MANCHESTER 1824

The University of Manchester

COMP23420 Lecture 4

Structural Modelling: Domain Classes

Kung-Kiu Lau

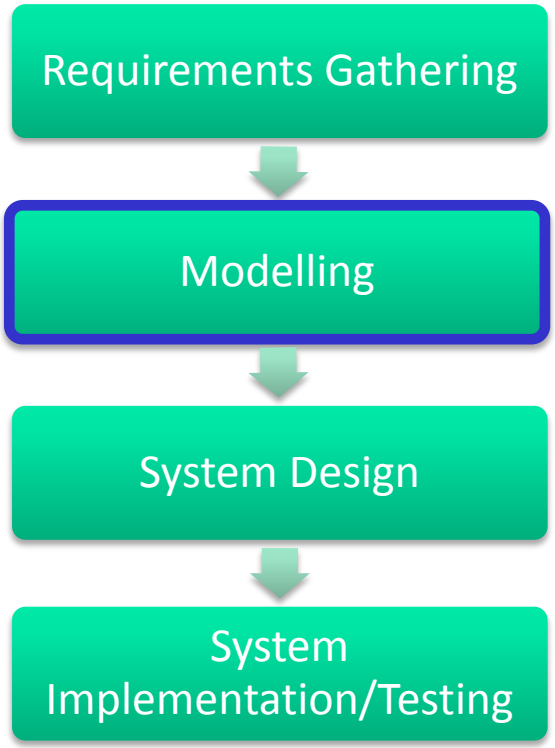
kung-kiu@cs.man.ac.uk

Office: Kilburn 2.68

Overview

- Where we are in the development process
- What are domain models?
- What are domain classes?
- How to identify domain classes and their relationships
- How to define domain classes and their relationships

Where we are in the Development Process



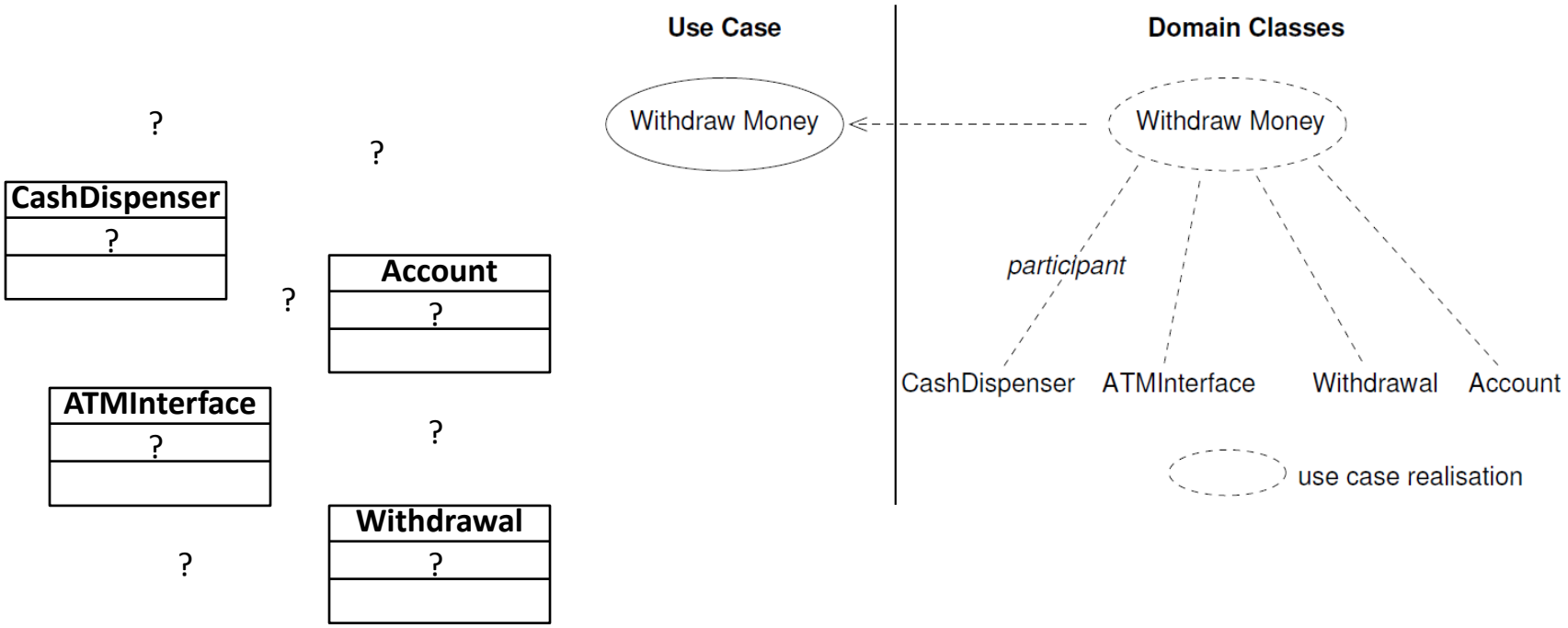
Functional Model
- Activity Diagram
- Use Case Diagram

Structural Model
- Domain Model (Classes)
- System Classes

Behavioural Model
- Sequence Diagrams
- Communication Diagrams
- State Machine Diagrams

We have the requirements document, the use case descriptions, the use case diagrams, etc.; and now we begin to construct the structural model, starting with the domain model.

Domain Classes Realise Use Cases



But what are domain classes? And where do they come from?

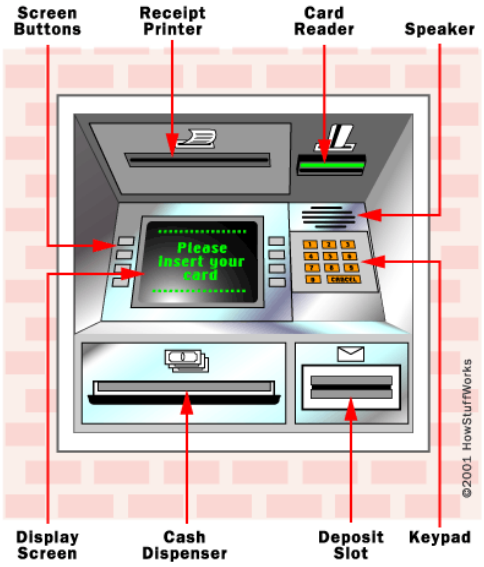
They are conceptual classes defined in the domain model

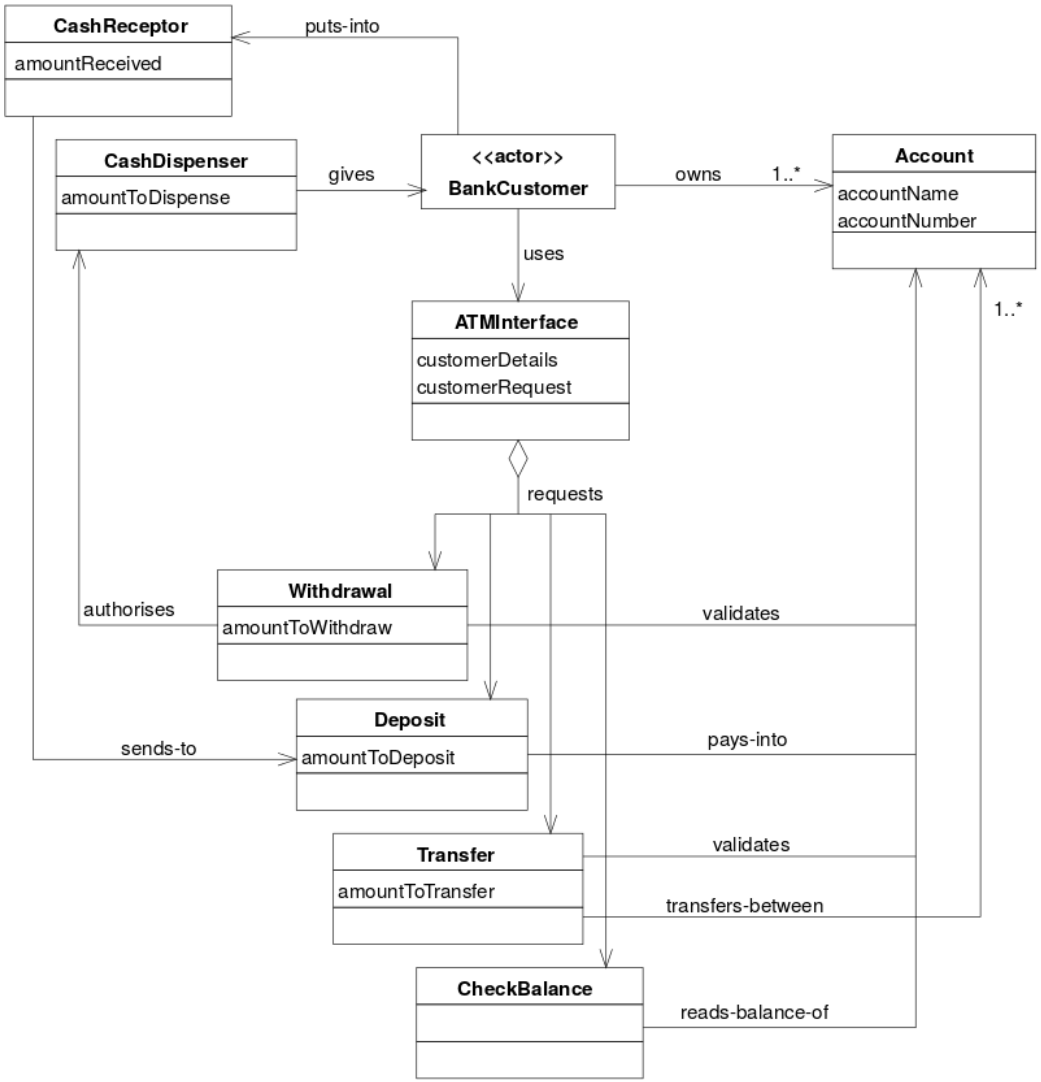
A domain model is a model of the problem (domain)

What is a Domain Model?

A domain model captures

- the most important types of entities in the context of the system, i.e. the (problem) domain
- and their relationships



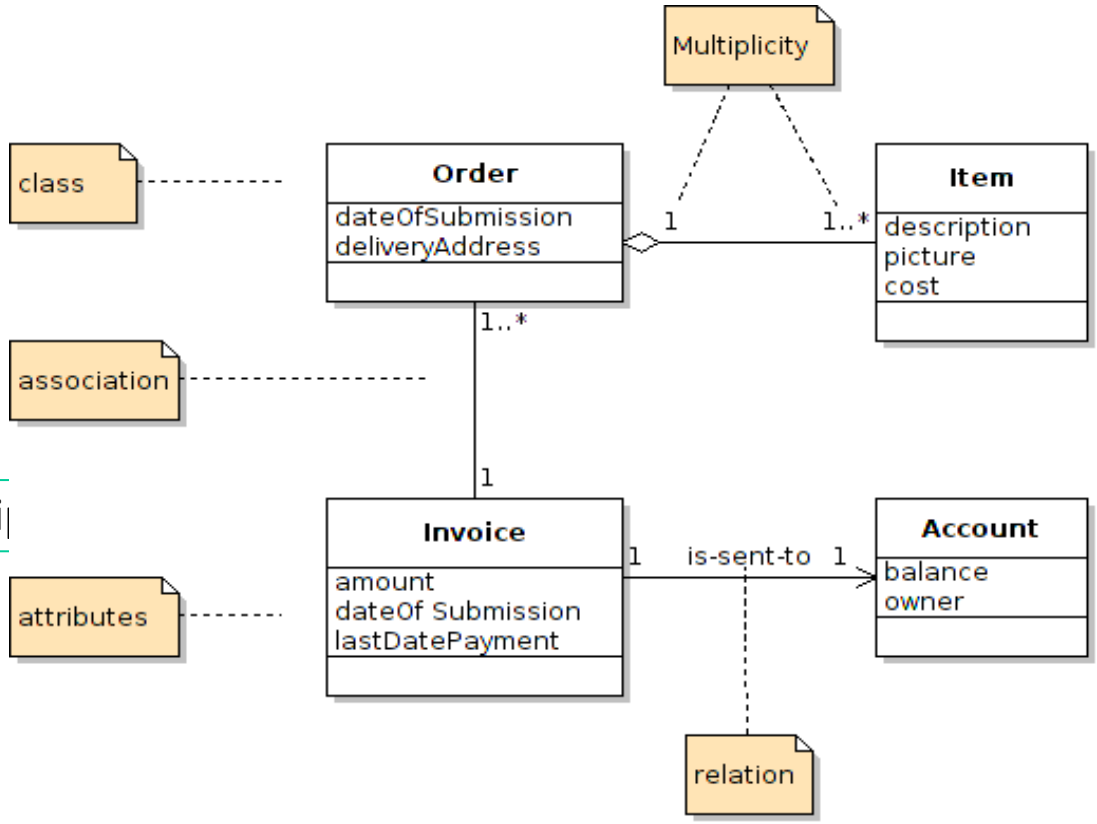


Domain model for ATM example

Domain Models in UML

In UML a domain model is defined as a set of related classes

- with attributes
- but no operations
- with associations (as relationshi



Domain model of a sales invoicing system

Domain entities are also confusingly called domain objects

Domain Modelling

To define a domain model we define:

- conceptual classes (domain classes) that represent the (most important) entities in the domain
- and their relationships

Defining a domain model is usually an iterative process

The less important (but possibly numerous) classes are kept as entries in a glossary of terms

Constructing a Domain Model

Domain classes participate in use case realisations

A domain model must contain domain classes involved in all the use case realisations

Therefore, to construct a domain model, we need to go through all use case realisations and identify domain classes that participate in them

Steps of Domain Modelling

For each use case realisation:

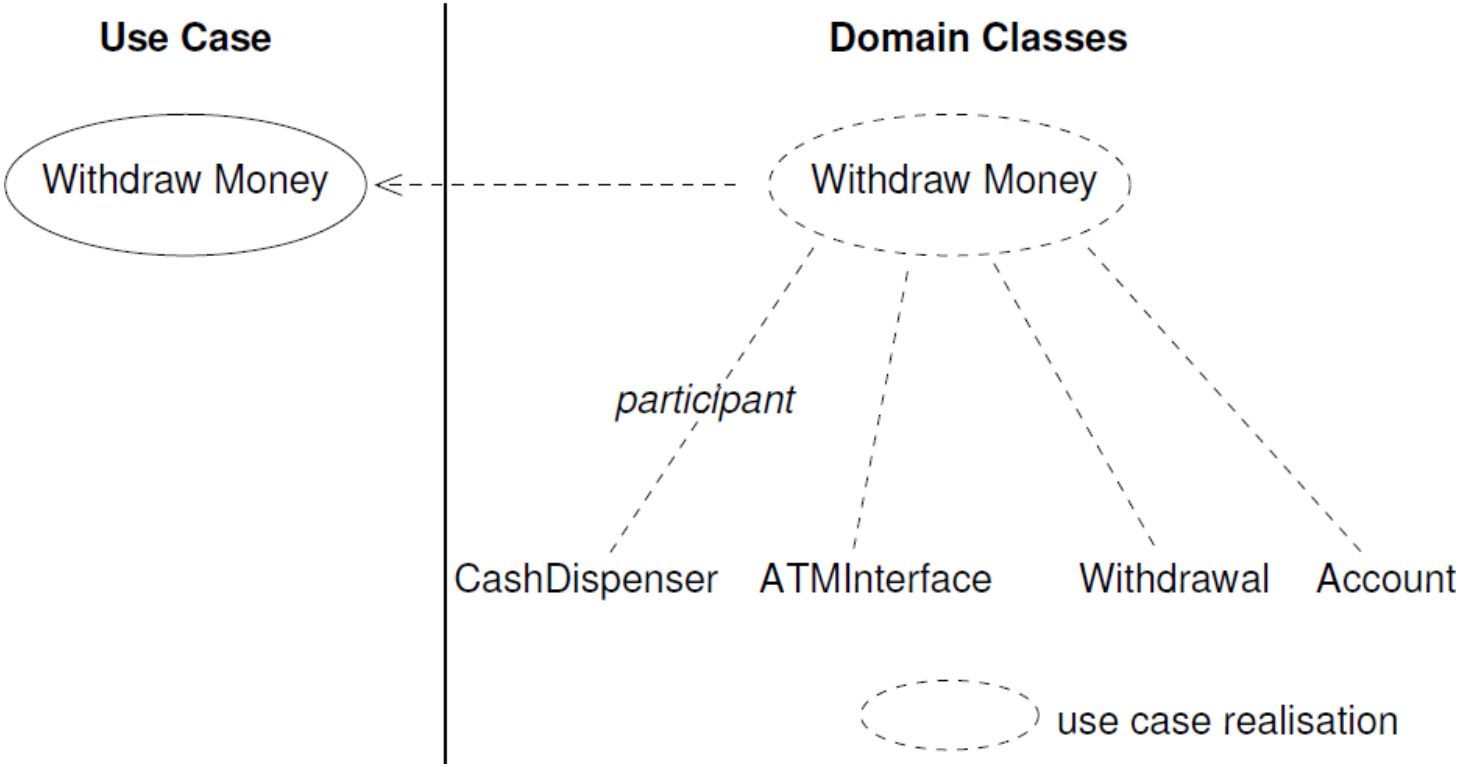
- Go through available data sources (requirements, use case descriptions, glossary) and identify noun phrases as possible domain classes (and draw them as classes in a UML class diagram)
- Identify basic relationships (associations) between domain classes (and add them to the class diagram)
- Add key attributes (to the class diagram)
- Look more closely at relationships (names and multiplicities) – important to get these right

Aggregate the domain classes for all the use case realisations

Iterate as necessary

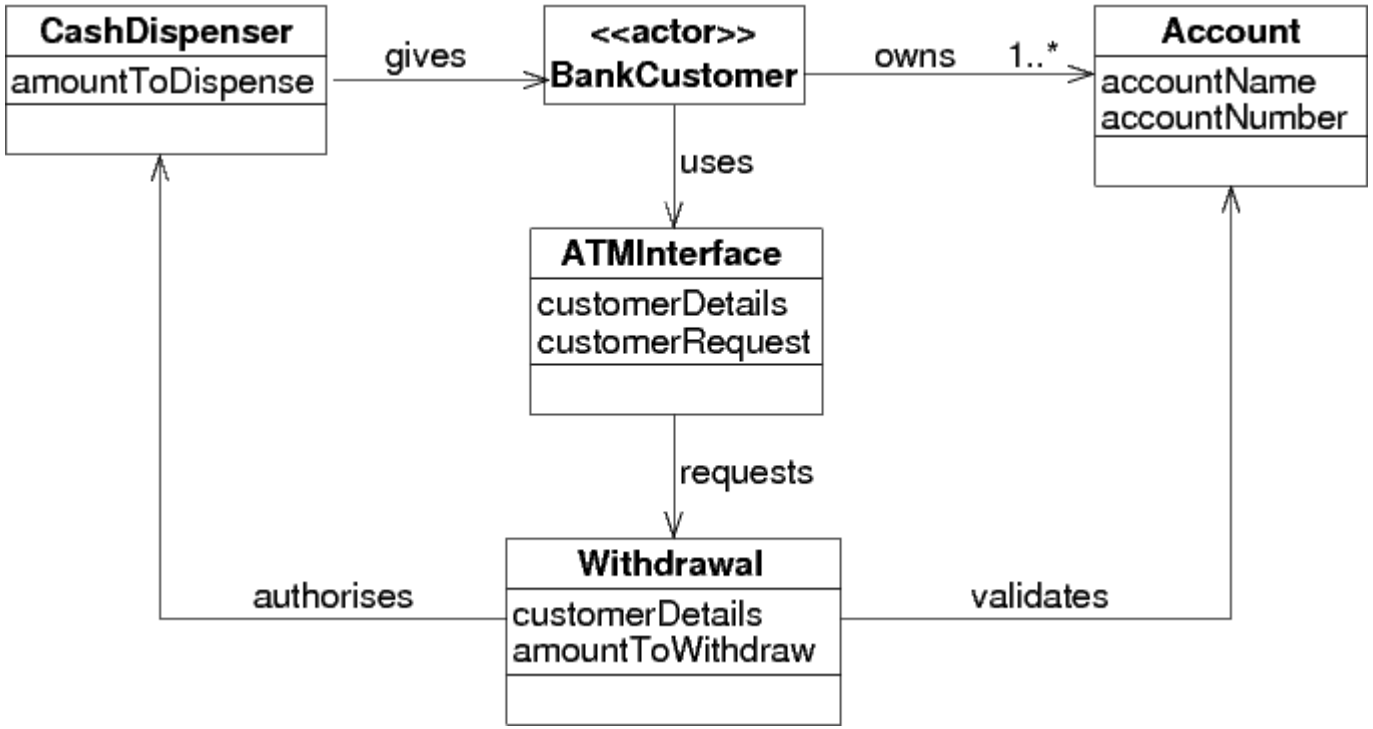
Update the original data sources – especially the glossary

Domain Modelling: The ATM Example



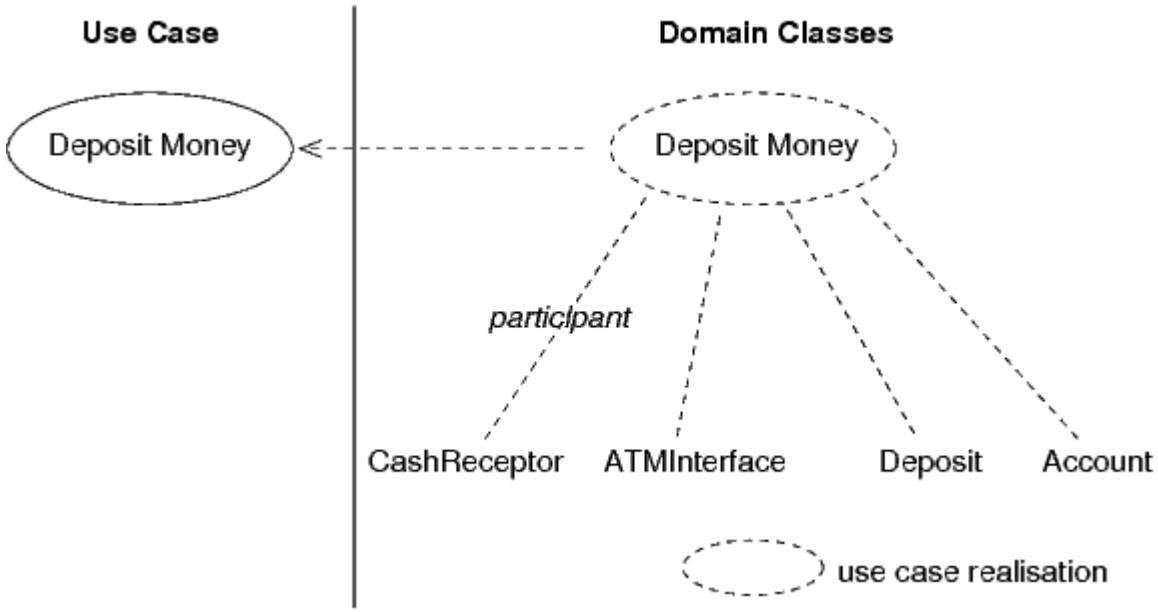
Domain classes that realise the Withdraw Money use case

Domain Modelling: The ATM Example



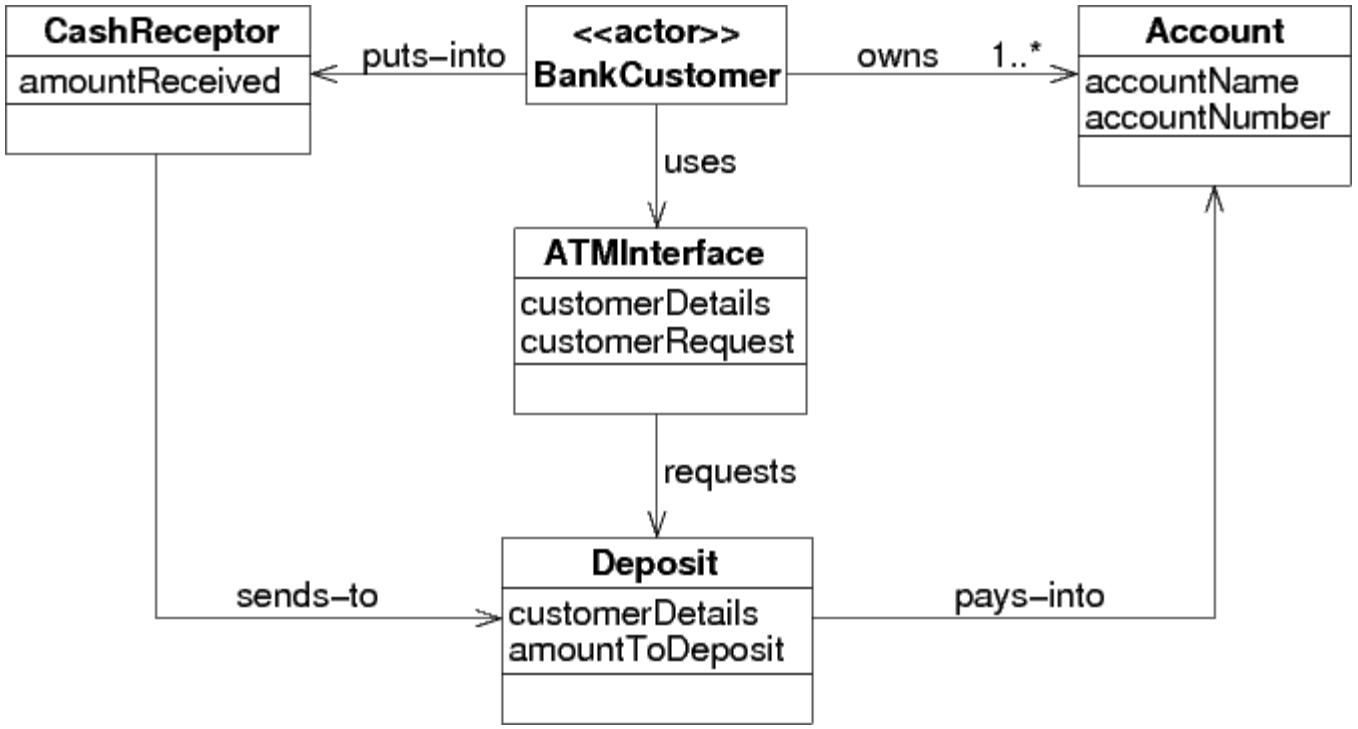
Domain class diagram for the Withdraw Money use case

Domain Modelling: The ATM Example



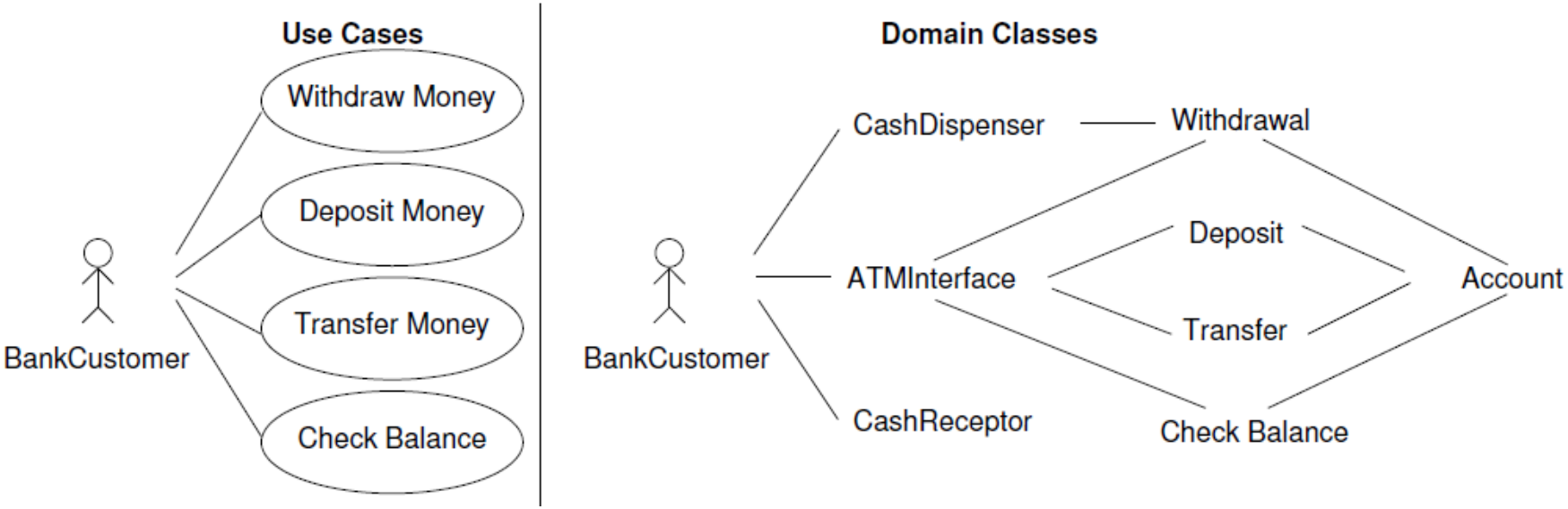
Domain classes that realise the Deposit Money use case

Domain Modelling: The ATM Example



Domain class diagram for the Deposit Money use case

Domain Modelling: The ATM Example

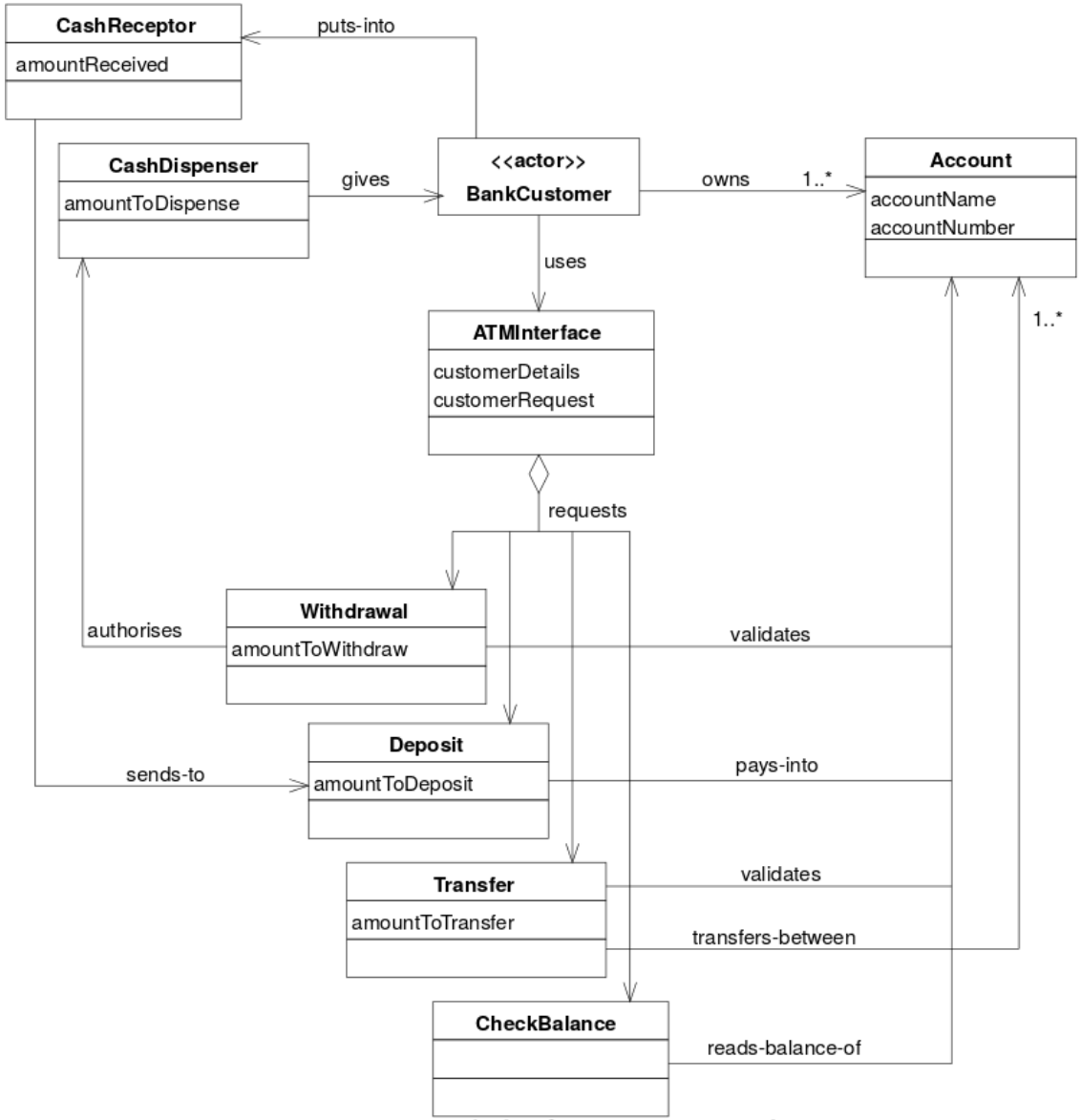


All the use cases and their realisations by domain classes

Domain Modelling: The ATM Example

Domain class diagram for all the use cases

This is the domain model



Domain model of ATM example

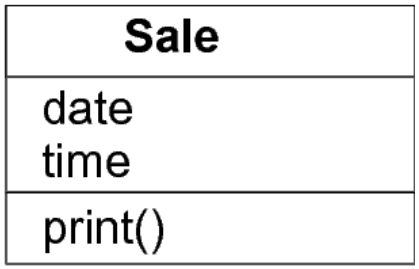
Some Guidelines: Avoid Software Classes

Domain classes represent entities in a real-situation (problem) domain.

They are not software artefacts or classes

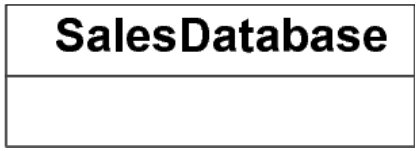
Visualisation of a real-world concept in the domain of interest. Not a picture of a software class

avoid



Software class! Not part of the domain model

avoid



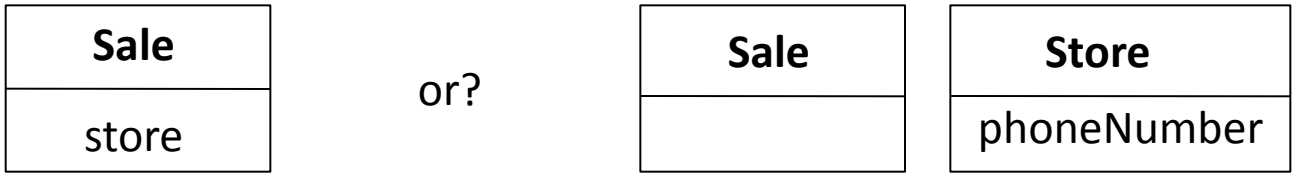
Software artefact! Not part of the domain model

Acknowledgement: These diagrams (and similar ones in other slides) are taken from Larman, *Applying UML and Patterns*, third edition, Prentice Hall, 2005.

Some Guidelines: Attributes vs Classes

If we do not think of X as a number or text in the real world, then X is probably a (conceptual/domain) class, not an attribute.

For example, should store be an attribute of Sale, or a separate (domain) class Store?



It should be a separate (domain) class Store.

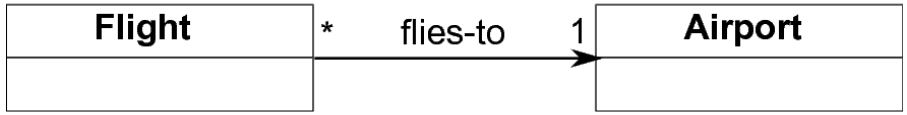
Some Guidelines: Suitable Attribute Types

Attribute types should be (primitive) data types, not complex domain concepts, or foreign keys

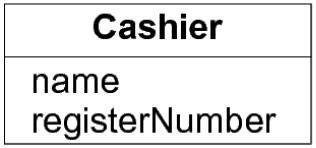
Worse

`destination' is a complex concept

Better

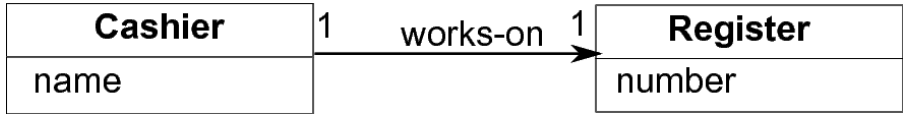


Worse



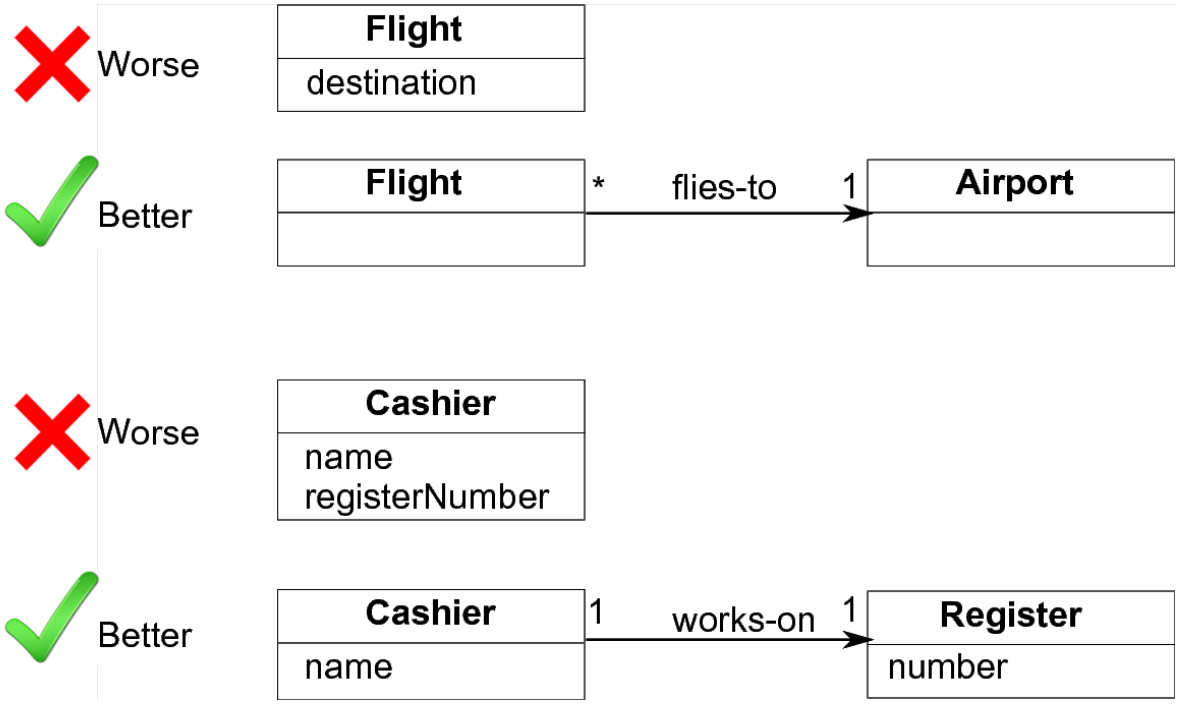
`registerNumber' is being used as a foreign key to relate to another entity

Better



Some Guidelines: Suitable Attribute Types

Relate domain classes with an association, not an attribute.



Summary

Domain modelling is about representing the context in which the software must operate, to understand it better.

Domain classes have only attributes and no operations.

Primarily UML class diagrams, sometimes interaction diagrams (see later) are used too.

Remember you are not modelling the software to be built.

The primary guideline is Keep It Simple (and iterate).

Appendix: UML & Class Diagrams

Masses of notation exist in UML, but a subset is enough for most purposes.

Can be used for many purposes (good) but people often confuse different uses (bad).

Most common error is showing too much detail: remember UML is mainly a tool for communication.

So it is important to be clear about our goals when using UML notation, especially class diagrams.

Class diagrams show classes and their relationships.

For details of syntax and notation, see slides in the Supporting Material section of the Moodle site.

In the Supporting Material section of the Moodle site, there is also an interactive learning tool for UML (the result of a final year project).