Domain modeling

CS 414, Fall 2018

Laura Moreno
lmorenoc@colostate.edu

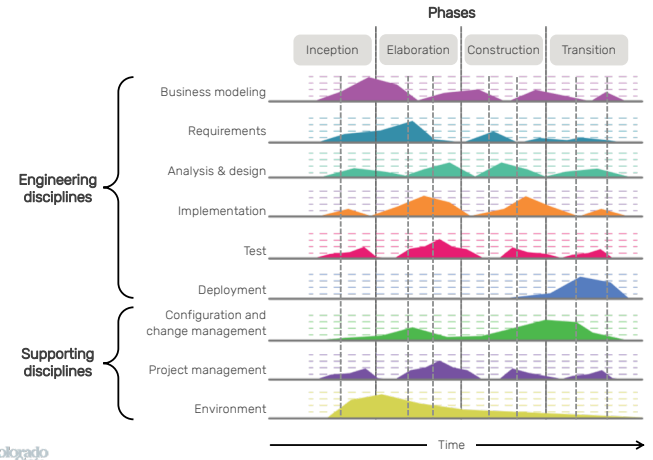
Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General domain modeling guidelines
- Exercises

Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General domain modeling guidelines
- Exercises

The UP phases



Elaboration phase

Core, risky software architecture is programmed and tested

Majority of requirements are discovered and stabilized

Major risks are mitigated or retired

UP elaboration artifacts

Domain model (topic of this module)

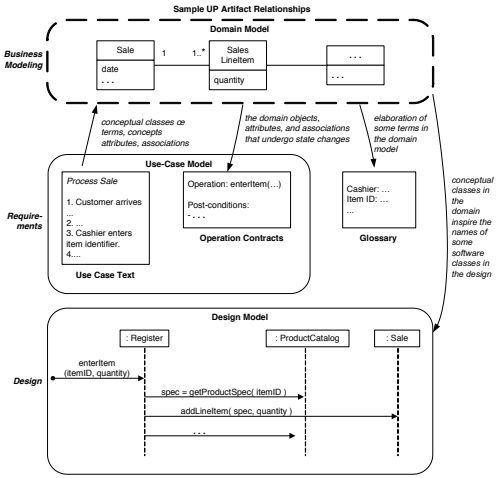
Design model

Software architecture document

Data model

UI prototypes

UP elaboration artifacts in context



Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General domain modeling guidelines
- Exercises

What is a class?

A class is a description of a set of objects that share the same properties

- At the requirements level a class describes a concept in the problem domain; properties modeled by attributes and relationships
- At the design level a class describes a concept in the solution domain; properties modeled by attributes, relationships, and operations
- At the programming level a class defines objects that will perform computations; properties modeled by attributes, relationships, and operations

An object is a concept, abstraction, or thing with identity that has meaning for an application.

- An object is an instance of a class
- Each object "knows" its class

What is a class model?

Syntactically, a class model is a structure of classes

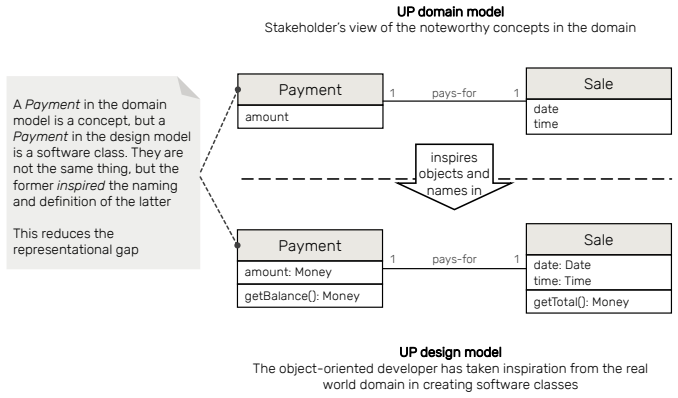
Semantically,

- a requirements class model (aka domain model) describes problem concepts and their relationships
- a design class model describes solution concepts and their relationships
- an implementation class model describes program-level classes (e.g., Java classes) and their links

Key class modeling question: What are the objects of interest in the problem space?

- What are their basic properties (in terms of attributes)?
- What are their relationships?

Domain model vs. design model



General style guidelines for classes

Center class name

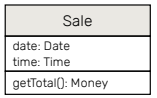
Capitalize the first letter of class names

Left justify attributes and operations in plain face

begin attribute and operation names with a lowercase letter

Class name should be in italics if the class is abstract

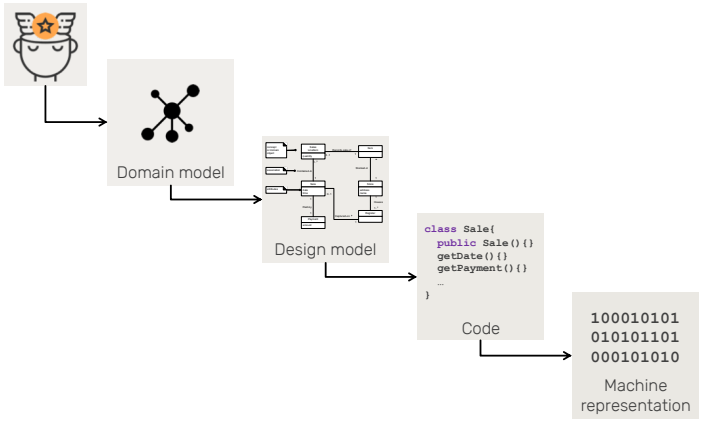
- An abstract class is one whose instances must be instances of a specialized class
- At the implementation level, this translates to a class that cannot be instantiated



Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General modeling guidelines
- Exercises

Why OO analysis and design?



What is a domain model?

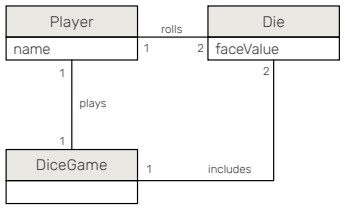
A domain model is a visual representation of real-world **conceptual classes** in a **domain** of interest

Aka requirements class model, domain object model, analysis object model

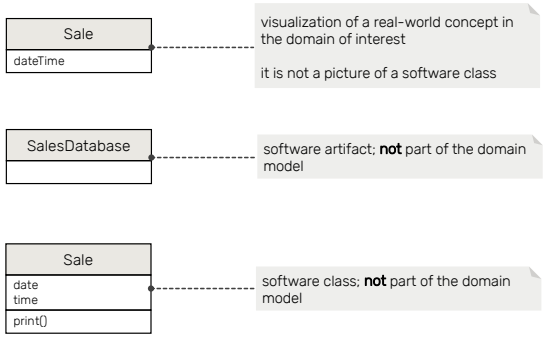
In UML, a domain model is illustrated with a set of **class diagrams** in which **no operations** are defined

A domain model shows

- Domain objects or conceptual classes
- Associations between classes
- Attributes of conceptual classes



What is it shown in a domain model?



Domain model ≠ Data model ≠ Design model

Creating a domain model

Start from the requirements specification

Work iteratively

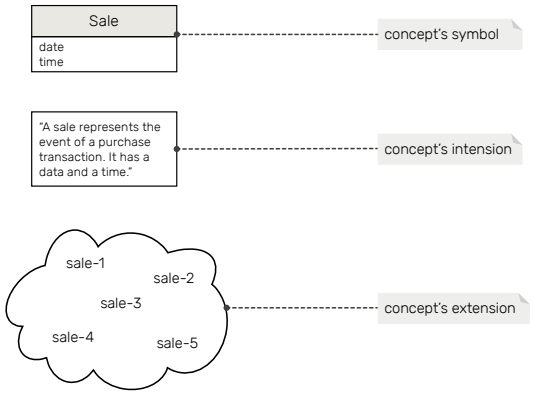
- Find conceptual classes
  - Reuse or modify existing models
  - Use a category list
  - Identify noun phrases
- Draw concepts as classes in a UML diagram
- Add associations and attributes

Avoid "analysis paralysis"—don't waste time on a too thorough model

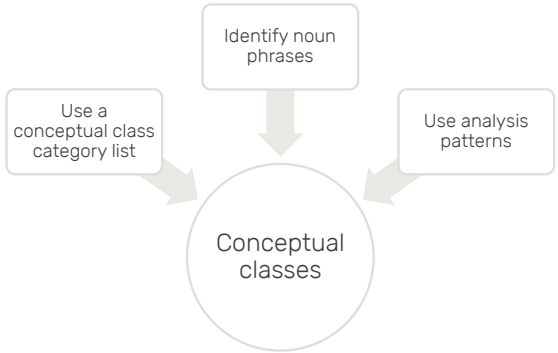
Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General domain modeling guidelines
- Exercises

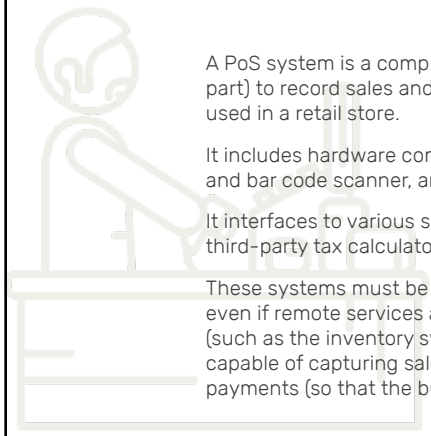
What are conceptual classes?



Identifying conceptual classes



The PoS system case



A PoS system is a computerized application used (in part) to record sales and handle payments; it is typically used in a retail store.

It includes hardware components such as a computer and bar code scanner, and software to run the system.

It interfaces to various service applications, such as a third-party tax calculator and inventory control.

These systems must be relatively fault-tolerant; that is, even if remote services are temporarily unavailable (such as the inventory system), they must still be capable of capturing sales and handling at least cash payments (so that the business is not crippled).

Conceptual class category list

Conceptual class category	Domain concepts
Physical or tangible objects	
Specifications, designs, or descriptions of things	
Places	
Business transactions	
Roles of people	
Containers of things	
Things in containers	
Collaborating systems	
Abstract noun concepts	
Organizations	
Events	
Records of finance	

Conceptual class category list

Conceptual class category	Domain concepts
Physical or tangible objects	Register, Item
Specifications, designs, or descriptions of things	ProductDescription
Places	Store
Business transactions	Sale, Payment
Roles of people	Cashier, Customer, Store
Containers of things	Store, Bin
Things in containers	Item
Collaborating systems	CreditPaymentAuthorizationSystem
Abstract noun concepts	Hunger
Organizations	SalesDepartment
Events	Sale, Payment, Meeting
Records of finance	Receipt, Ledger, EmploymentContract

Noun phrase identification

Main Success Scenario (or Basic Flow):

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. Price is calculated from a set of price rules.

Cashier repeats steps 2-3 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 the completed sale and sends sale and payment information to the external Accounting (for accounting and commissions) and Inventory systems (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows): ...

7.a. Paying by cash:

1. Cashier enters the cash amount tendered.
2. System presents the balance due, and releases the cash drawer.
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

Noun phrase identification

Main Success Scenario (or Basic Flow):

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. Price is calculated from a set of price rules.

Cashier repeats steps 2-3 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 the completed sale and sends sale and payment information to the external Accounting (for accounting and commissions) and Inventory systems (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows): …

7.a. Paying by cash:

1. Cashier enters the cash amount tendered.
2. System presents the balance due, and releases the cash drawer.
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

Structure of a conceptual class

A conceptual class has the following structure:

- Name compartment (mandatory)
- Attributes compartment (optional)

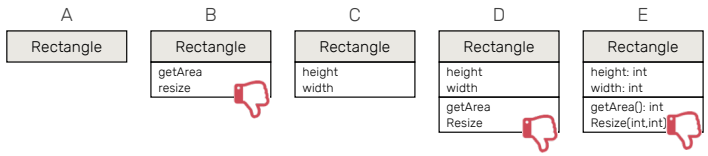
Every class must have a unique name

An object of a conceptual class must have values associated with each attribute of the class

A conceptual class does not contain operations

- Assigning operations to classes is often a design decision
- Required behavior is described by use cases

Conceptual classes ::: Don'ts



No operations!

Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General modeling guidelines
- Exercises

Attributes

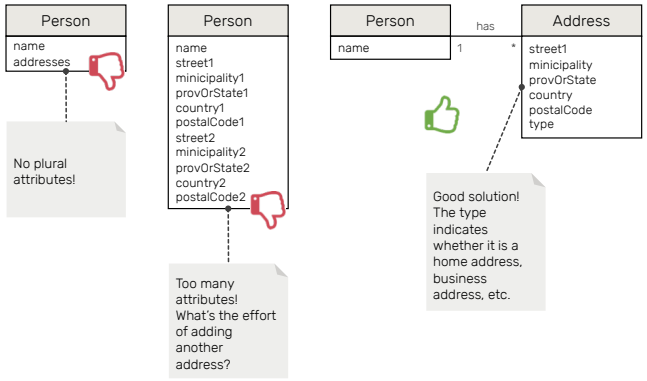
An attribute is a named property

- Each class instance associates value(s) with each attribute of a class

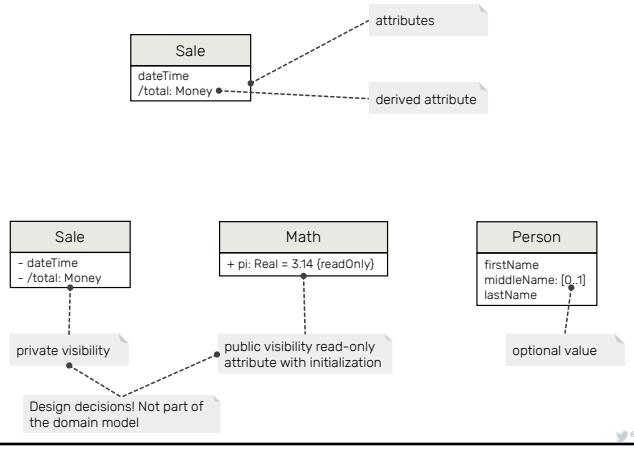
What should be modeled as an attribute in a conceptual class?

- Properties of problem concepts that we want to treat as primitive (i.e., not decomposable)

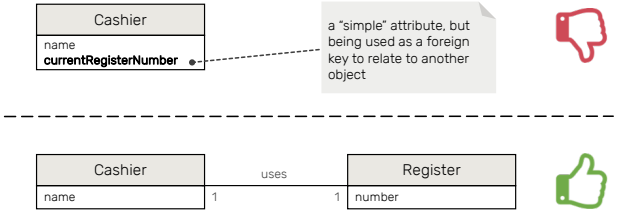
Attributes ::: Dos & Don'ts



Attribute notation

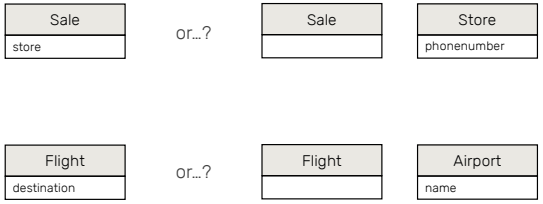


Don't use attributes as foreign keys



Leave the design/implementation for later; just show an association

Common mistake ::: Is it a class?



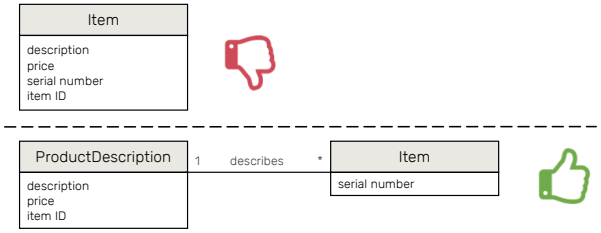
If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute

Is a *Description* class needed?

If a description is needed independent of the current existence of the described item, a new class is necessary

Reduce redundant or duplicate information

Deleting instances of the described item results in the loss of information that needs to be maintained

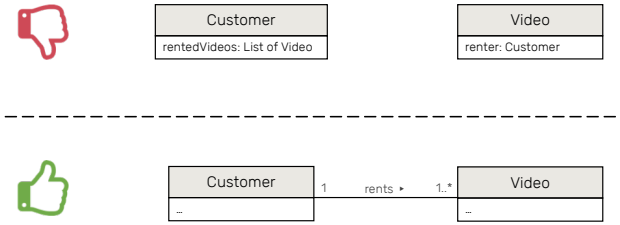


Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General domain modeling guidelines
- Exercises

How to relate concept in class models?

Do not use attributes to relate concepts



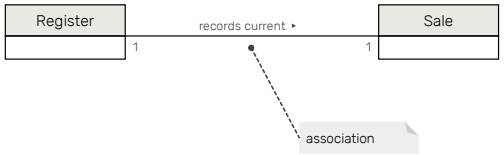
Use associations between the concepts

Associations

Relationships between classes that indicates meaningful and interesting connections

Needed to satisfy the information requirements

Aid in understanding the domain

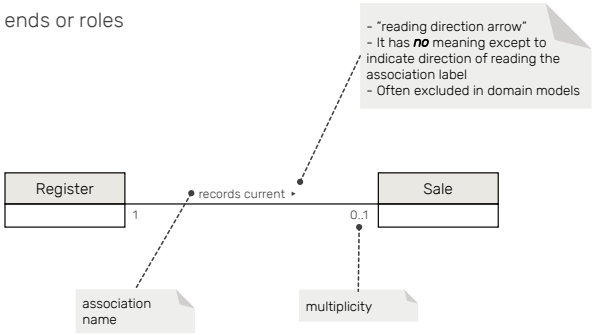


Showing associations in UML

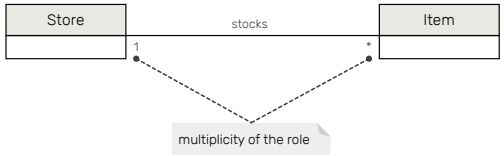
Naming

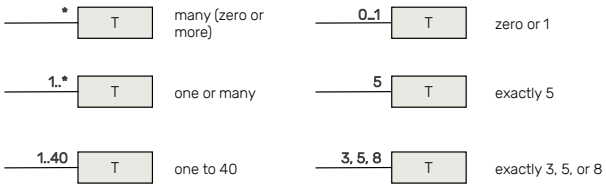
Association ends or roles

Multiplicity



Multiplicity





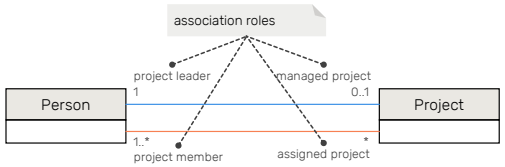
Association roles

When a class is part of an association, its objects (which are linked via the association) play a role in the relationship

You can name the role that an object plays in an association by placing the name at the class's association end

Formally, a class role is the set of objects that are linked via the association

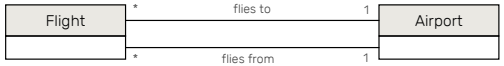
Association roles ::: Example



project leader = {Bob, Keisha, Jane}
managed project = {proj1, proj2, proj3}

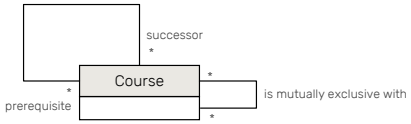
project member = {Bob, Bill, Keisha}
managed project = {proj1, proj2, proj3}

Multiple associations between two classes

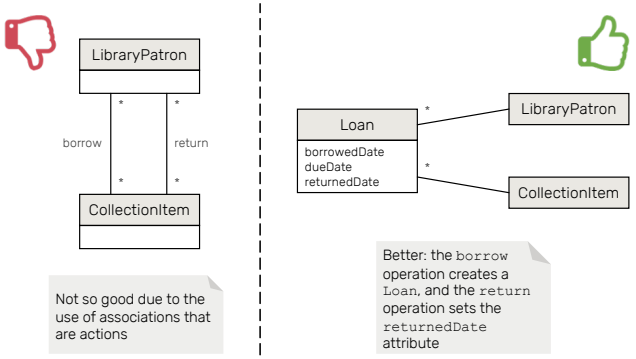


Reflexive associations

It is possible for an association to connect a class to itself

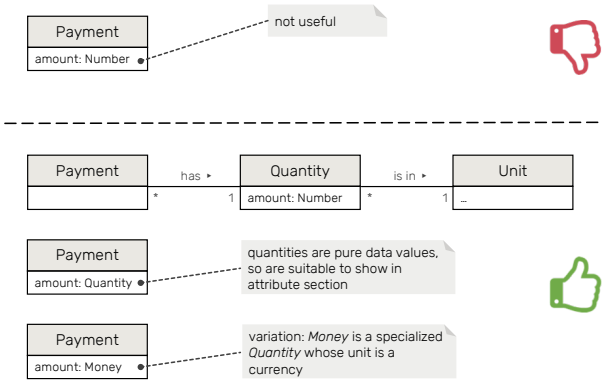


Common mistake ::: Is it an association?



A common mistake is to represent actions as associations

Associations ::: Quantities and units



Associations ::: Guidelines

Focus on associations for which knowledge of the relationship must be preserved for some duration

- An association should exist if a class
  - possesses
  - controls
  - is connected to
  - is related to
  - is a part of
  - has as parts
  - is a member of, or
  - has as member

  another class

More important to identify concepts than associations

- Too many associations can lead to confusing models

Avoid showing redundant/derivable associations

Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General domain modeling guidelines
- Exercises

Domain models ::: Guidelines

Not all identified domain concepts are part of the domain model

Do not add navigation arrows on associations in conceptual class diagrams

- Determining which class is visible to another class is a design decision

Do not add operations in conceptual class diagrams

- There can be more than one way to assign responsibilities to a class; this is a design decision
- To avoid making design decisions too early, novice modelers should not include operations in requirements class models

All association ends must have multiplicities

- There is a default, but most people do not remember it; better to be explicit to avoid confusing the reader

All associations must be named (either an association name or a role name at each end)

Roadmap

- The elaboration phase
- Class models
- Domain models
  - Conceptual classes
  - Attributes
  - Associations
- General domain modeling guidelines
- Exercises

Course management system (CMS)

During an academic term a lecturer reads one or more lectures

Sometimes the lecturer is on leave to focus on doing research, in this case (s)he does not give a lecture

A student usually attends one or more lectures, unless (s)he has something better to do

During the academic term and for a given lecture, there will be several exercises which are meant to be solved by small study groups

Each student is assigned to one particular study group for the whole academic term

A study group consists of two or three students

After submission, the solution of an exercise is graded by a tutor

Students get a bonus by the end of the academic term, which reflects the relative number of exercise points gained during the academic term

Automated teller machine (ATM)

Design the software to support a computerized banking network that includes both human cashiers and automatic teller machines (ATMs) to be shared by a consortium of banks.

Each bank provides its own computer to maintain its own accounts and process transactions against them.

Cashier stations are owned by individual banks and communicate directly with their own bank's computers. Human cashiers enter account and transaction data. Automatic teller machines communicate with a central computer which clears transactions with the appropriate banks.

An automatic teller machine accepts a cash card, interacts with the user, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts.

The system requires appropriate record-keeping and security provisions. The system must handle concurrent accesses to the same account correctly.

The banks will provide their own software for their own computers; you are to design the software for the ATMs and the network.

The cost of the shared system will be apportioned to the banks according to the number of customers with cash cards.