
OO System Models

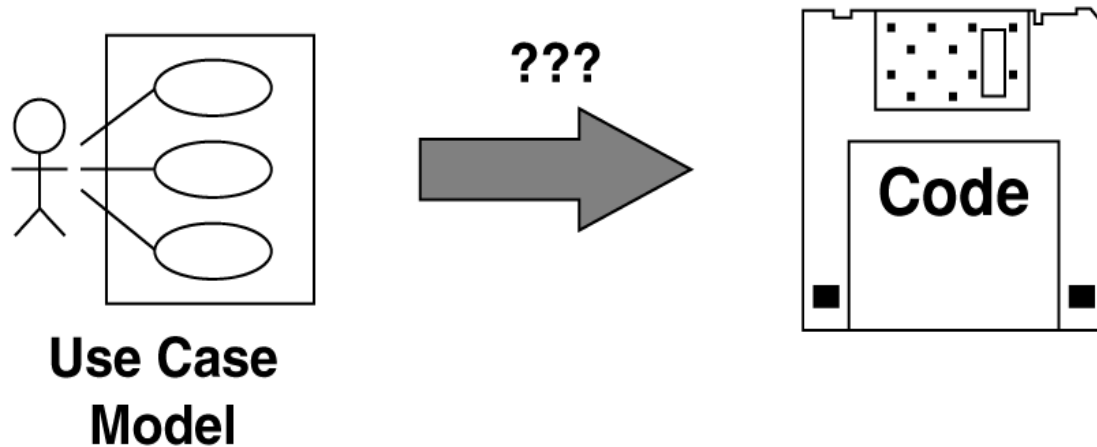
UML Use Case Driven Object Modeling

Objective

- Introduce the requirements view of a system
- Explain the sections of a use case text
- Provide the student with a template for writing the use case description
- Introduce the use case and context diagrams
- Describe the artifacts used with a Use Case
- Explain a logic via artifacts decision tables or decision tree
- Explain use cases relations, e.g., include, extend, generalise

From use cases to code

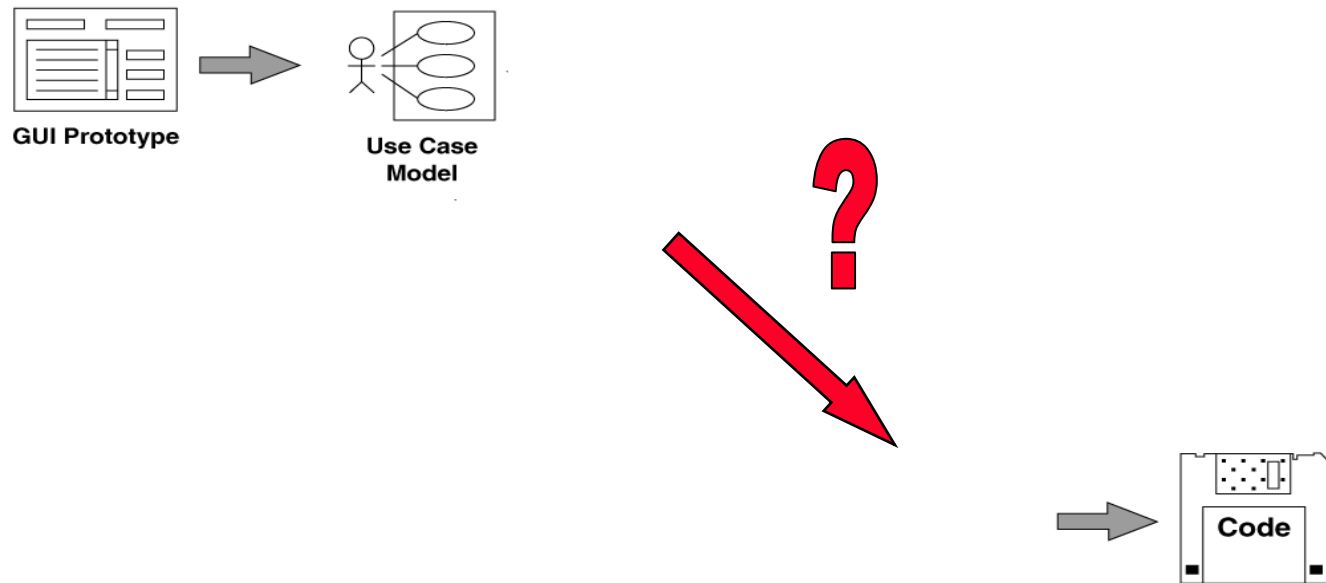
•



How do we get from use cases to code?

Work backwards from code

- Do a little prototyping,
- and start to write some use cases.



Design-level class diagram

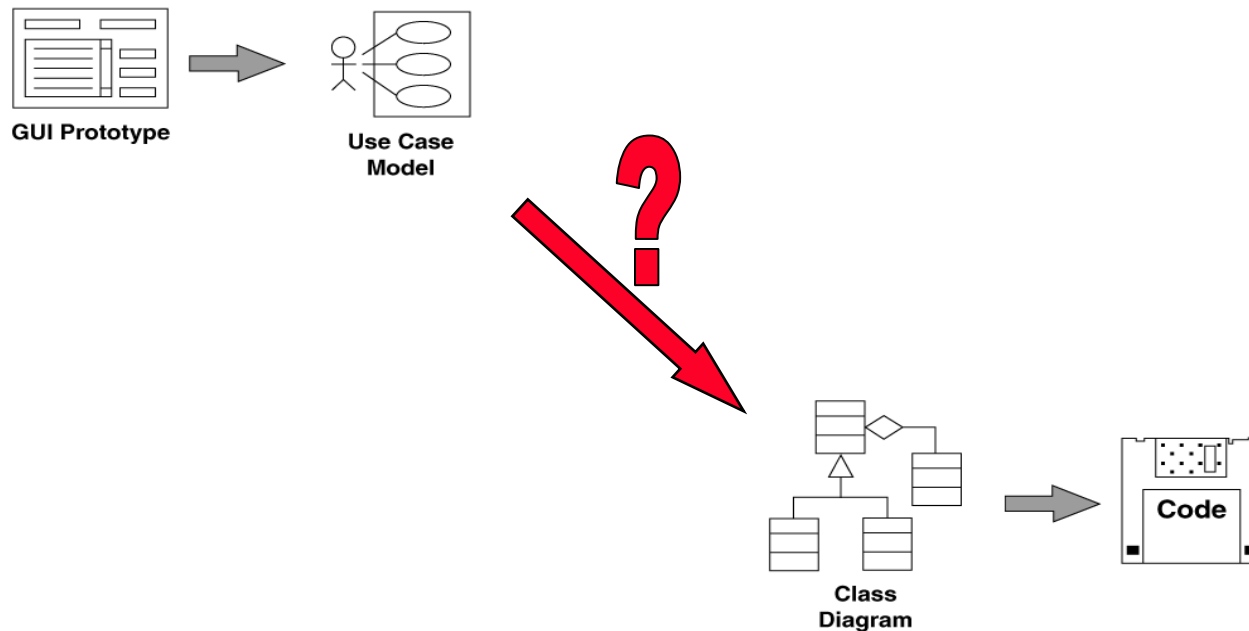
Before getting to code:

- Get **design-level** class diagram (details):
 - All attributes and operations (methods), visibility,..
 - Relationships (generalization, association, aggregation, composition)

- Note:
 - **Analysis-level** class diagram includes all classes but with **less details** on attributes and operations

How to get Design-Level Class Diagram

Design-level class diagrams serve as the **structure for the code**

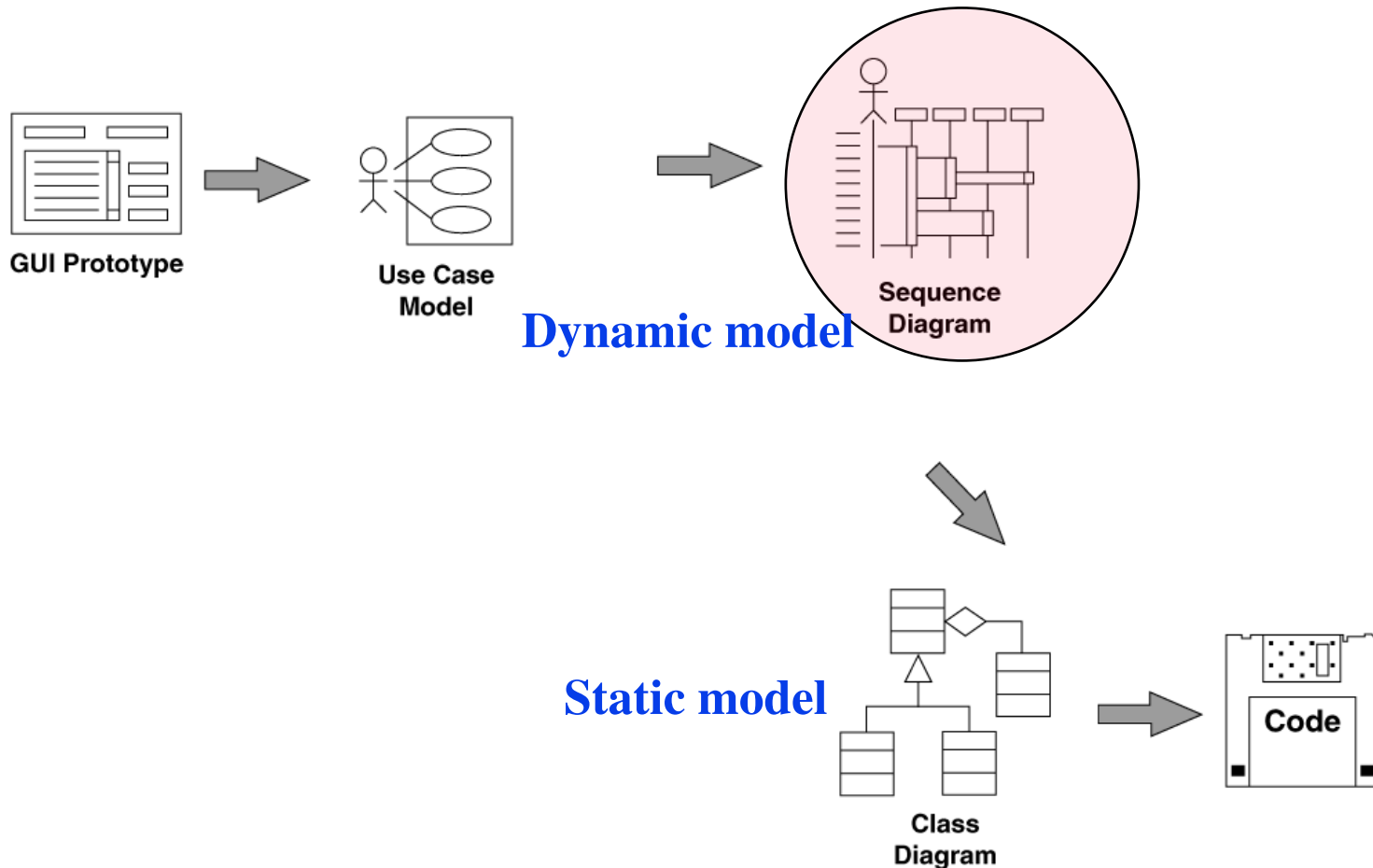


How to get Design-Level Class Diagram

- Draw sequence diagrams to allocate methods to classes
- We need to allocate behavior into classes
- Sequence diagrams help you decide **which classes are responsible for which methods**
- Draw a sequence diagram for each **use case scenario** (**sequence of steps**)

Sequence Diagrams

Allocate methods to classes as you draw sequence diagrams



Before you do sequence diagrams..

You need to have a good idea about:

- what **objects** will be performing in which use case,
- what **functions** the system will perform as a result of user actions.

UML Use Case Model

“Requirements View”

USE CASE ‘Usage Case’

Use case

- Equivalent to **requirements**
- Discover and record **functional requirements** - stories of using a sys
- Use cases are text docs

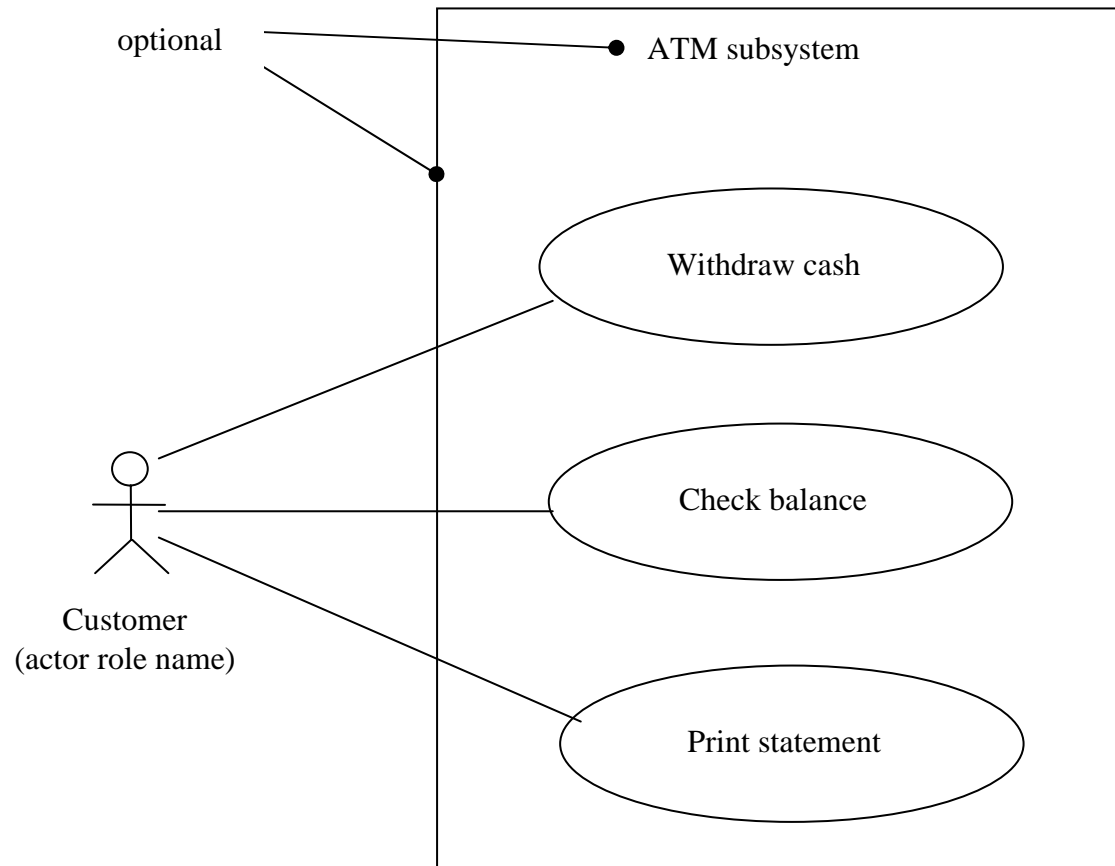
Use case diagrams: UML defines UC diag to show:

- the names of Ucs & actors,
 - their relationships –
 - may be used as contract doc with sys customer
-
- In UML UCs are the driver for the rest of the diagrams of UML.
 - UC should focus on the question “ How can using the sys provide observable value to the user, or **fulfill their goals**”

UC Diagram – Modeling the Context of a System

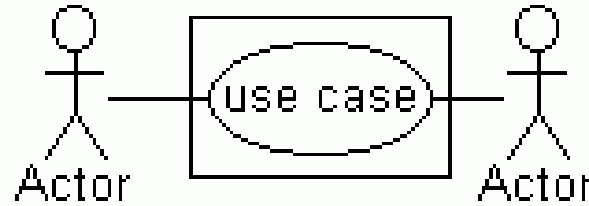
- **Components:**
 - System: rectangle
 - Actors: outside the system rectangle (external to the system)
 - Use Cases: inside the system rectangle
- **Actors**
 - People interacting with the use case
 - Other sys interacting with the UC
 - Hardware,...
- An actor **needs not** to initiate the UC; association shows actor involvement

Use case diagram for ATM subsystem

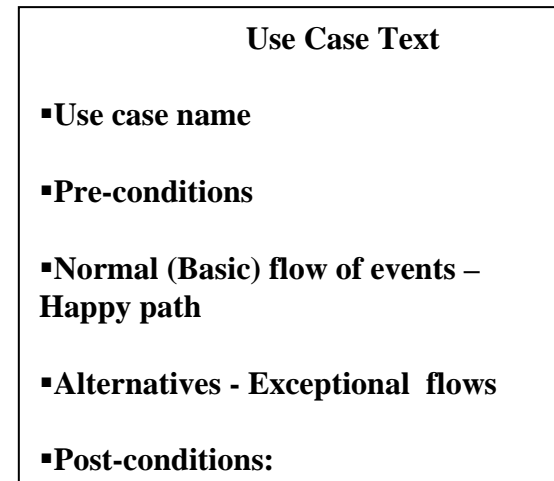


Use Case Model

- Use Case diagram →



- Use Case Text
(description) →



Use Case model

- Shows **User–System interaction** across the system boundary
- Actors interact with the **system as a whole** (not with some specific part of it)
- System is viewed as a **black box**
- Defines software **boundaries**
- Shows **functional** requirements

Actor-Goal list

- Ex: **POS** system

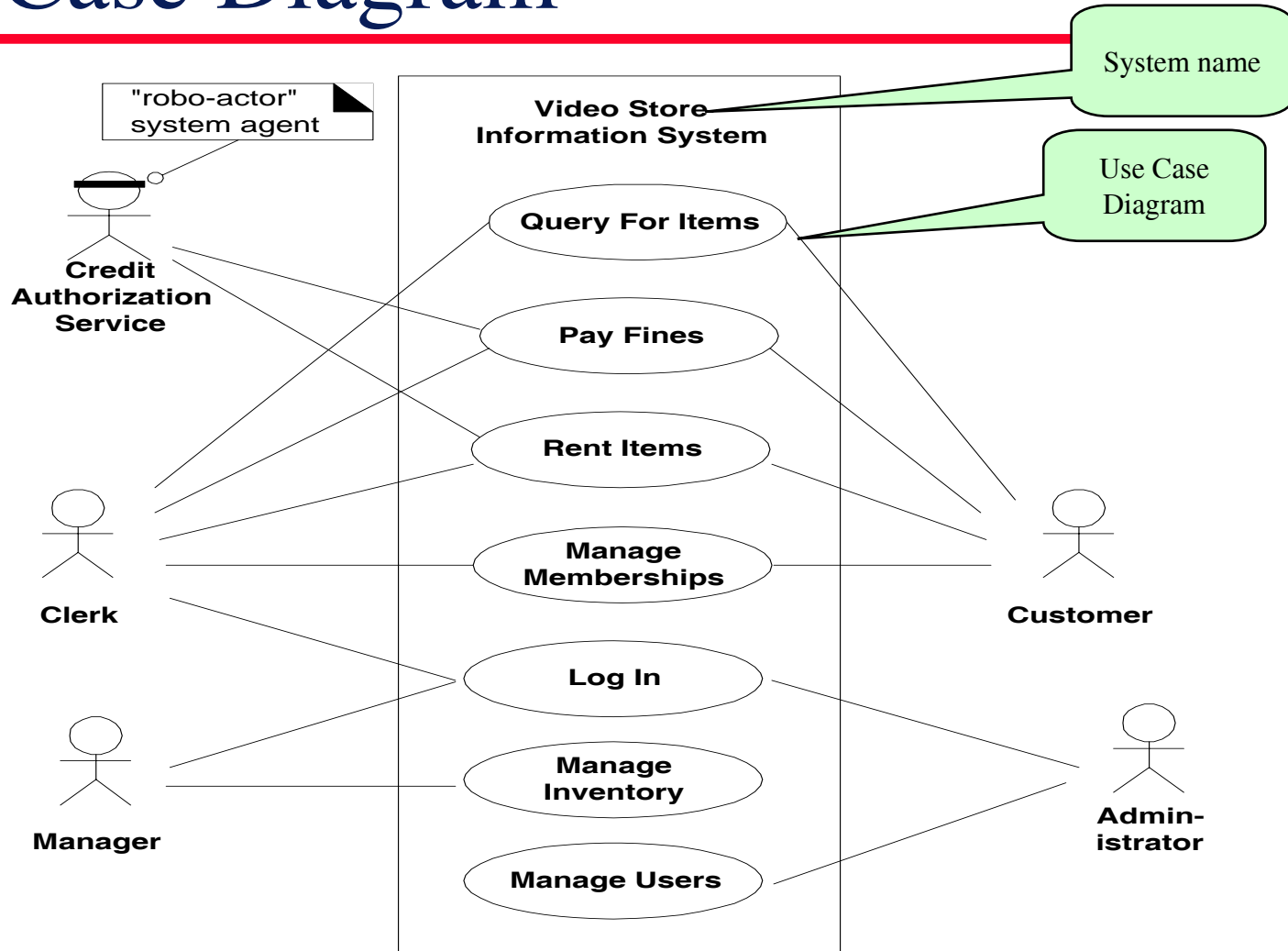
Sales activity system is a remote application that will frequently request sales data from each POS node on the network

Actor	Goal
Cashier	<ul style="list-style-type: none">•Process sales•Process returns•Cash in•Cash out
Manager	<ul style="list-style-type: none">•Start up•Shut down
System administrator	<ul style="list-style-type: none">•Add users•Modify users•Delete users•Manage security
Sales activity system (external computer sys)	<ul style="list-style-type: none">•Analyse sales & performance data

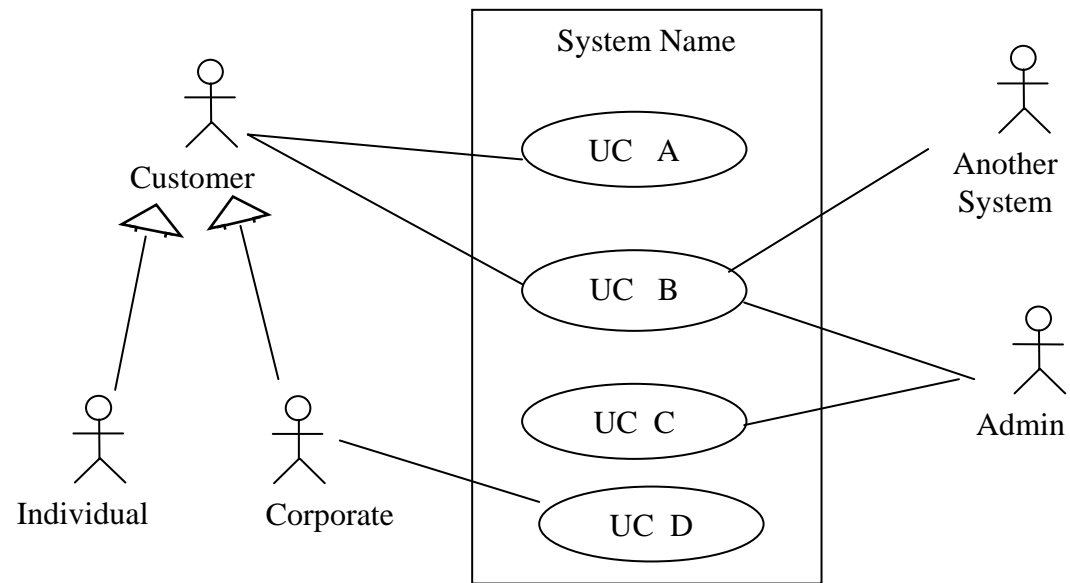
Important:

Be suspicious if you can't find primary actors (as external computer sys, ...)

Use Case Diagram



UC Diagram



Use Case Text (description)

Format

- One-column format
- Two-column format

- Prefer two-column format: clearly shows Actor action and system response

- For each step: **Actor does** x (or **system does** y)

Use Case Text: Sections

- Name
- ID
- Primary actor
- Stakeholders
- Goal
- Priority
- Risk
- Trigger
- Relationships: Association, Includes, Extends

Use Case Text: Sections (cont.)

- Input
- **Pre-conditions**
 - *What validity checks or constraints apply on the inputs (or the internal system as a whole, in some cases)*
- **Normal (Basic) flow of events – Happy path –Successful path - Main Success Scenario**

- **Alternatives: successful scenarios**
- **Exceptional flows: failure**

extensions

- **Post-conditions:**
 - *What changes does the Use Case make to the internal system state*
- Output
- Test Cases: Unit tests and functional tests
- Use Case Points UCP: for cost estimation

Use Case Text - **Extensions**

- For alternatives or additions to the main success scenario
- **Scenario: sequence of steps**
- Each extension is described separately
- Each extension describes a new UC that extends the main UC
- The **last step** in an extension may be:
 - **Fail:** UC goal unsatisfied
 - **Stop:** UC goal satisfied
 - **Resume n:** Continue with next step n in the main scenario

Brief Use Case

Example:

- **Use Case:** Identify Client
- **Primary Actor:** Client
- **Trigger:** Client inserts his ATM (Automatic Teller Machine) card
- **Goal:** The intention of the Client is to identify him/herself to the System. A project (operational) constraint states that identification is made with a card and a personal identification number (PIN).
- **Basic (Normal) Flow** “Successful Scenario”, “Happy Path”
- **Alternate Flows** “Extensions”: leads to successful use case
- **Exceptional Flows:** leads to failure of use case

Basic Flow “Successful Scenario”, “Happy Path” – One column format

Basic Flow “Successful Scenario”, “Happy Path”:

1. Client provides Card Reader with card.
2. System validates card type.
3. Client provides PIN to System.
4. System requests BAT System to verify identification information*.
5. BAT System informs System that identification information is valid, and System informs Client.

Basic Flow “Successful Scenario”, “Happy Path” – **Two- column format**

- **Two- column format: conversational**
- **Example:**

Actor action	System responsibility
1. Client provides Card Reader with card.	2. System validates card type.
3. Client provides PIN to System.	4. System requests BAT System to verify identification information
	5. BAT System informs System that identification information is valid, and System informs Client.

- **Use the format you are comfortable with.**

Extensions

Example of Extensions Flows:  Failure / Success / Resume

(1-6)a. (at any time) Client cancels the identification process.

(1-6)a.1. System requests Card Reader to eject card; use case ends in **failure**.

2a. System ascertains that card type is unknown:

2a.1. The System informs the Client and requests the Card Reader to eject the card; the use case ends in **failure**.

2b. System informs Client that it is currently "out of service": use case ends in **failure**.

3a. System times out on waiting for Client to provide PIN:

3a.1. System requests Card Reader to eject card; use case ends in **failure**.

Alternative Flows

5a. BAT System informs System that password is incorrect:

5a.1. System informs Client and prompts him/her to retry; use case **continues** at step 3.

5a.1a. System ascertains that Client entered an incorrect PIN for the third time:

5a.1a.1. System swallows card and informs Client to see Bank for further details; use case ends in **failure**.

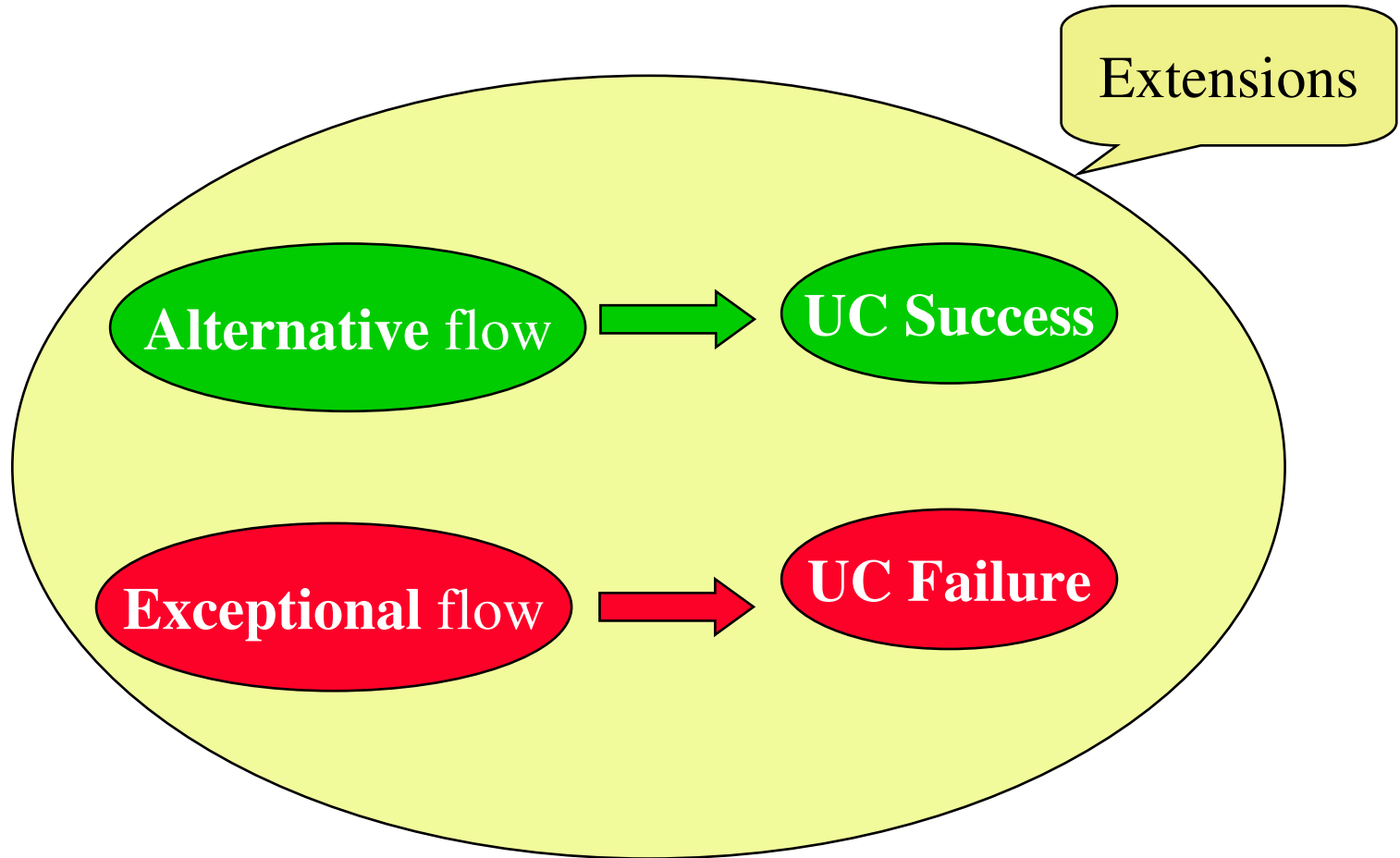
5b. BAT System informs System that card information is invalid:

5b.1. System informs Client and requests Card Reader to eject card; use case ends in **failure**.

5c. System is unable to communicate with BAT System:

5c.1. System informs Client that it is now out of service and requests Card Reader to eject card; use case ends in **failure****.

Extension: Alternative & Exceptional Flows



Extension: Alternative & Exceptional Flows

An extension may be:

- An **exceptional** flow leading to **failure** of the use case
- An **alternative** flow leading to **success** of the use case
- **exceptional flow**

UC Diagram: The 4 types of associations / relationships

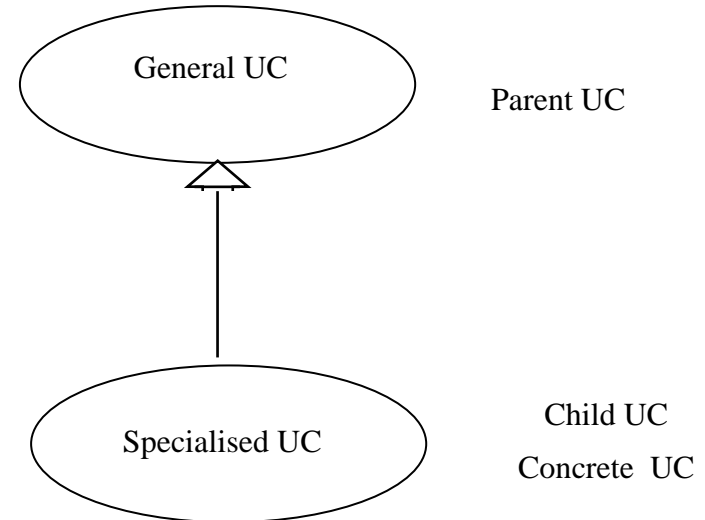
UC Diagram: The 4 types of associations / relationships

- **Generalisation** between:
 - actors
 - use cases
- **Include** relationship between use cases
- **Extend** relationship between use cases

Generalisation between use cases

- Like generalization among classes
- A child UC **inherits the behaviour & meaning** of the parent UC
- The child UC **may add to or override** the behaviour of its parent UC
- The child UC may be substituted any where the parent UC appears (both parent & child UCs may have concrete instances)
- Is often implemented by inheritance

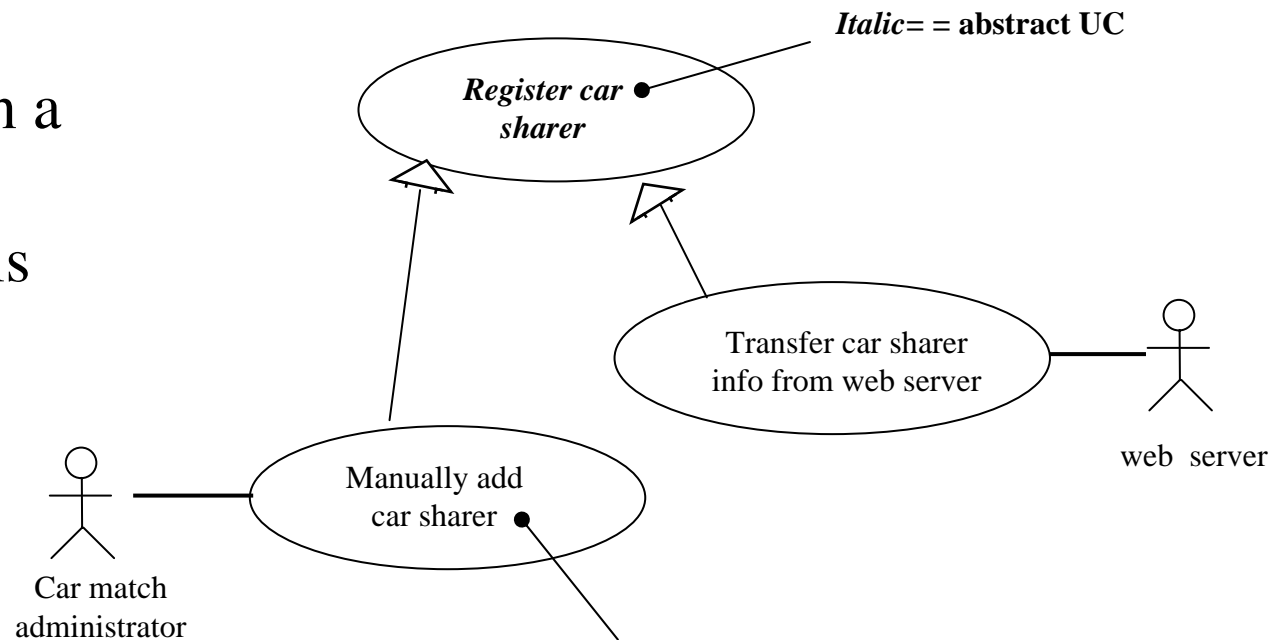
Generalisation between use cases:



Generalisation between use cases: Abstract UC

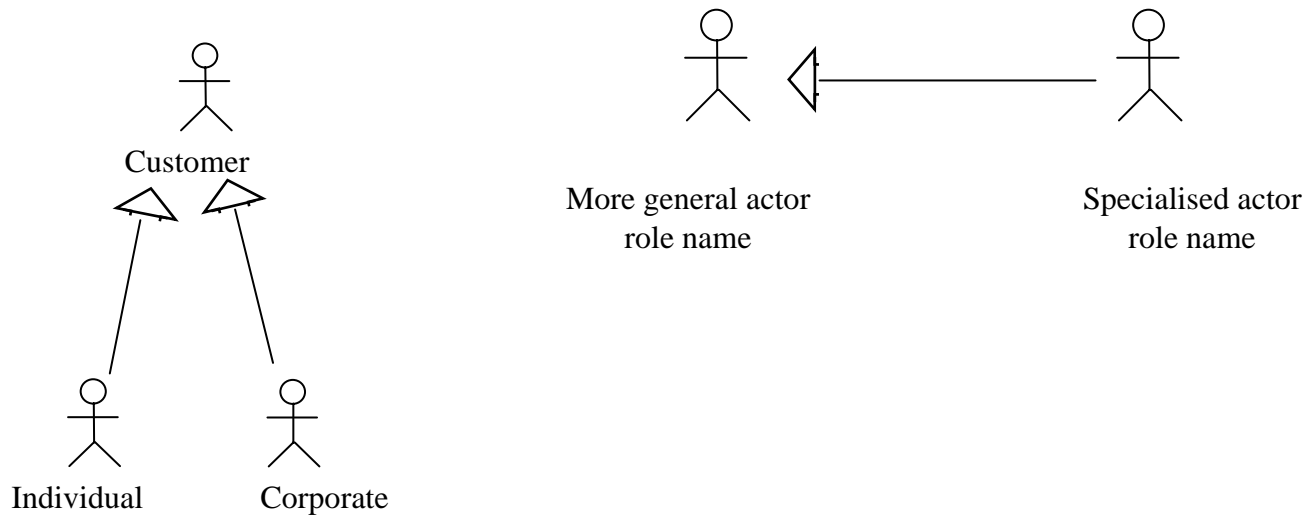
Abstract UC:

- general UC that will never exist in a real sys
- it defines what is **common** to specialized UCs

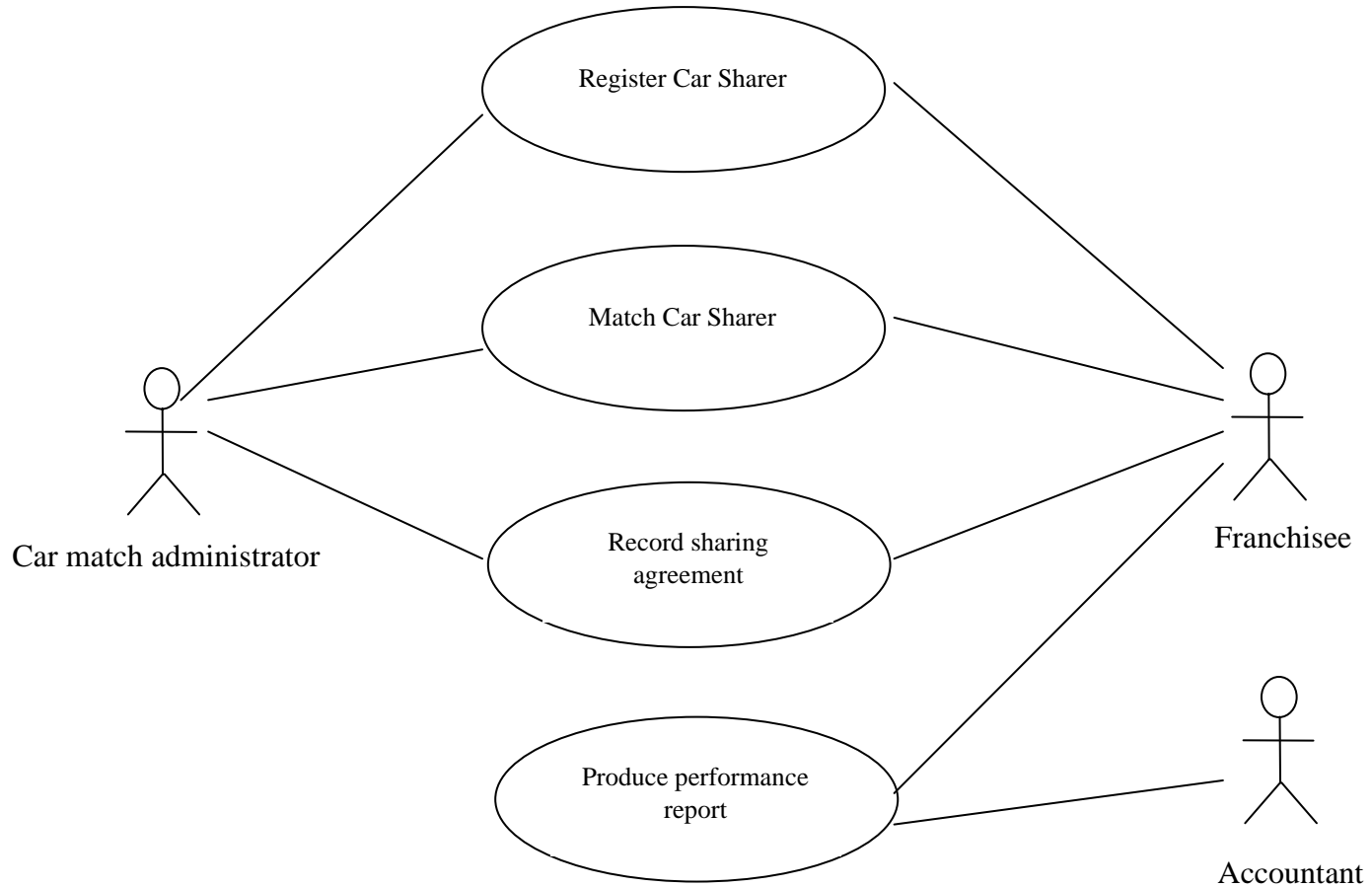


- *name in italic* or using {abstract}

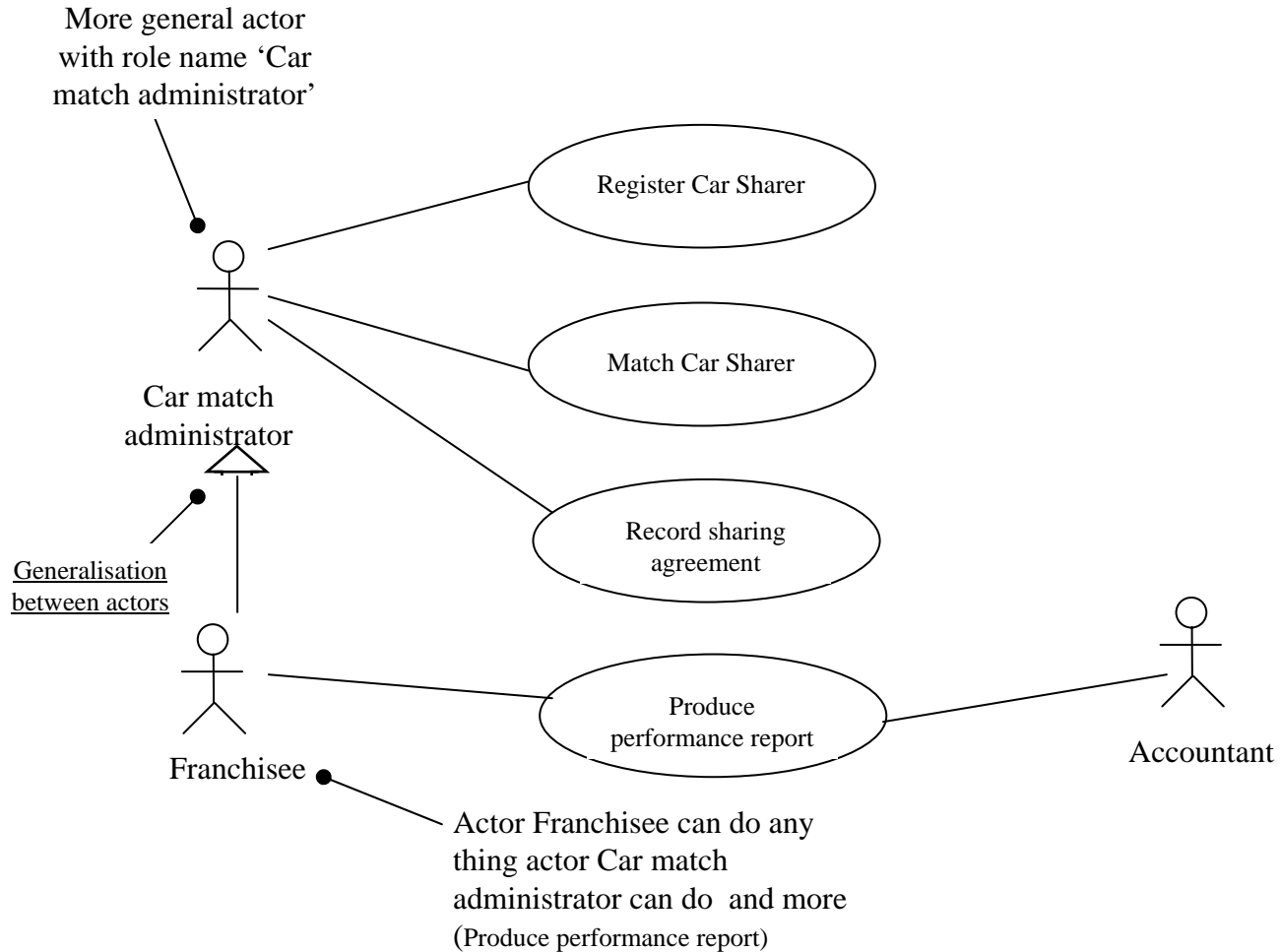
Generalisation between Actors



Example: No generalisation between actors



Same ex with Generalisation between actors

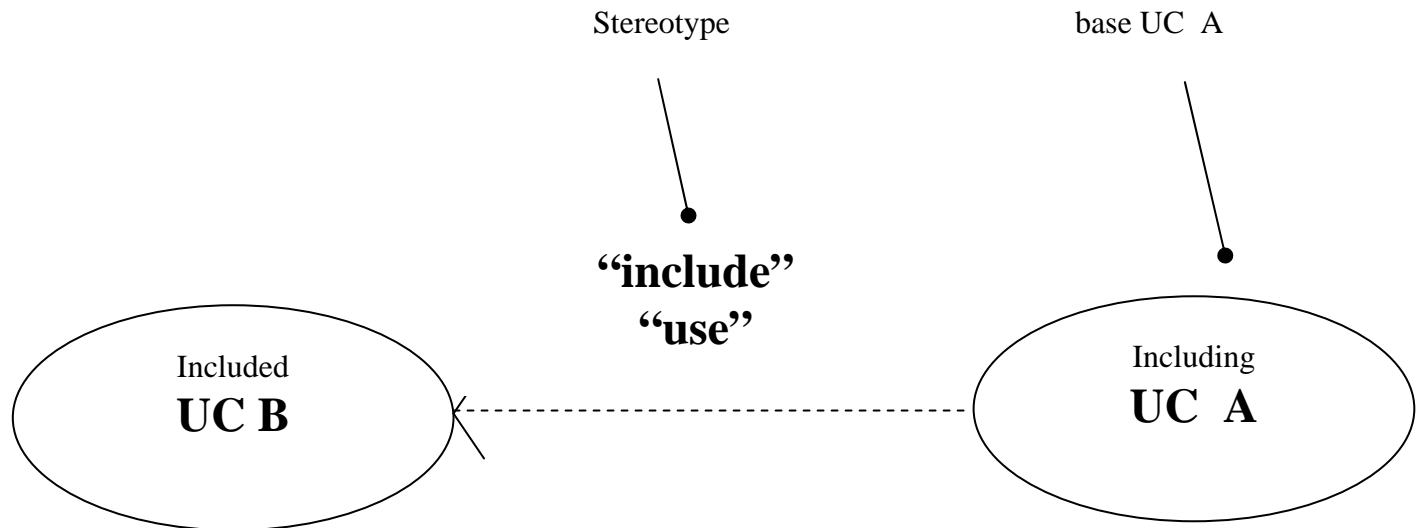


Include Relationship between Use Cases

- **Mandatory** Behaviour
- One UC **always includes** another
- A base UC explicitly include the behaviour of another UC at a location specified in the base UC
- Encapsulate some functionalities in the included UC that is used at several points (avoid repetition)
- UC A ‘include’ (or ‘use’) UC B

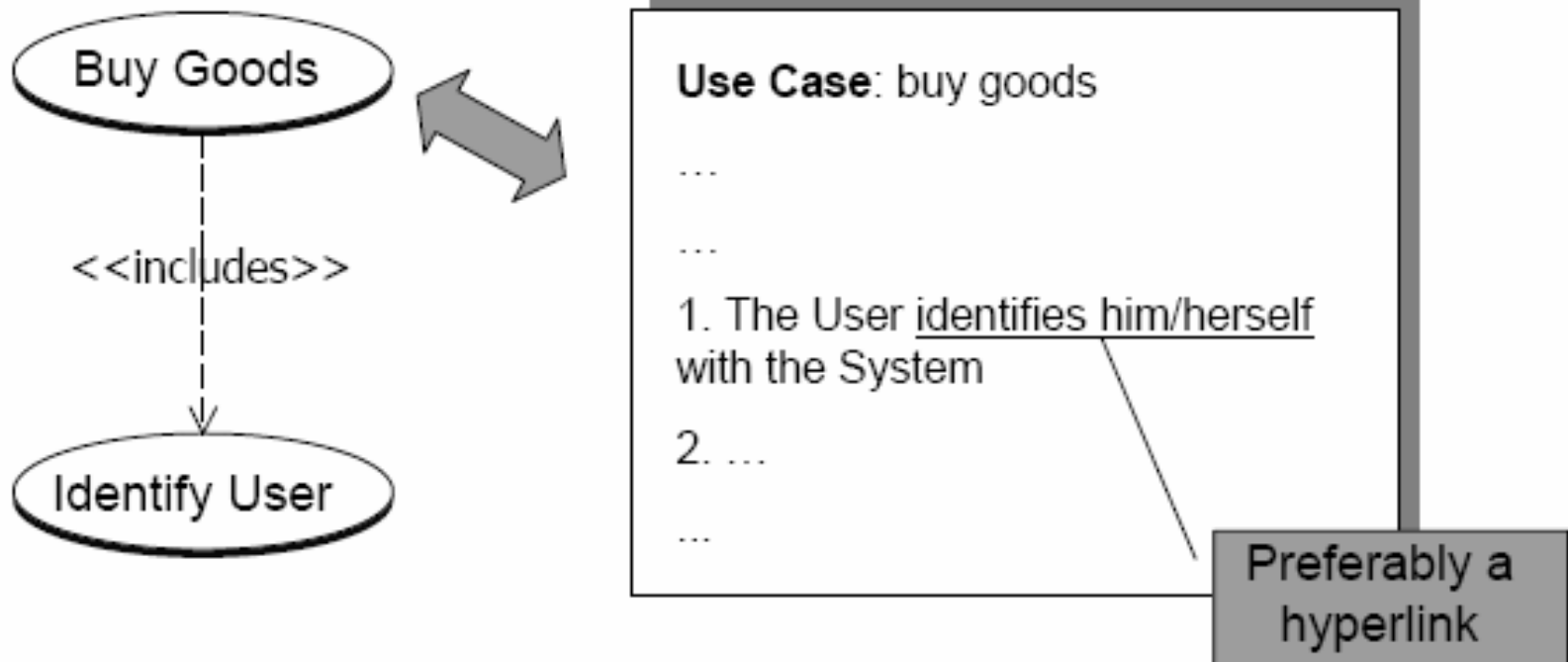
Include Relationship between Use Cases

- UC A **‘include’** (or “use”) UC B



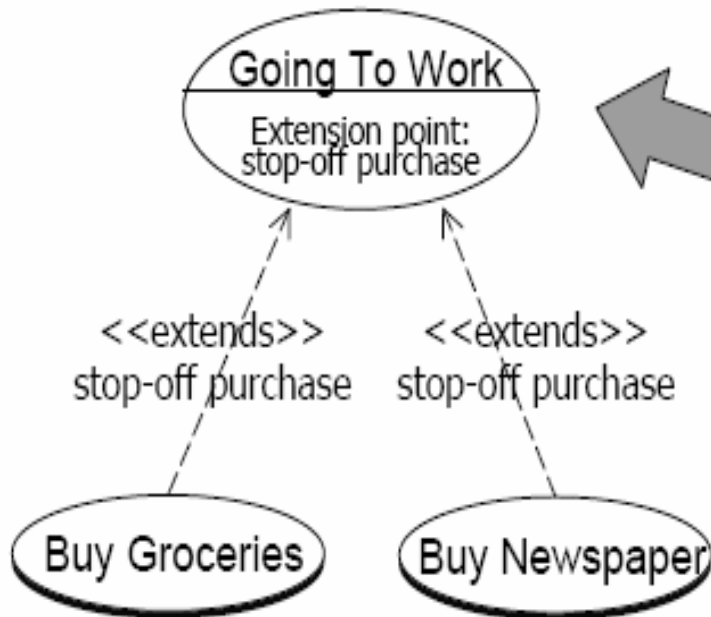
Include Relationship between Use Cases

- **Mandatory** Behaviour: 'Buy Goods' use case **always includes** 'Identify User' use case



Extends Relationship between Use Cases

- **Optional behavior**
- ‘Buy Newspaper’ UC **Optional extends** the behavior of ‘Going To Work’ UC (at a specified location within ‘Going To Work’ UC).



Use Case: going to work

...

Basic Flow:

...

4. Worker leaves train station

...

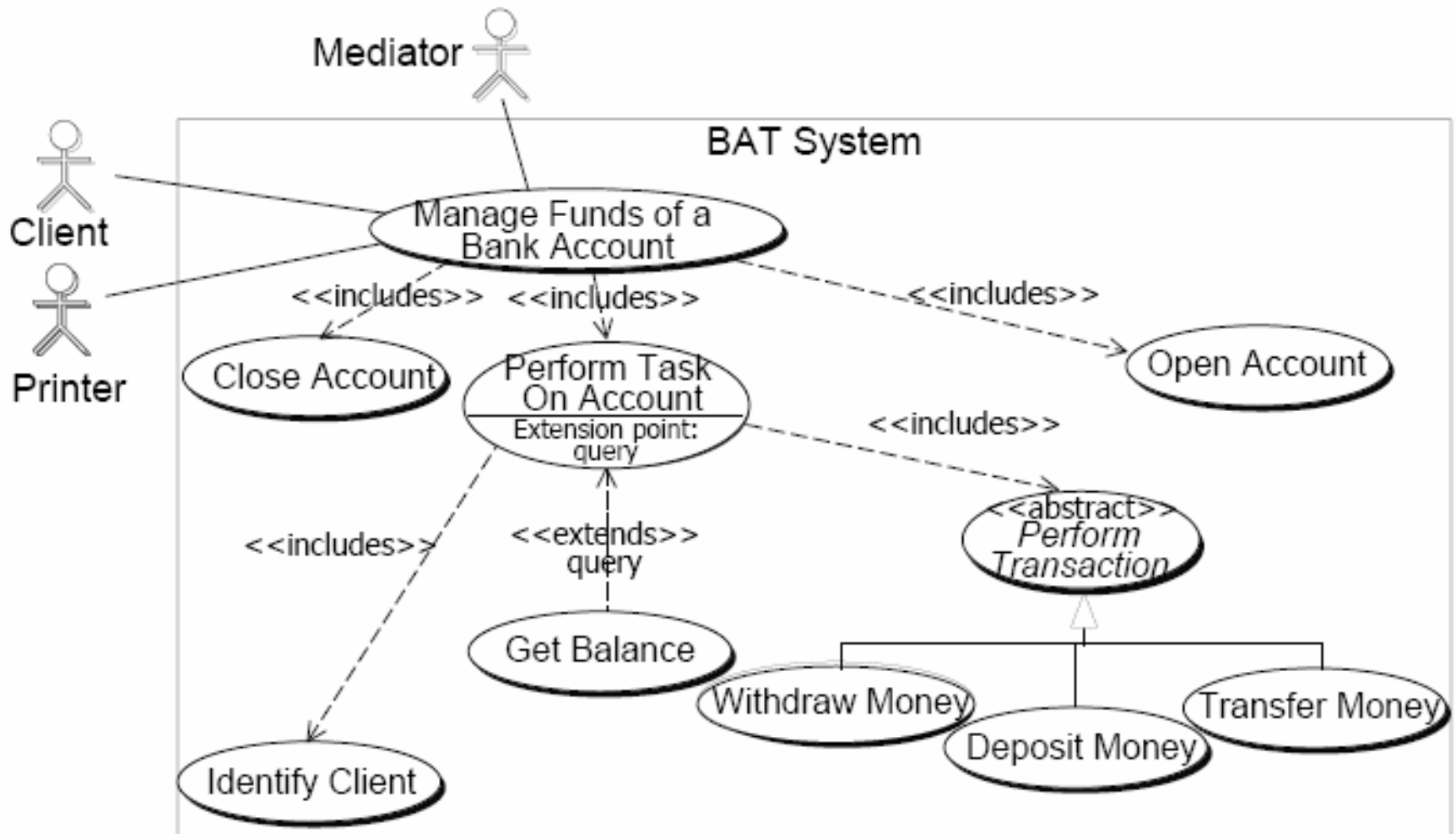
Alternative Flows:

...

4||a. The Worker makes a purchase
[extension point: stop-off purchase]

...

UML Use Case Diagram: Example



EBP: Elementary Business Process

EBP is a process:

- performed in response to business event
- adds measurable business value
- leaves the data in consistent state

Ex:

- Register course
- Drop course
- Process sale
- Issue PO

- A UC represents a complete functionality serving a goal for the actor
 - **Set of actions**; not a single action

- **Do not include UCs such as: delete an item, add new item, etc...**

Artifacts used with a Use Case

Artifacts used with a Use Case

Artifact: is a man-produced work-product:

- Pseudo code
- Table
- Database schema
- Text document
- Diagram
- Models
- Web graphic
- Test case
- etc

Artifacts used with a Use Case

- A use case may be supplemented with one or more artifacts- links to explain an issue
- A decision logic needs to be documented
- Example: decision logic for grading
 - if mark ≥ 95 then grade = A+
 - if mark ≥ 90 and < 95 then grade = A
 -

Artifacts for Logic Modelling

Logic Modelling of a step within use case scenario is expressed via artifacts:

- Structured English - Psuedo code
- Decision Tables
- Decision Trees

Logic Modelling: Decision Tables

Conditions		Rules				
		R1	R2	R3	R4	...
C1						
C2						
Actions						
A1			X			
A2				X		
A3		X				
A4					X	

Reads: If C1 **AND** C2
then Action Ai

Logic Modelling: Decision Tables

Number of rules

If

- C1 has n1 values
- C2 has n2 values
-
- Ci has ni values
-
- Cm has nm values

Number of rules = $n_1 \times n_2 \times \dots \times n_i \times \dots \times n_m$

$$= \prod_{i=1}^{i=m} (n_i)$$

Logic Modelling: Decision Tables

How to create a decision table

- Name the condition and values each condition can assume
- Name all possible actions that can occur
- List all rules
- Define the actions for each rule
- **Simplify** the table

Logic Modelling: Decision Tables

Example: Decision table for a payroll system

- **Employee type: 2 values**
 - **Salaried 'S' :**
 - **By hour 'H':**
- **Hours worked: 3 values**
 - **< 40**
 - **= 40**
 - **> 40**
- **Number of rules = 6**

Logic Modelling: Decision Tables

Example: **Complete** decision table for a payroll system

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type (S/H)	S	H	S	H	S	H
	Hours worked (<40/40/>40)	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Logic Modelling: Decision Tables

Example: **SIMPLIFIED** decision table for a payroll system

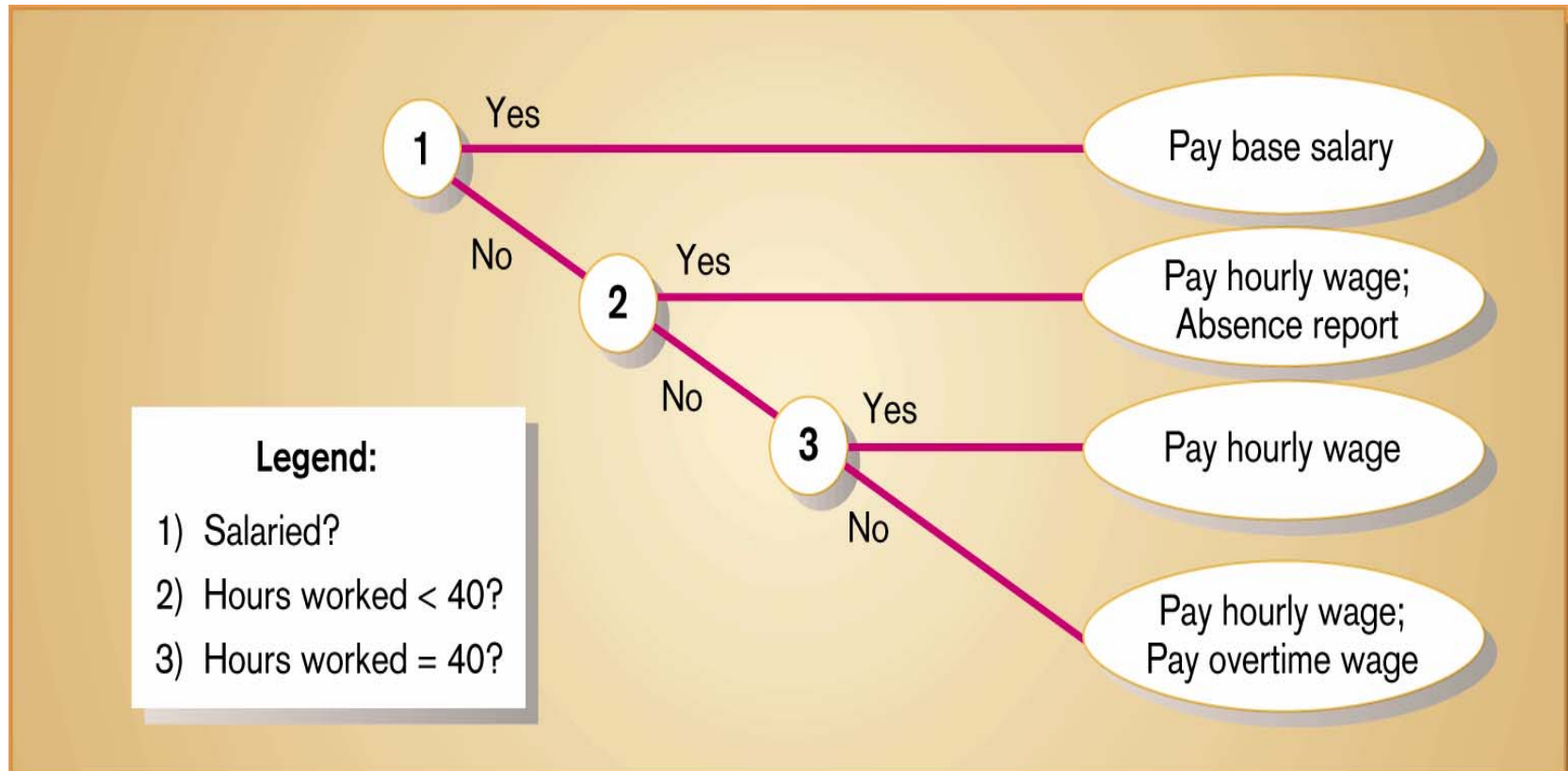
Conditions		Rules			
		R1	R2	R3	R4
C1	Employee type (S/H)	S	H	H	H
C2	Hours worked(<40/40/>40)	-	<40	40	> 40
A1	Pay base salary	X			
A2	Calculate hourly wage		X	X	X
A3	Calculate overtime				X
A4	Produce absence report		X		

Logic Modelling: Decision Tree

- A graphical representation of a decision situation
- Two elements:
 - **Nodes**: Decision points
 - **ovals**: Actions
- Read from left to right
- All possible actions are listed on the far right

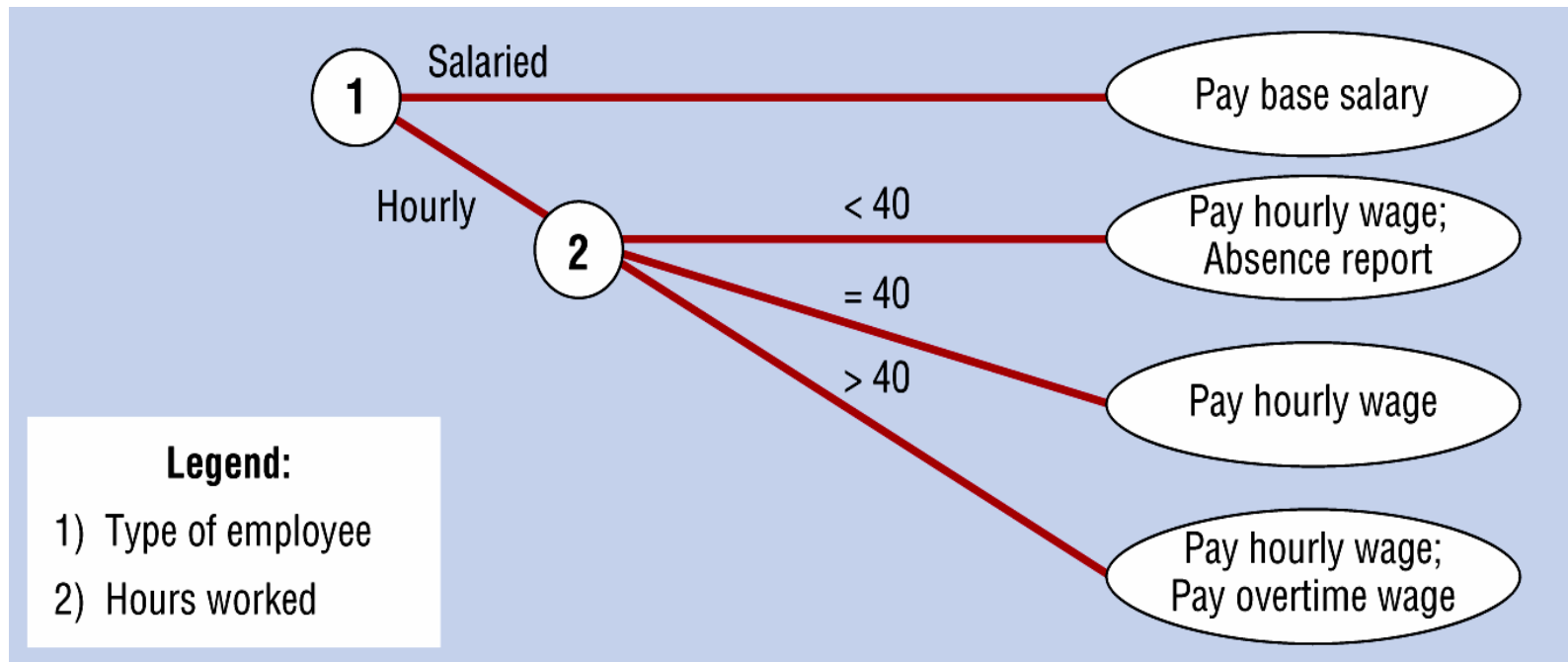
Logic Modelling: Decision Tree

Two choices per decision point



Logic Modelling: Decision Tree

Multiple Choices per Decision Point



Comparison of Logic Modelling Artifacts:

Criteria	Structured English	Decision Tables	Decision Trees
Determining Conditions and Actions	Second Best	Third Best	Best
Transforming Conditions and Actions into Sequence	Best	Third Best	Best
Checking Consistency and Completeness	Third Best	Best	Best

Decision table Example for System Use Case: Process Life Insurance Application

Basic flow:

1. User enters application information.
2. **System validates eligibility. (See accompanying decision table artifact.)**
- 3 System adds application to “Adjuster” queue.

Alternate flows

3a Referred application:

- .1 System adds application to referral queue.
- .2 The use case ends (**success**).

Exception flows

3b Rejected application:

- .1 System adds application to rejection queue.
- .2 The use case ends in **failure**.

Ex: System Use Case: Process Life Insurance Application

Artifact Decision Table: Validate eligibility

		Rules							
		1	2	3	4	5	6	7	8
C O N D I T I O N	Medical condition (<u>P</u> oor/ <u>G</u> ood/)	P	P	P	P	G	G	G	G
	Substance abuse? (Y/N)	Y	Y	N	N	Y	Y	N	N
	Previous rejections? (Y/N)	Y	N	Y	N	Y	N	Y	N
A C T I O	Accept								X
	Reject	X	X	X		X		X	

Identifying actors and use cases

Identifying actors and use cases

- Brainstorming session
- Actors have goals (needs)
 - The sys should respond to actor's goals
- Tasks hierarchy:
- Tasks can be grouped at many levels of granularity:
 - One / few small steps
 - Up to organisation-level activity
- Q: At which level & scope should use cases be expressed?
....
- A: Focus on UCs at the EBP level

EBP: Elementary Business Process

A process

- performed in response to a business event
- adds **measurable business value**
- leaves the **data in consistent state**

- Ex of valid UC:
 - Process sale, Price items, Issue PO
- A UC represents a complete functionality serving a goal for the actor
 - Set of actions; **not a single** action

Example on EBP: Point Of Sales (POS) sys

Interview questions& answers between System analyst (SA) & cashier

- Q1: SA: what are your goals in the context of POS sys
- A1: Cashier: To quickly login and to capture sales

- Q2: What is the higher level goal for logging in?
- A2: to identify myself to the sys

- Q3: higher than that?
- A3: to prevent theft, data corruption

- Q4: higher than that?
- A4: to process sales.

Example on EBP: Point Of Sales (POS) sys

From the above:

- “Prevent theft “ is higher than a user goal – more as org goal
- “identify myself to the sys..”
 - is close to user goal level...
 - but is it at EBP level ????
 - It does not add observable / measurable business value ..
 - If the manager asked “what did you do today?” the answer “I logged in 10 times” is not useful to the business..
 - Thus, this is a secondary goal in the service of doing something useful
 - It is not an EBP or user goal
- Conclusion: “Process sales” is a user goal satisfying EBP

UC Best Practices

Identify user goal-level UCs -

- that serve each goal of the primary actor
- Use EBP guide lines if a UC is at a suitable level: goals hierarchy and sub goals

Find the user goals:

- **ask: what are your goals?**
- **Do not ask what are the use cases**
- **Do not ask what do you do?**

UC Best Practices

- Define a use case for each goal
- Answers to the question ‘**what are your goals?**’ combined with a question to move higher up the goal hierarchy (‘**what is the goal of this goal**’) will:
 - open up the vision for new & improved solutions
 - focus on the business
 - get to the heart of what the stakeholders want from the system
- Goals may be composed of many sub goals (sub functional goals) leading to a primary UC and sub UCs

System boundary

- What is **inside**?
- If difficult to identify, then define what is **outside** as external primary and supporting actors

Primary & Supporting Actors

- Primary actors: **have goals** to be fulfilled through sys services
- Supporting actor: **provide services** to the system under **design**

Finding Primary Actors Goals & Use Cases

Use cases are defined to satisfy the goals of the primary actors:

- 1) Choose the sys boundary
- 2) Identify primary actors
- 3) For each primary actor, identify their user goals.
 - Raise goals to the highest goal level that satisfy the EBP guidelines
- 4) Name a UC according to its goal:
 - usually, a 1:1 relationship
 - common exception to 1:1 relationship: **CRUD** (create, retrieve, update, delete) goals into one CRUD UC called “**manage**”

Create Read Update Delete (CRUD) Use Cases

- Create, Read, Update and Delete are performed on a common (business) object, but each one corresponds to a separate goal.
- Start with a higher-level use case (often summary), **Manage <business object>**
 - Easier to track
 - Break out any complex CRUD units into a new use case

Create Read Update Delete (CRUD) Use Cases

- **Don't put CRUD use cases first (avoid seeing a tree as a forest),**
- instead keep focused on use cases that provide the most value to the primary actors.

Who-what & why?

- Who are the people who will use the sys **and why (their goals of use)?**
 - People who will enter info to the system
 - people who will receive info from the system
 - people who will interact with the system n(administration)
 - etc..
- What are the **systems** that the current sys will interact with and their goals?

Use Case Diagram versus Context Diagram

OO versus Structured Development

Increased development time :

- It often takes until the **3rd or 4th project** before development time improvements over the structured approach are realized (due to a steep learning curve and the fact that it takes a few projects until you have a library of reusable items).

Longer run-time:

- Systems programmed using OO take longer to run:
 - OO language C++ is about **10% slower** than its structured counterpart, C.

OO versus Structured Development

Higher Cost:

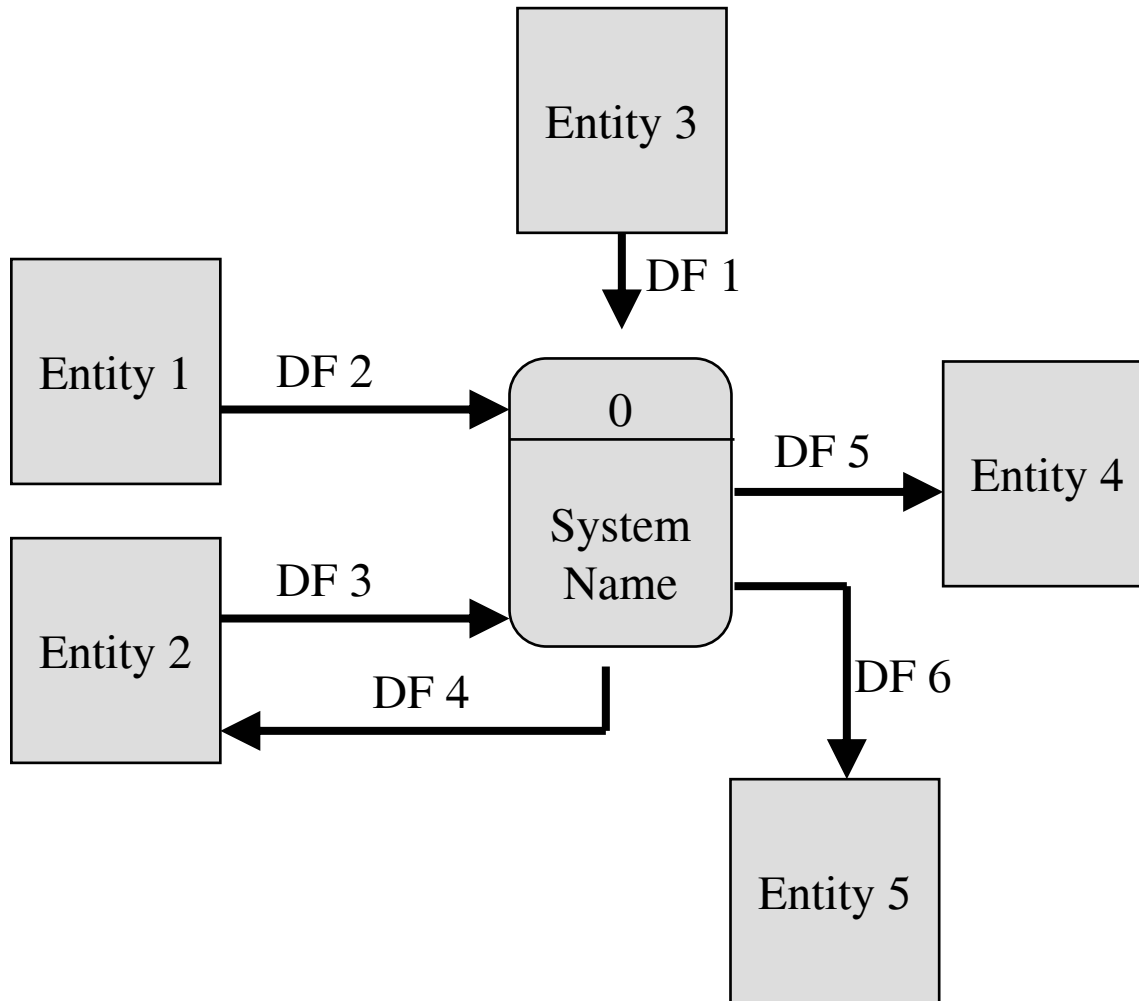
- OO Software Analysis and Design tools to support modeling tend to be **more expensive** than their “structured” counterparts

Oversell:

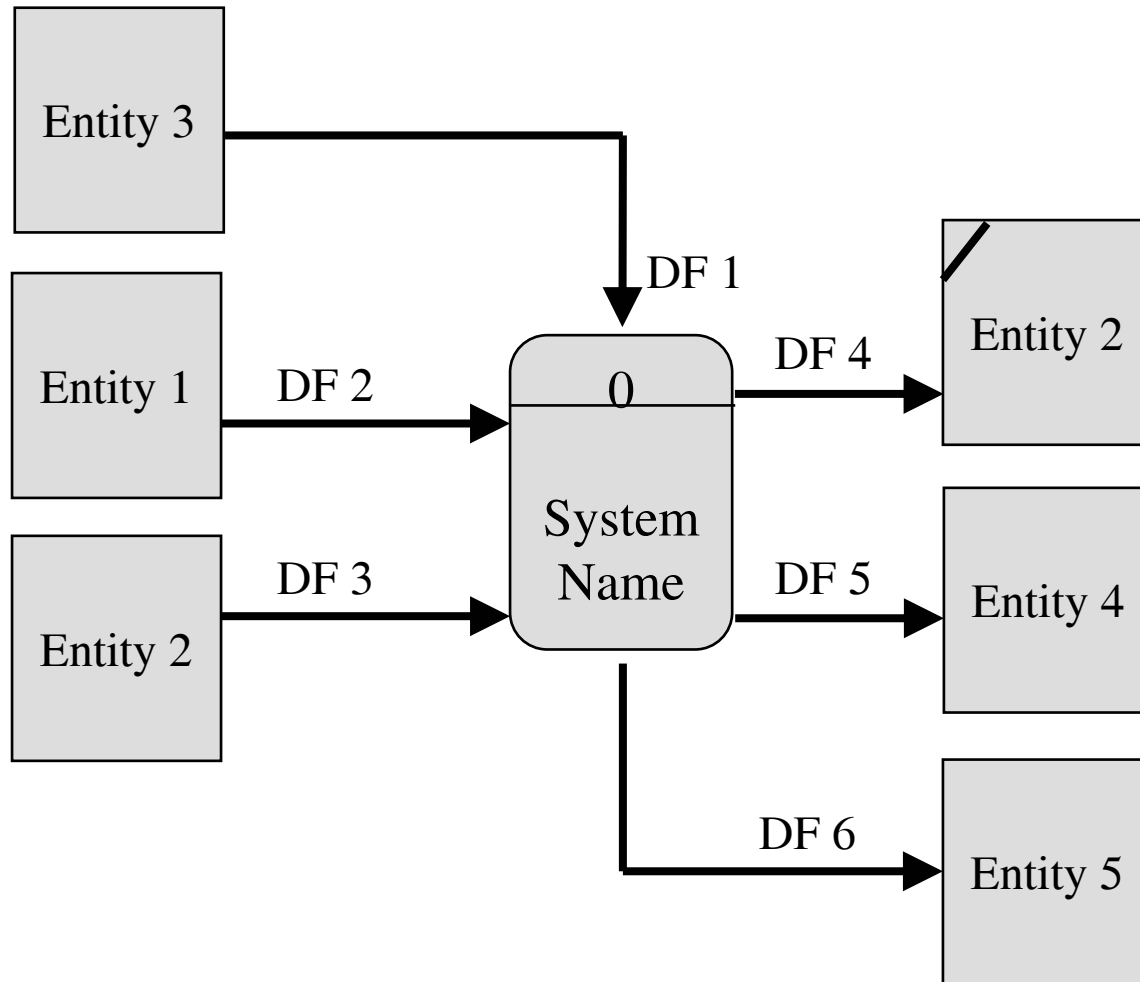
- OO is not as intuitive as its adherents claim. When OO proponents say the approach is “intuitive”, what they really mean is that it is based on ways of structuring thought that are inherent in the human mind – but that does not mean that OOA is easy to do.

.

Structured Approach: Context Diagram “System Interface”



Structured Approach: Context Diagram “System Interface”



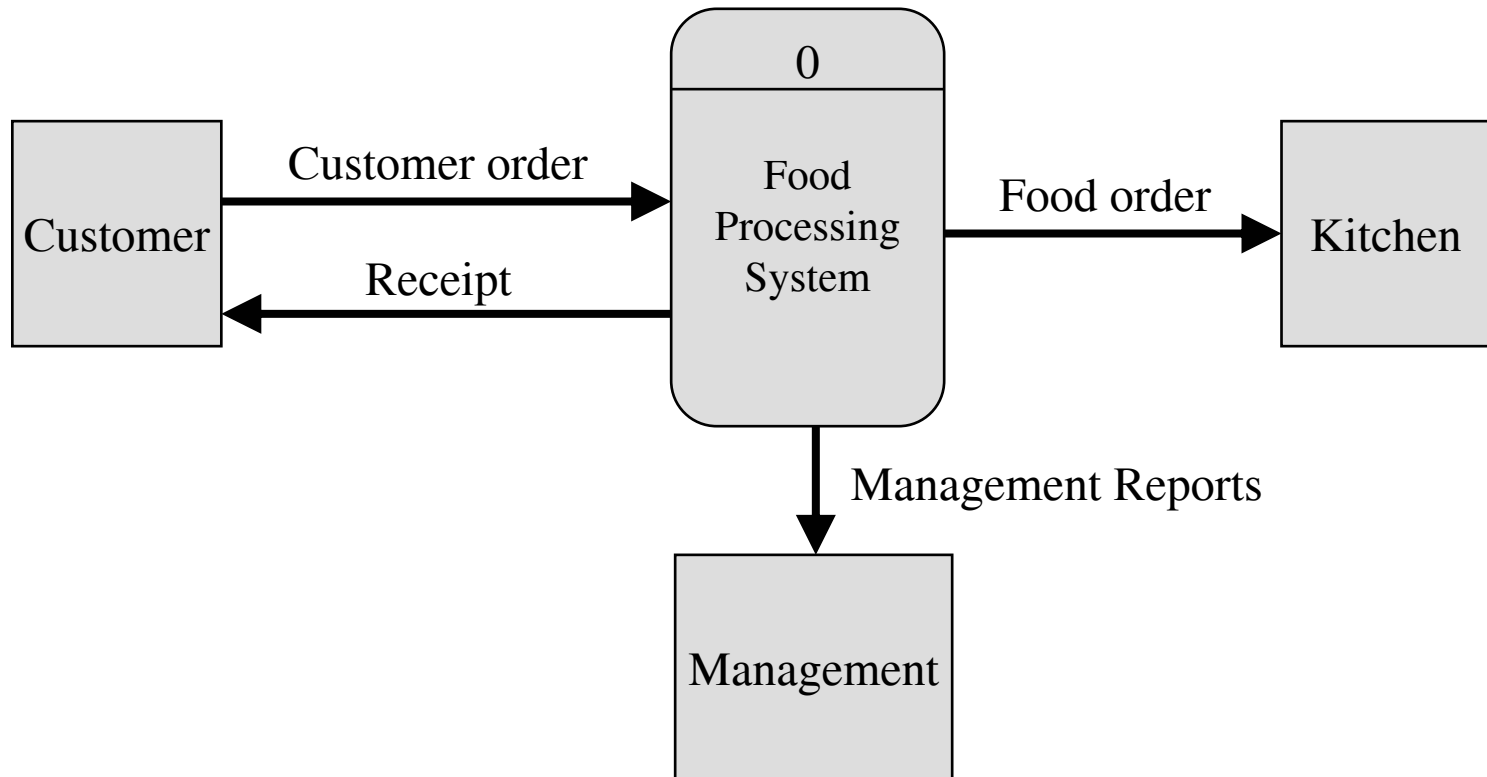
Structured Approach: Context Diagram

- **System Interface**
- **Bird's view of the system**

- **Context Diagram**
 - Shows the **scope** of the system
 - Shows system **boundaries**
 - Shows **external entities** that interact with the system
 - » An entity may be another **system** interfacing with the system under study (ex: Bank, GOSI)
 - Shows I/O flows between the entities and the system

Structured Approach: Context Diagram “Sys Interface”

Example: Food Ordering System



Structured Approach: Context Diagram Entity

A gray rectangular box with a black border containing the text "Entity Name" in a serif font, centered within the box.

Entity
Name

An Entity “External Entity” may be:

- A **person** interacting with the system (ex: Student , Customer).
 - The person may be inside or outside the business unit under study
- Another **organization** sending/receiving info (ex: Supplier, Department)
- Another **system** interfacing with the system under study (ex: Bank, GOSI)

Use Case Diagram versus Context Diagram

