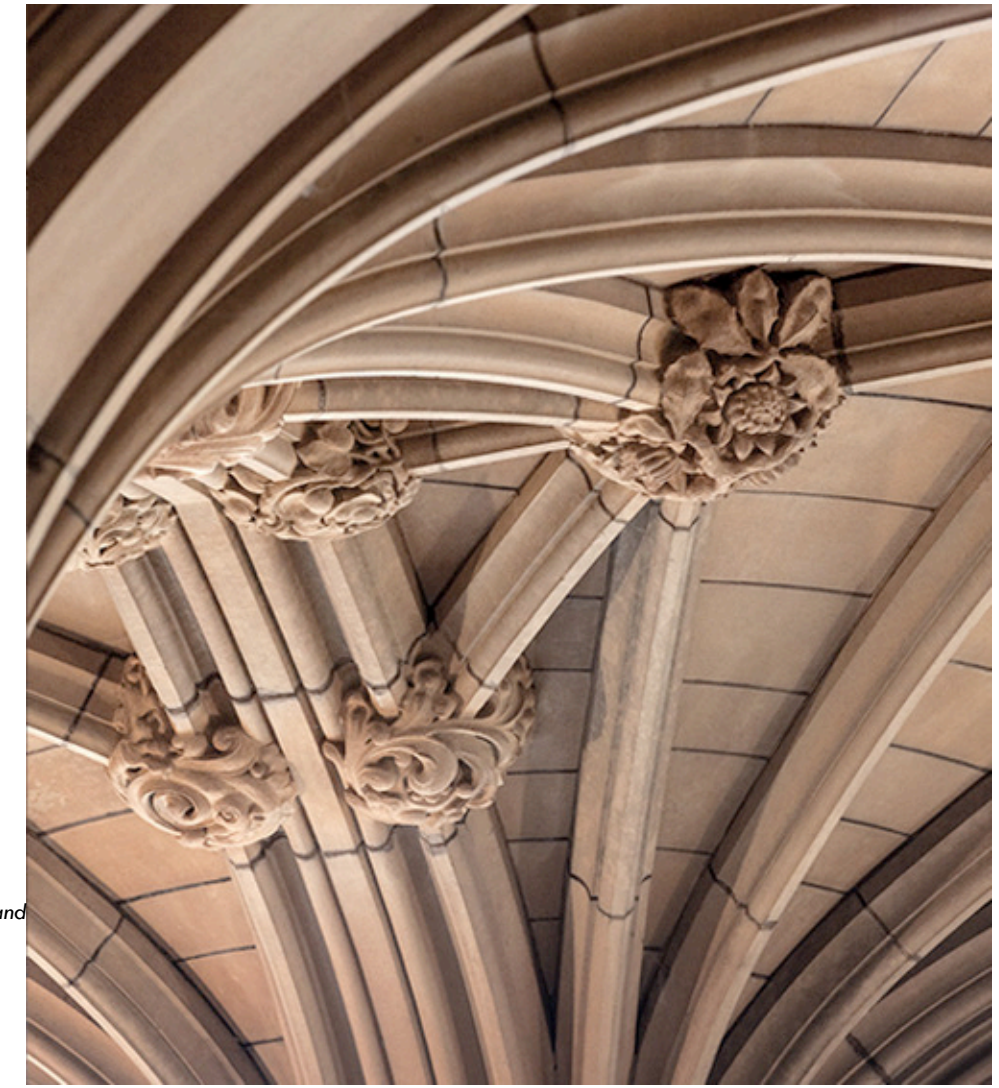
THE UNIVERSITY OF SYDNEY

Next Gen Point-of-Sale (POS) System

- A POS is a computerized application used (in part) to record sales and handle payments
  - Hardware: computer, bar code scanner
  - Software
  - Interfaces to service applications: tax calculator, inventory control
  - Must be fault-tolerant (can capture sales and handle cash payments even if remote services are temporarily unavailable
  - Must support multiple client-side terminals and interfaces; web browser terminal, PC with appropriate GUI, touch screen input, and Wireless PDAs
  - Used by small businesses in different scenarios such as initiation of new sales, adding new line item, etc.

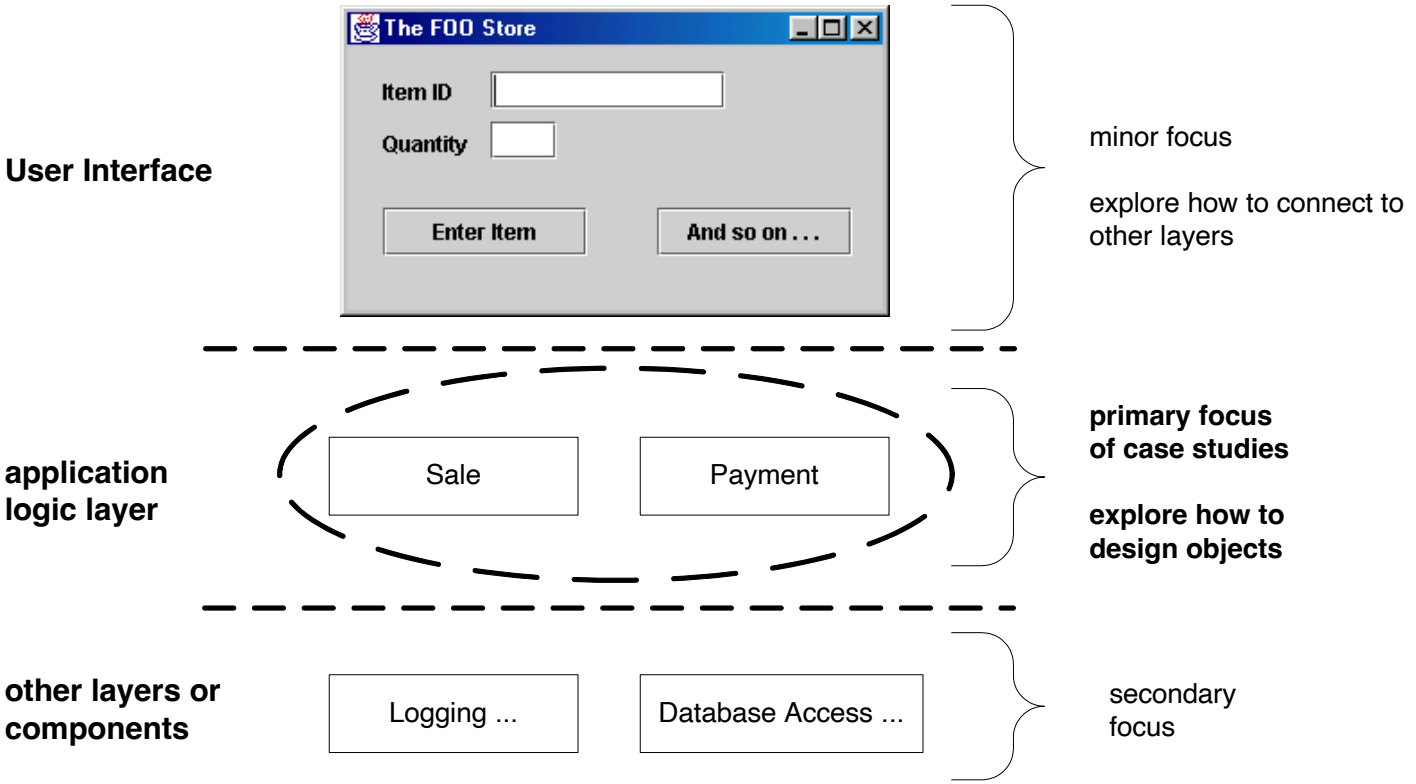
Next Gen POS Analysis

Scope of OOA & D and Process Iteration

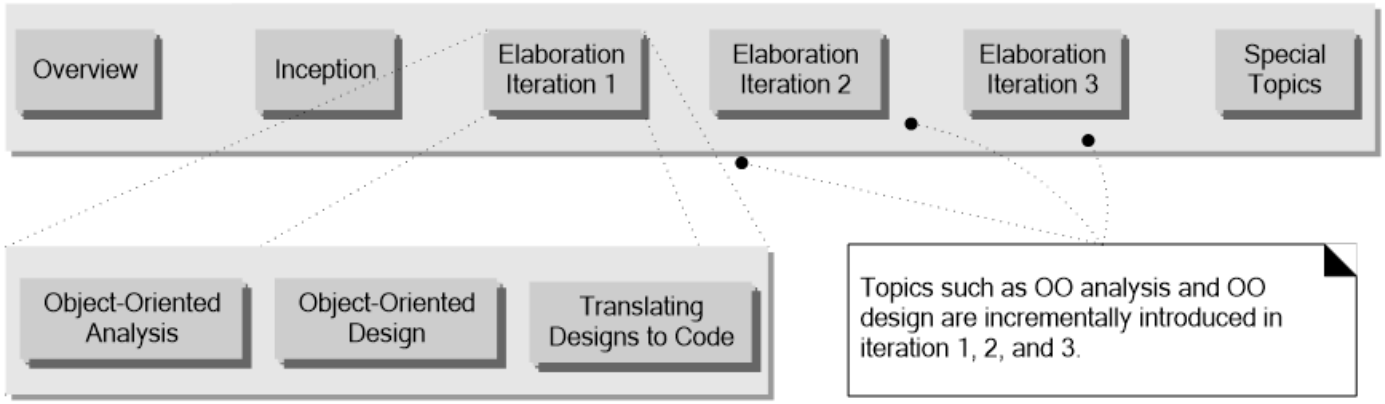


Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*.

Next Gen POS – Scope (Analysis & Design)



Iteration and Scope – Design and Construction



Iteration 1

Introduces just those analysis and design skills related to iteration one.

Iteration 2

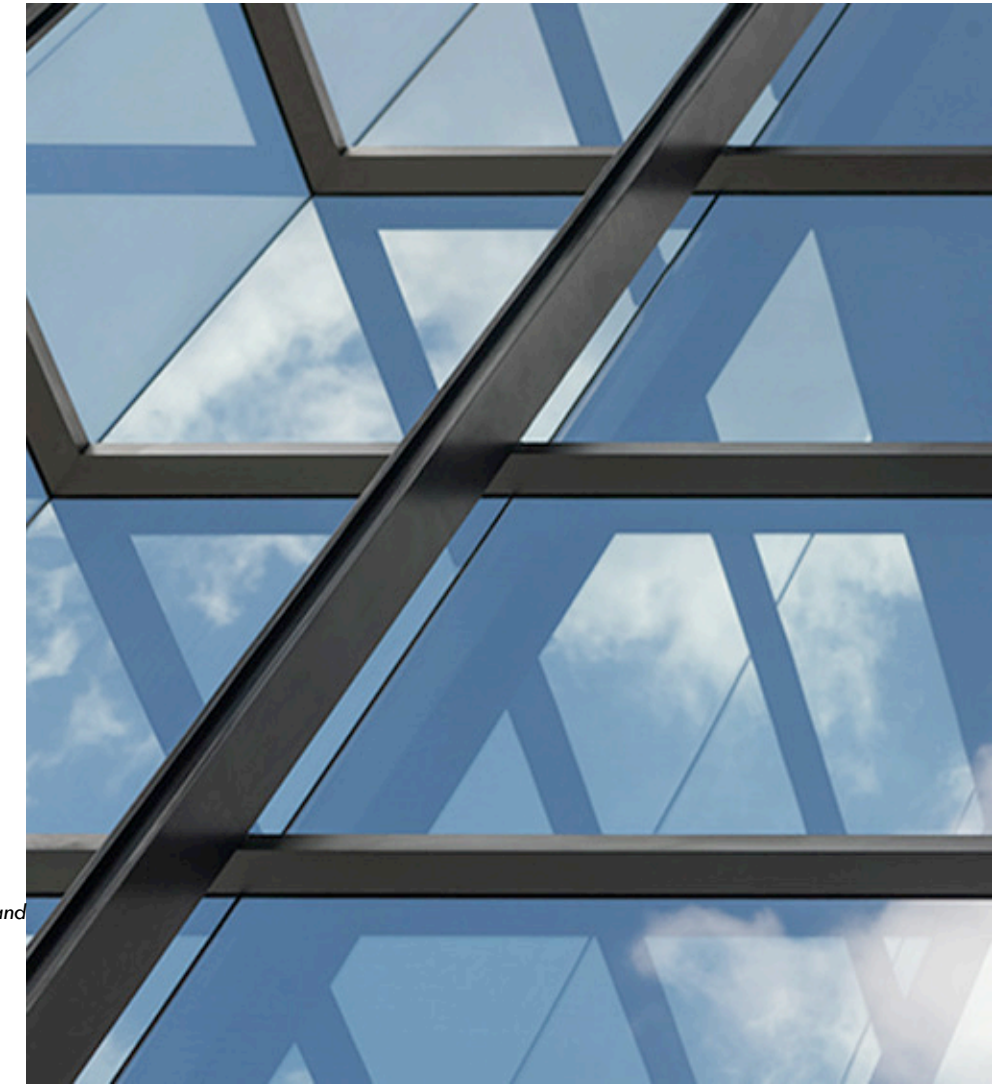
Additional analysis and design skills introduced.

Iteration 3

Likewise.

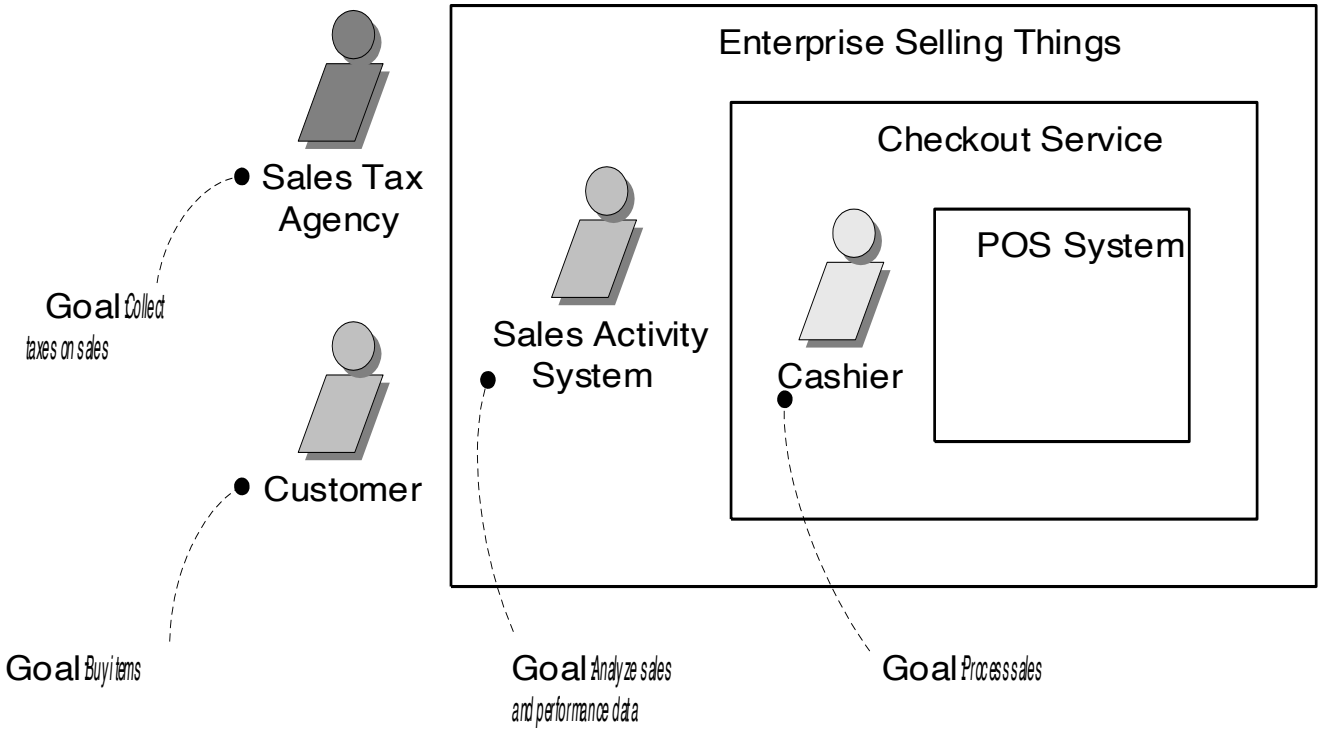
Next Gen POS Case Study: Analysis

OO Analysis with UML



Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*.

Analysis (Requirements): Actors, Goals, System Boundaries



NextGen POS: Process Sale Use Case Description

Use case UC1: Process Sale
Primary Actor: Cashier
Stakeholders and Interests:
-Cashier: Wants accurate and fast entry, no payment errors, …
-Salesperson: Wants sales commissions updated.
…
Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions):
-Sale is saved. Tax correctly calculated.
…
Main success scenario (or basic flow):
Extensions (or alternative flows): [see next slide]
Special requirements: Touch screen UI, …
Technology and Data Variations List:
-Identifier entered by bar code scanner,…
Open issues: What are the tax law variations? …

Main success scenario (or basic flow):
The Customer arrives at a POS checkout with items to purchase.
The cashier records the identifier for each item. If there is more than one of the same item, the Cashier can enter the quantity as well.
The 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 presented.
On completion of item entry, the Cashier indicates to the POS system that item entry is complete.
The System calculates and presents the sale total.
The Cashier tells the customer the total.
The Customer gives a cash payment ("cash tendered") possibly greater
than the sale total.
Extensions (or alternative flows):
If invalid identifier entered. Indicate error.
If customer didn't have enough cash, cancel sales transaction.

Next Gen POS Use Case Diagram

UC1: ***Process Sale***

…

Main Success Scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.

…

7. Customer pays and System handles payment

Extensions:

7b. Paying by credit: Include *Handle Credit Payment.*

7c. Paying by check: Include *Handle Check Payment*

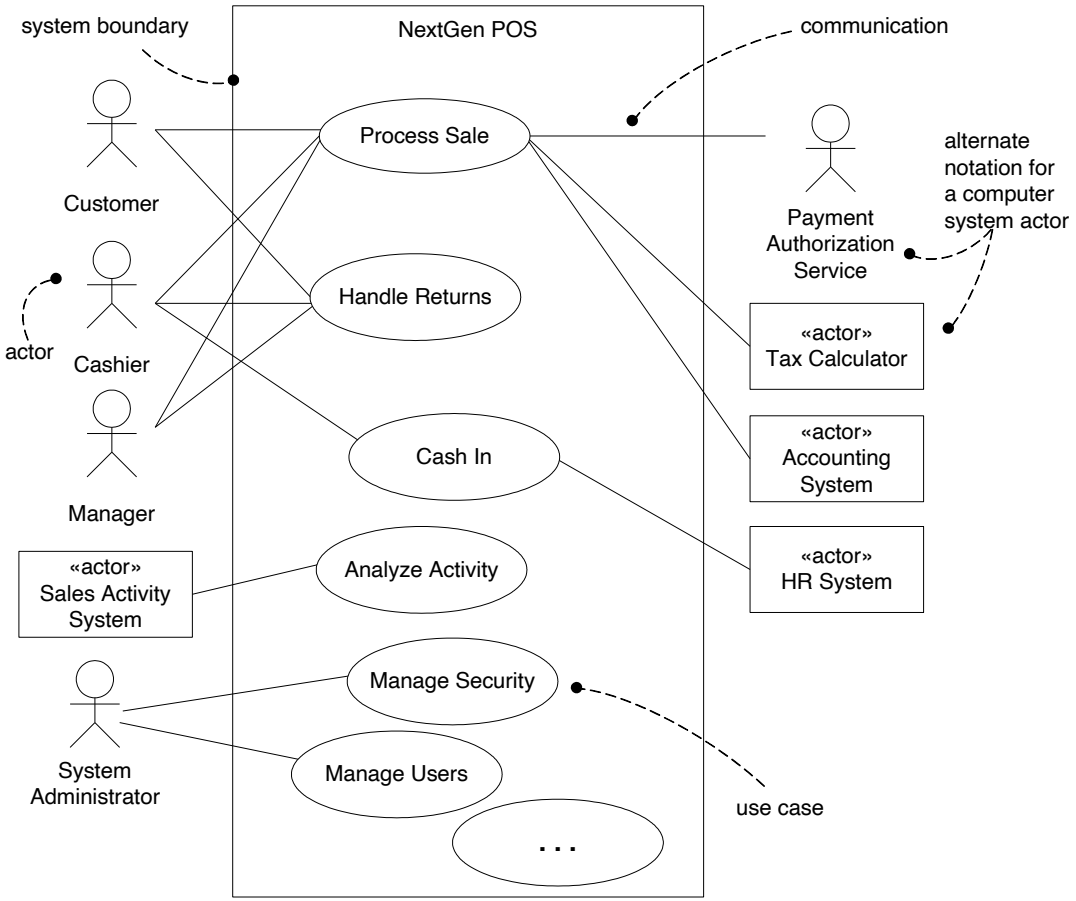
UC7: Process Rental

…

Extensions:

6b. Paying by credit: *Handle Credit Payment.*

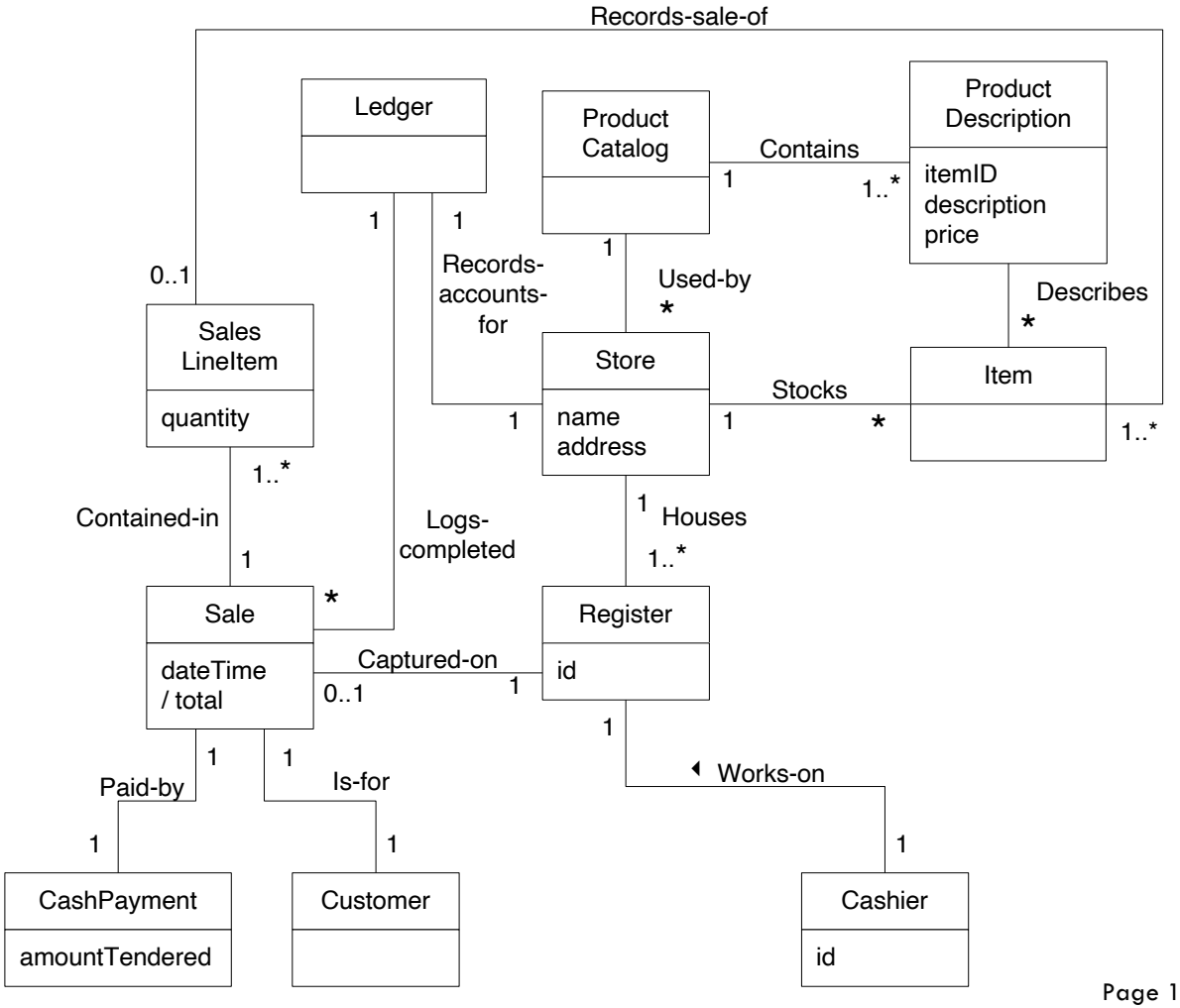
…



NextGen POS Analysis: Domain Model

A conceptual perspective model
Partial domain model drawn with UML class diagram

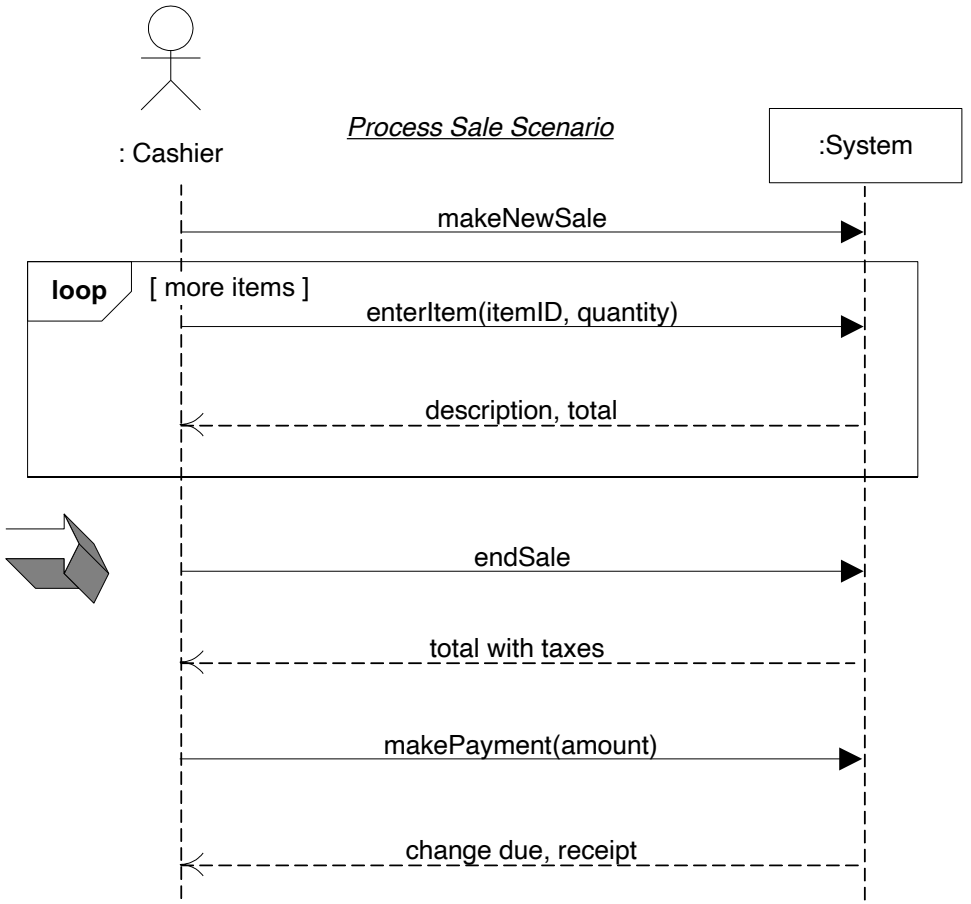
It shows conceptual classes with key associations



NextGen POS Analysis: System Sequence Diagram (SSD)

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.
...



NextGen POS Case Study: Design

OO Design with UML

THE UNIVERSITY OF SYDNEY

Next Gen POS: From Analysis to Design

Requirements Analysis (OOA)

Business modelling – domain models

Use case diagrams

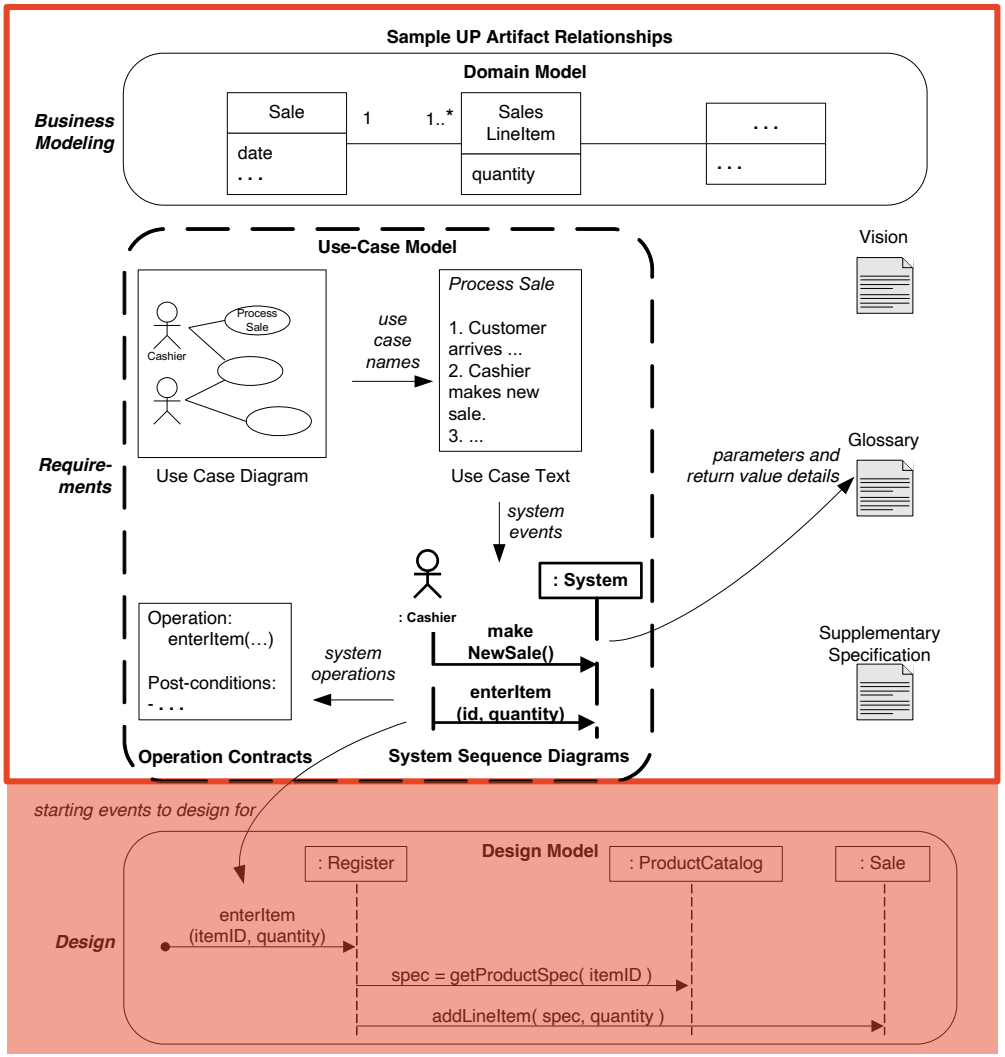
Use case description

System Sequence Diagrams

Design (OOD)

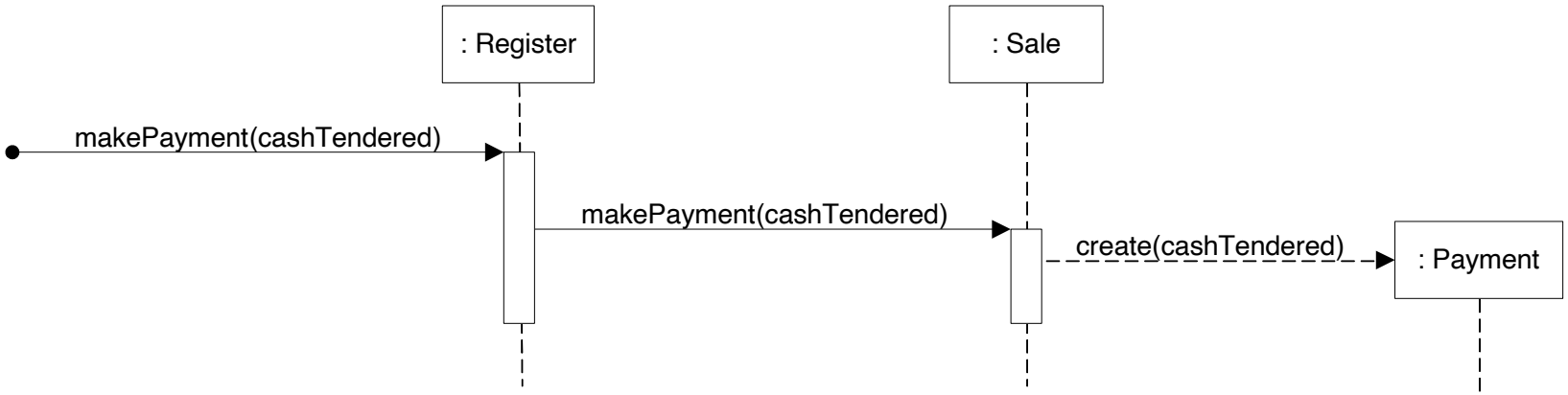
Sequence diagrams

Class diagrams

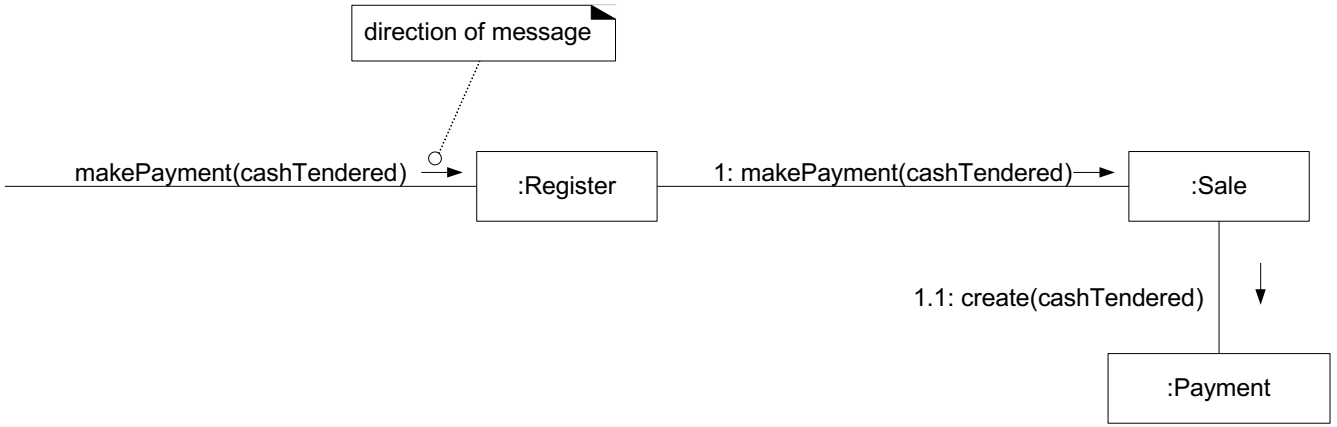


NextGen POS: Interaction Diagrams

Sequence Diagram



Communication Diagram

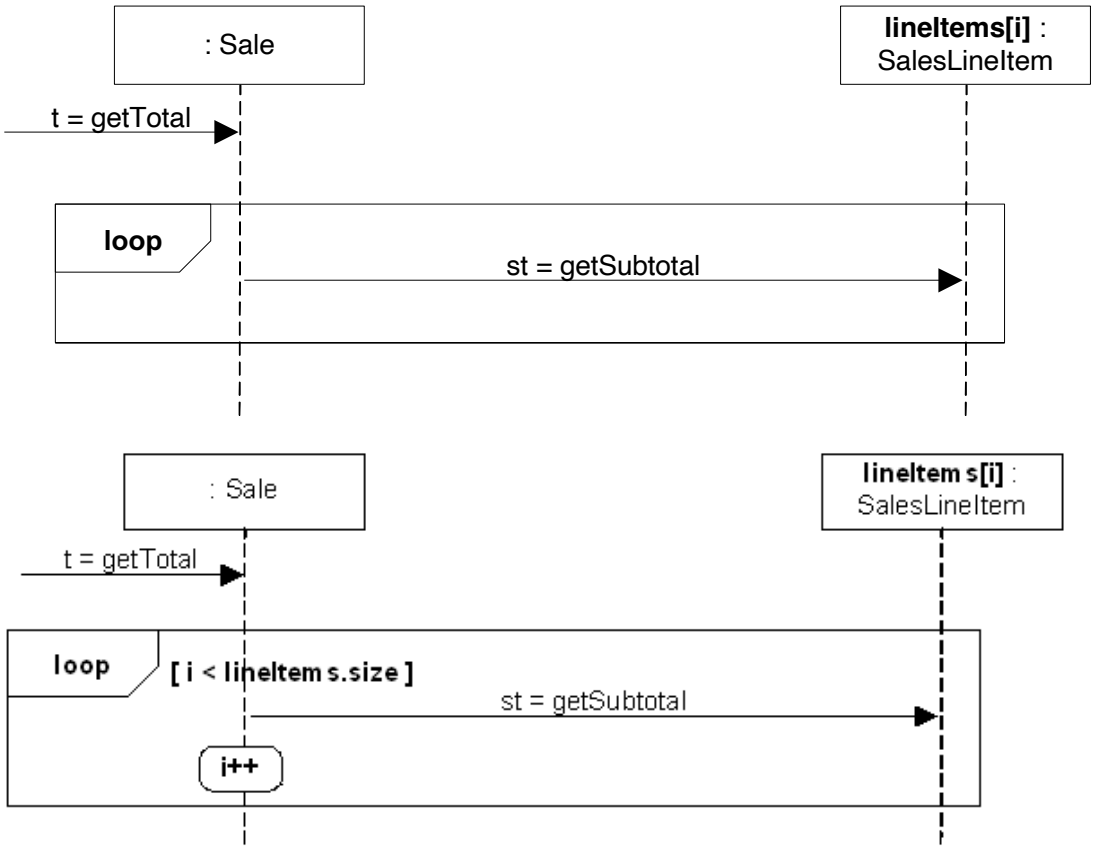


NextGen POS: Sequence Diagrams

1. The message *makePayment* is sent to an instance of a *Register*. The sender is not identified
2. The *Register* instance sends the *makePayment* message to a *Sale* instance.
3. The *Sale* instance creates an instance of a *Payment*.

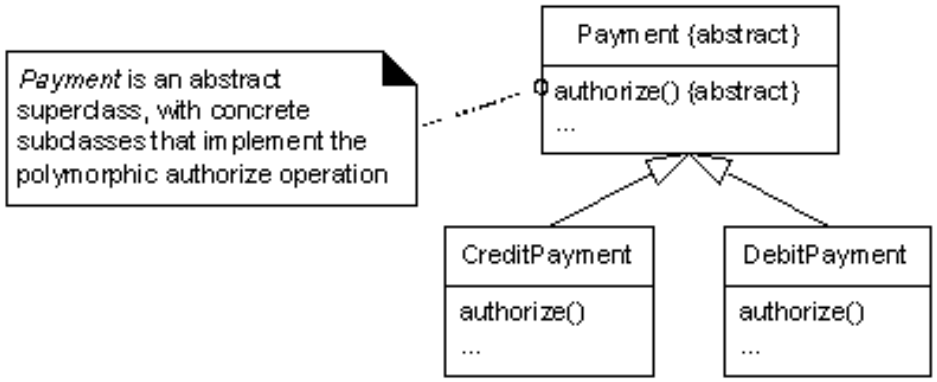
How the skeleton of the Sale class should look like?

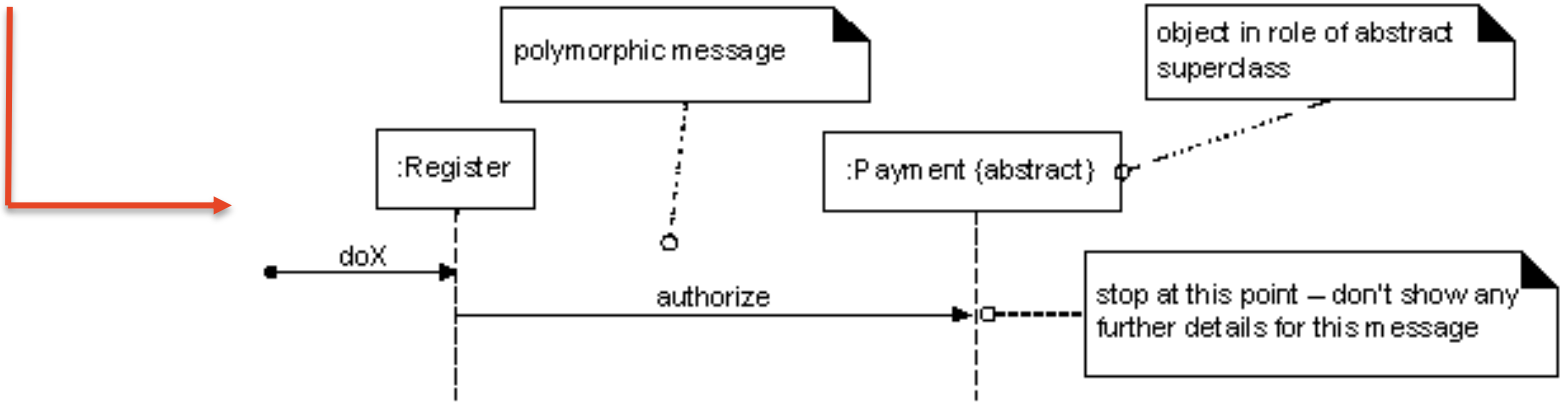
NextGen POS: Sequence Diagram (Iteration)



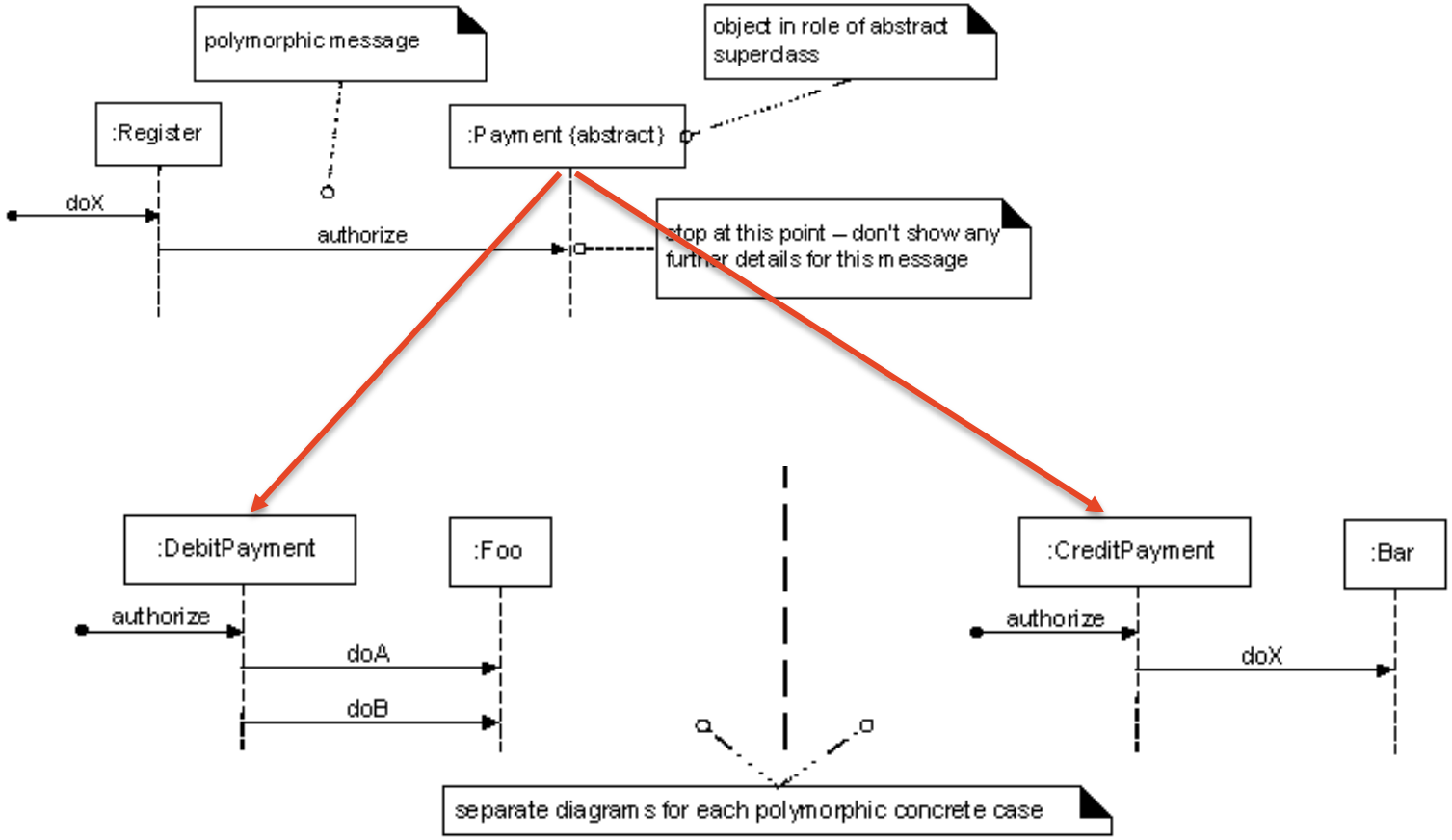

```java
public class Sale {
    private List<SalesLineItem>
            lineItems = new ArrayList<SalesLineItem>;

    public Money getTotal() {
        Money total = new Money();
        Money subtotal = null;

        for (SalesLineItem lineItem : LineItems){
            subtotal = lineItem.getSubtotal();
            total.add(subtotal);
        }
        return total;
    }
    // ...
}
```

NextGen POS: Polymorphic Messages

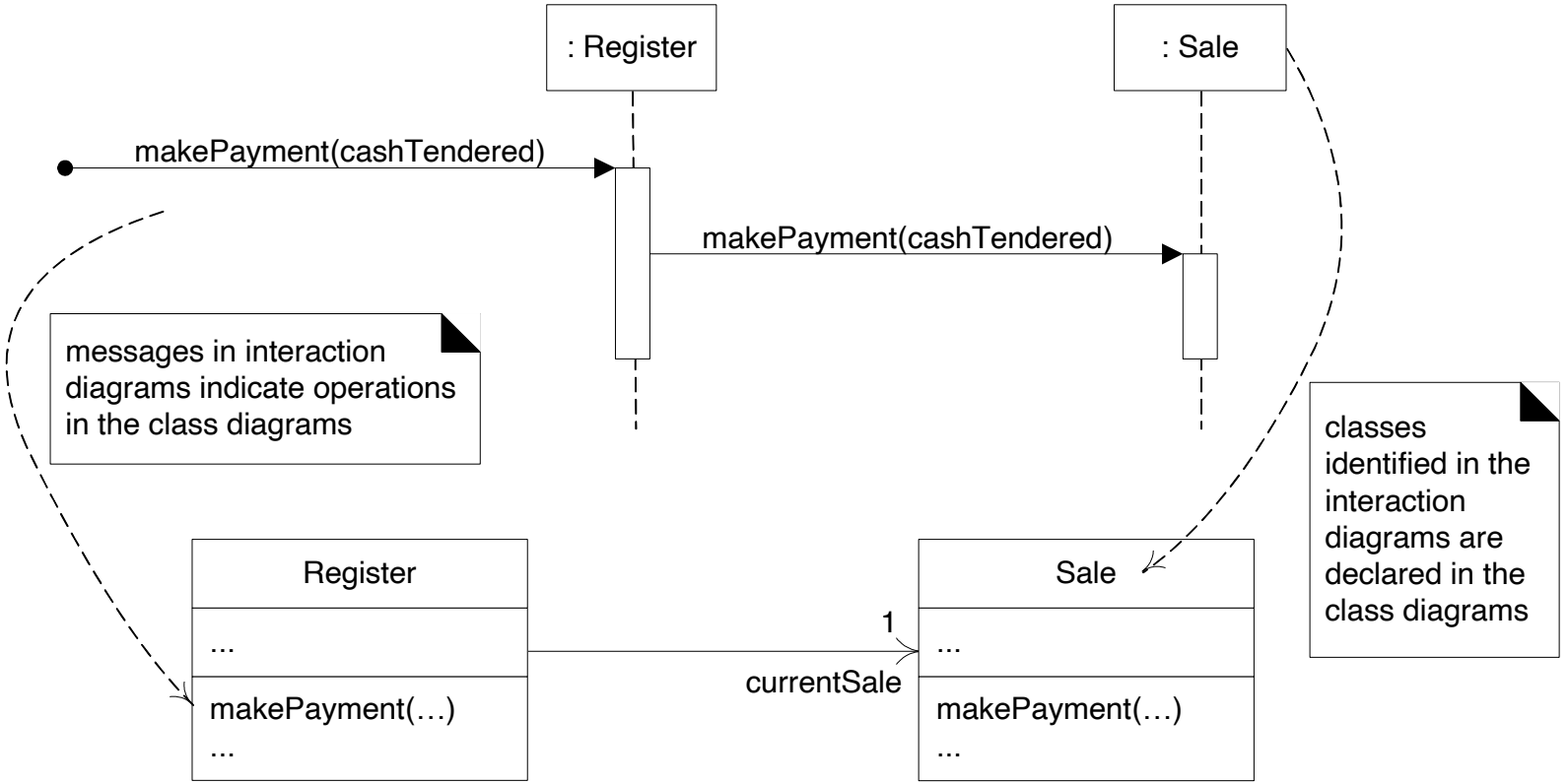




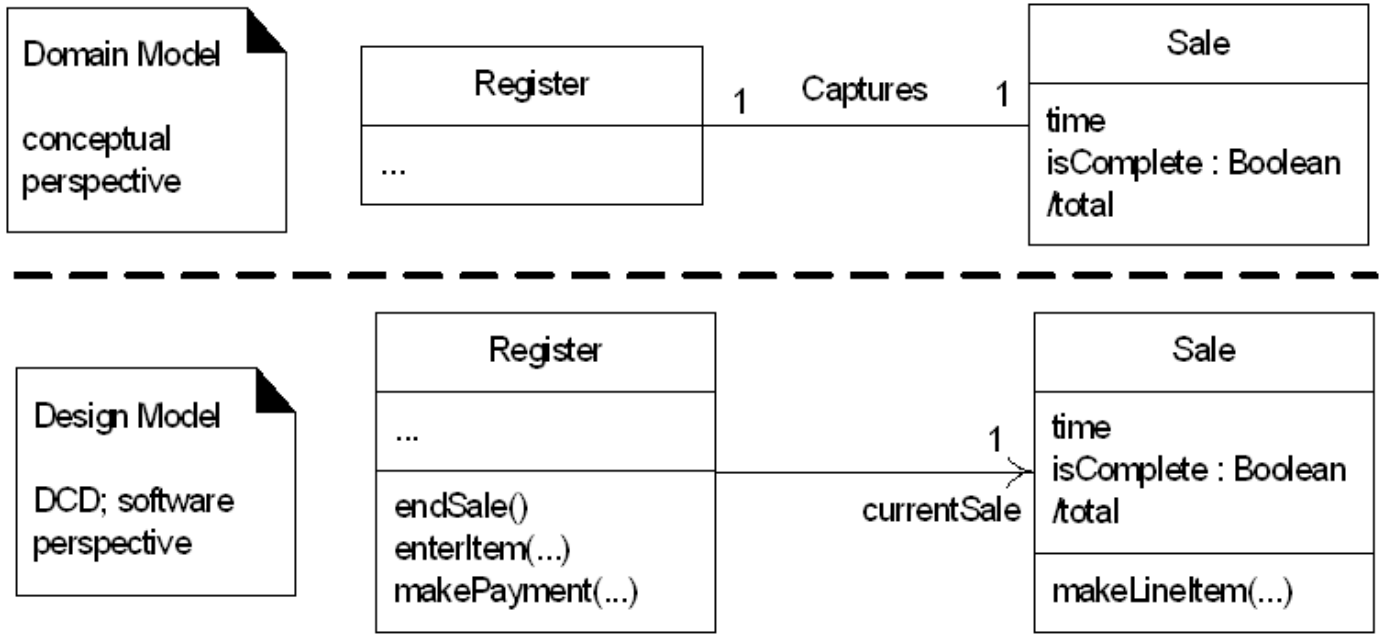
NextGen POS: Polymorphic Messages (Cont.)



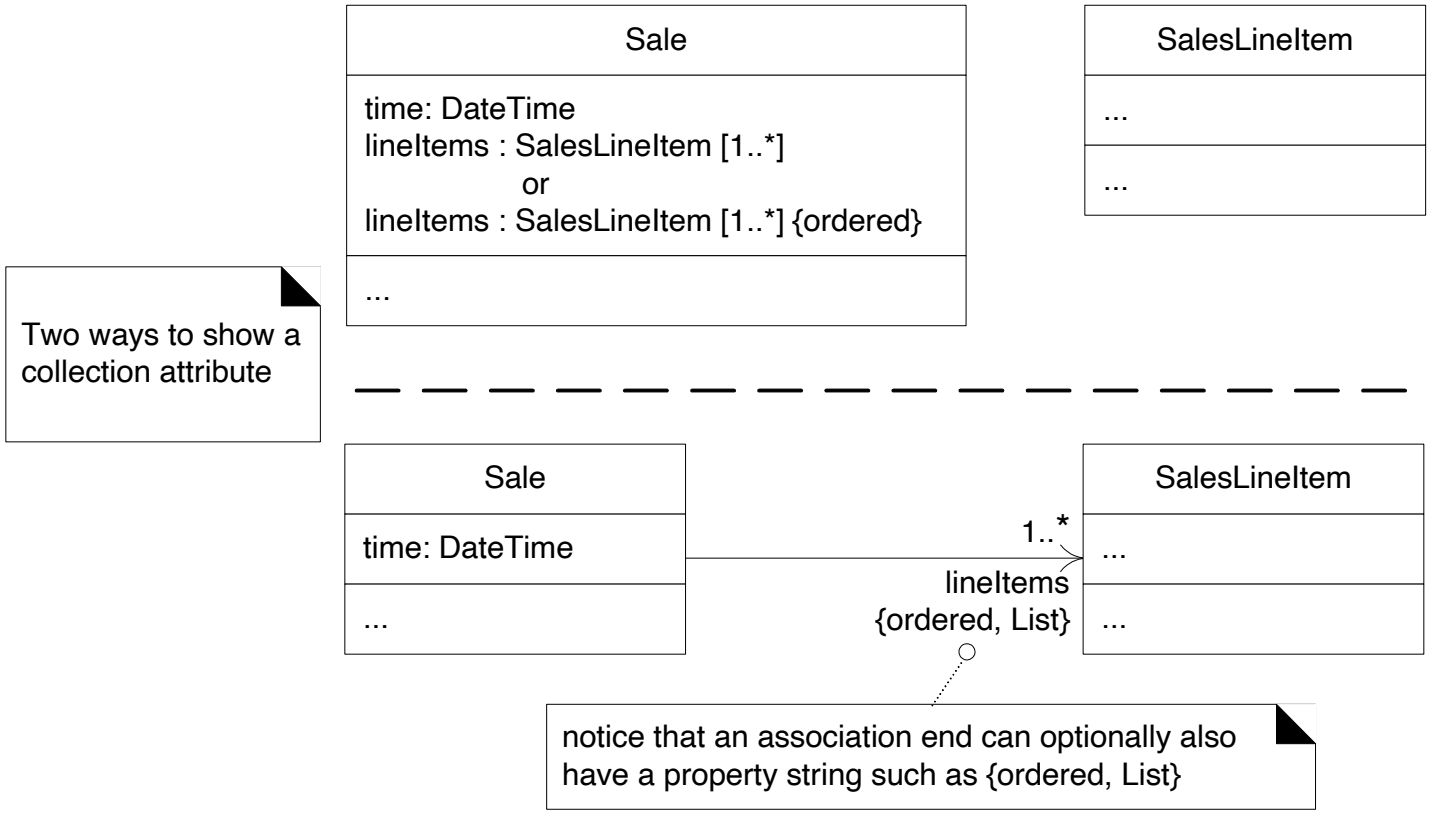
NextGen POS: Interaction and Class Diagrams



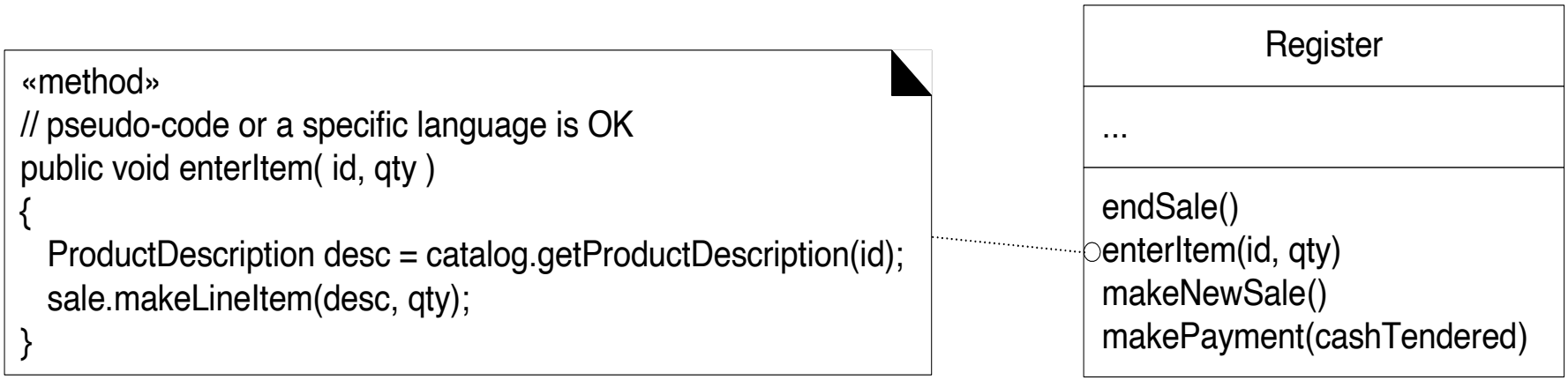
NextGen POS: Design Class Diagram



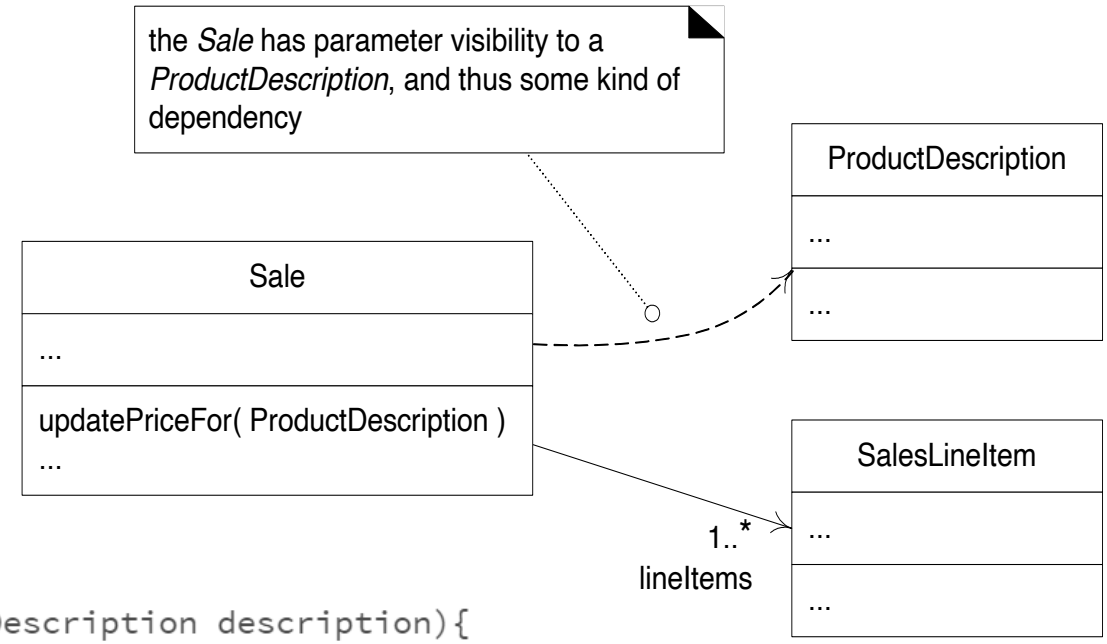
NextGen POS: Collection of Attributes



NextGen POS: Methods



NextGen POS: Dependency

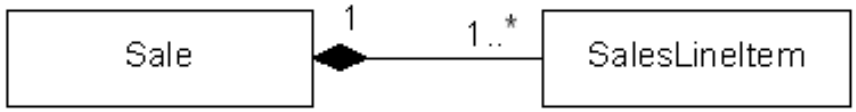



```
1     public class Sale{
2         public void updatePriceFor (ProductDescription description){
3             Money basePrice = description.getPrice();
4             //…
5         }
6 }
7
```

Next Gen POS: Composite Aggregation

SalesLineItem instance can only be part of one composite (*Sale*) at a time.

The composite has sole responsibility for management of its parts, especially creation and deletion



Composition (or Composite Aggregation is a strong kind of whole-part aggregation.
Use composition over aggregation as the latter was deemed by UML creators as "*Placebo*"