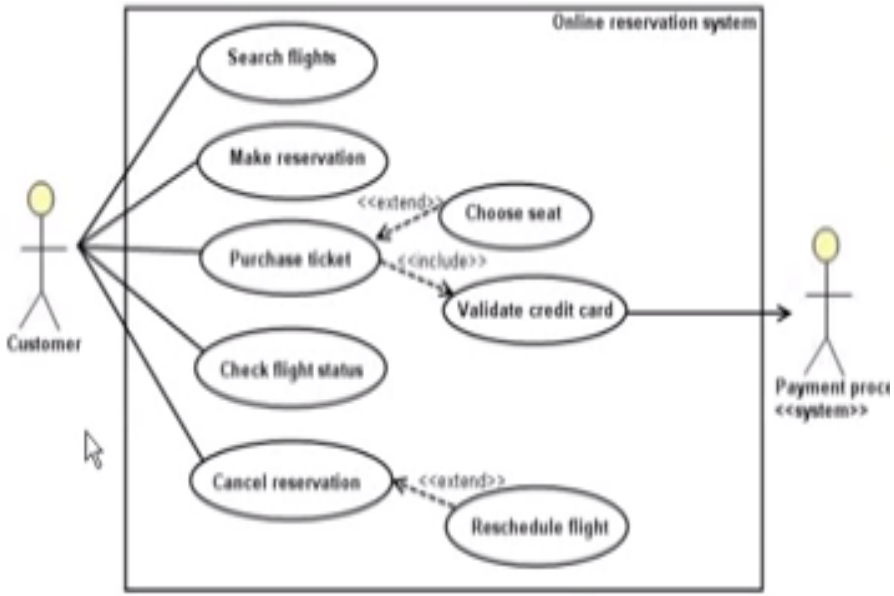
MONASH University

Information Technology

FIT2001 – Systems Development

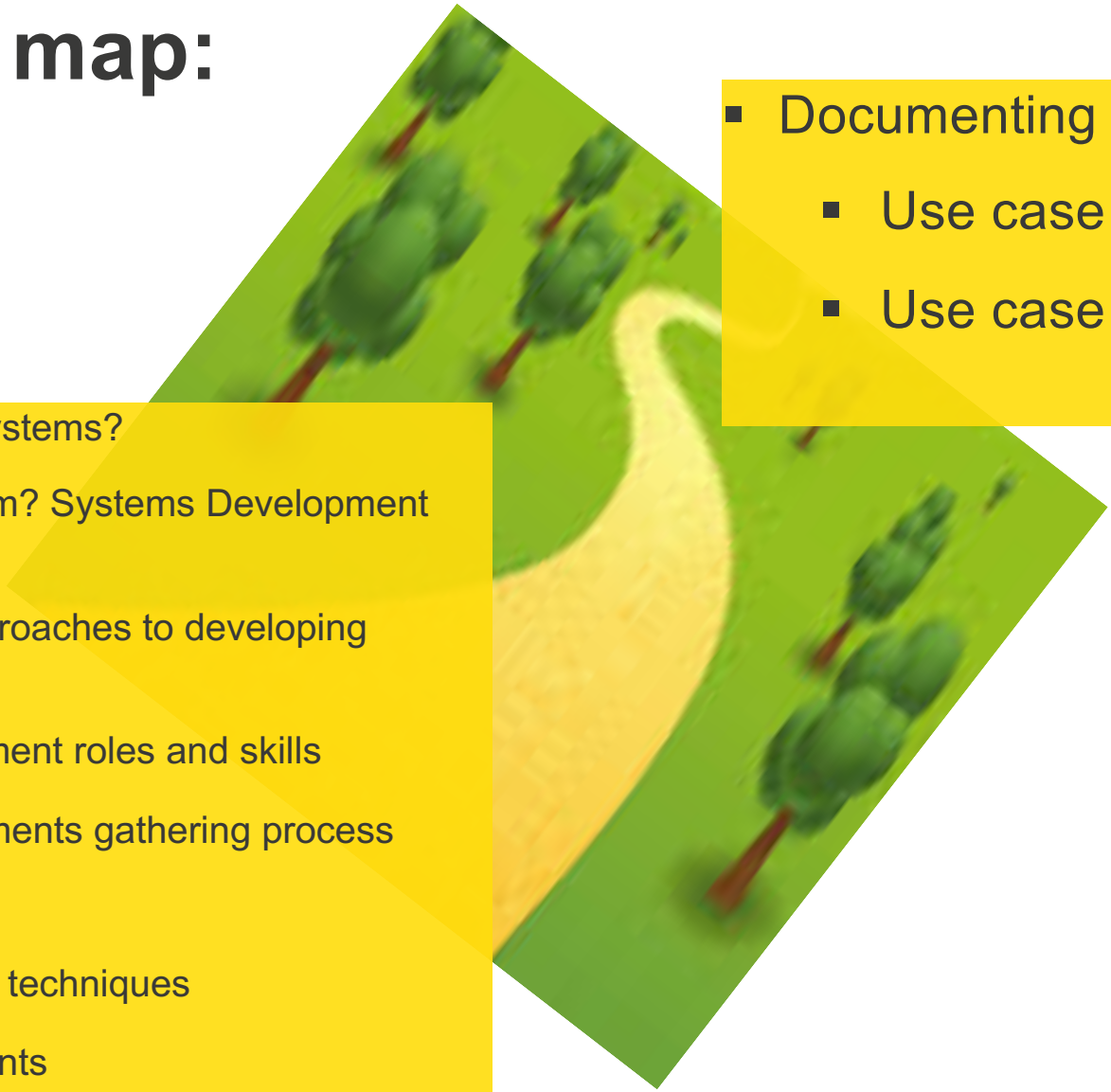
Seminar 5: Investigating and documenting system requirements
Use Case Diagrams & Use Case Descriptions

Chris Gonsalvez



Use Case Template	'Create Order' Use Case
Use Case Name	*Create order*
Use Case Description	*Create order is the ability to request the purchase of a product*
Actor	*Order Creator*
Pre-conditions	• *Order Creator has been identified*
Basic Flow	1. *Order Creator selects 'order product' action* 2. *System requests customer/product identification information* 3. *Order Creator provides customer/product identification information* 4. *System requests mailing information* 5. *Order Creator provides mailing information* 6. *System verifies mailing information* 7. *System requests order be submitted* 8. *Order Creator submits order* 9. *System submits product order for processing* 10. *System confirms product order*
Post-conditions	• *Product order has been created*
Alternate Flows	• *Product is not in stock* • *Product has been discontinued* • *A customer's initial order is over $200*

Our road map:



- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Requirements gathering techniques
- Documenting requirements

- Documenting requirements
  - Use case diagrams
  - Use case descriptions

At the end of this topic you will:

- Understand the purpose of Use Cases
- Be able to identify Use Cases
- Be able to draw Use Case diagrams and write Use Case descriptions

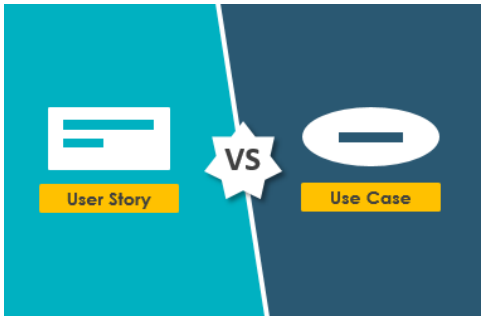
Use Cases

What are Use Cases?

- Invented by Ivar Jacobson in the late 1960s … started being used more frequently in late 1980s, not used as much in Agile
- A more traditional way of capturing functional requirements of a system - includes business rules and descriptions of system behavior
- A Use Case is a description of all the ways an end-user wants to "use" a system. Use Cases capture all the possible ways the user and system can interact that result in the user achieving the goal. They also capture all the things that can go wrong along the way that prevent the user from achieving the goal.

Use cases VS. User stories

User Stories	Use Cases
Focussed on the result and benefit of system functions	More granualar, describes how the system will act
Minimal documentation	Detailed documentation
Small increments for getting feedback	Most done up front



- Hybrid model - User Story as an important launching off point for the more detailed Use Case
- In Industry Use Cases used more often when projects have more stringent documentation requirements (e.g. government projects).

How to identify use cases

- User goal technique
  - Most commonly used in industry
  - User goals for interacting with the new system are identified
  - Simple and effective

- Event decomposition technique
  - Business events are identified

User goal technique.1

The following steps help identify use cases:

1. Identify all the users
2. Classify them
   - Functional roles – Sales, Marketing
   - Organisational roles – Executive, Management, Operational

User goal technique.2

Steps to identify use cases (cont.):

3. Interview each type of user to create a preliminary list of use cases
   - Find out *a list of specific goals* they will have when using the new system
   - Express them in VERB-NOUN format (e.g. Add customer)
   - Interview and ask them to describe the tasks the system can help them with
   - Probe further to refine the tasks into specific user goals, "I need to Ship items, Track a shipment, Create a return"
4. Look for duplicates … resolve inconsistencies
5. Look for users with same needs
6. Review with users

User goal technique: Example

User	User goal and resulting use case
Potential customer	Search for item Fill shopping cart View product rating and comments
Marketing manager	Add/update product information Add/update promotion Produce sales history report
Shipping personnel	Ship items Track shipment Create item return

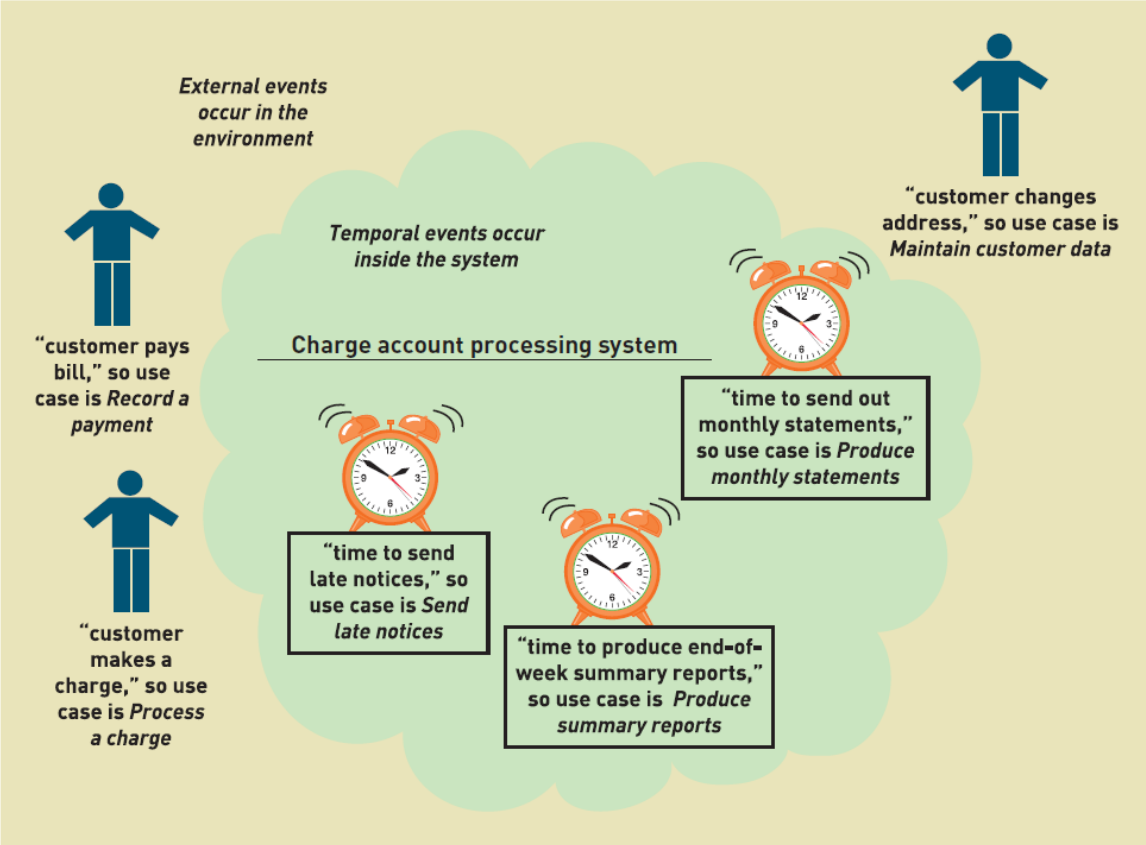
- Speak to potential customers through focus groups
- Interview users from shipping and marketing departments

Identify Use Cases - Event Decomposition Technique

- More comprehensive and complete technique
- Identify the events that occur to which the system must respond.
  - Event– something that occurs at a specific time and place, can be described, and should be remembered by the system
- For each event, name a use case (verb-noun) that describes what the system does when the event occurs

Types of Events – External Events

An event that occurs outside the system, usually initiated by an external agent or actor



Checklist:

Wants something resulting in a transaction - Customer buys a product

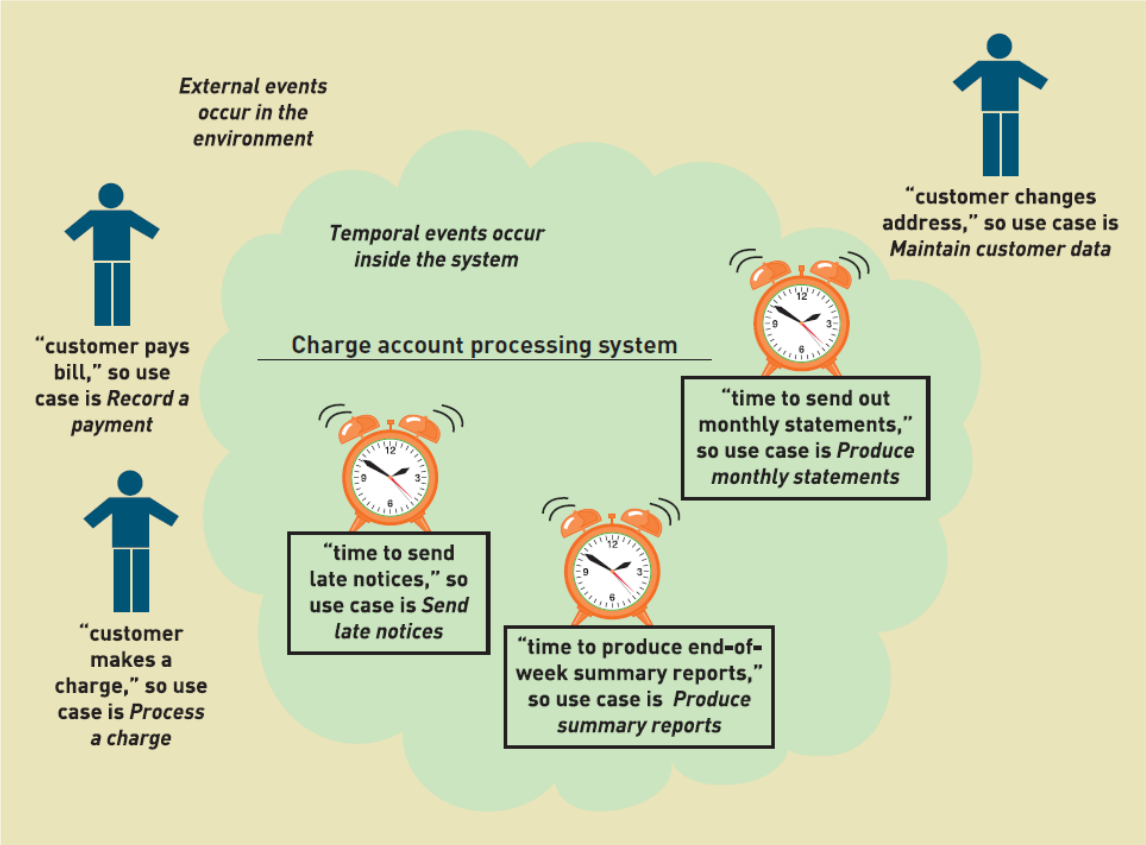
Wants some information - Customer wants to know product details

External data changed and needs to be updated - Customer has new address and phone

Management wants some information - Sales manager wants update on production plans

Types of Events – Temporal Events

An event that occurs as a result of reaching a point in time



Checklist:

Internal outputs needed at points in time

- Management reports (summary or exception)
- Operational reports (detailed transactions)
- Internal statements and documents (including payroll)

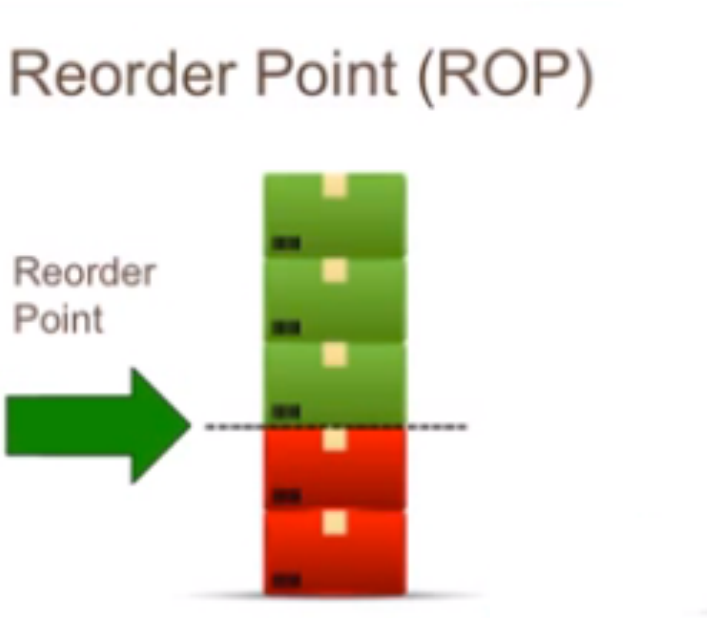
External outputs needed at points of time

- Statements, status reports, bills, reminders

Types of Events – State Events

- an event that occurs when something happens inside the system that triggers some process

Eg. reorder point is reached for inventory item



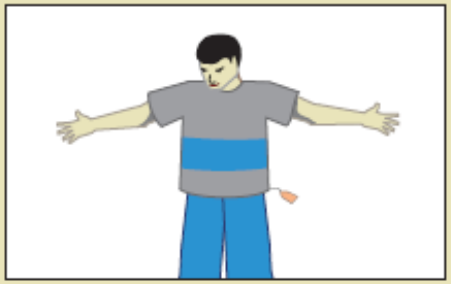
Find the actual event that affects the system

Customer thinks about getting a new shirt

Customer drives to the mall

Customer tries on a shirt at Sears

Customer goes to Walmart

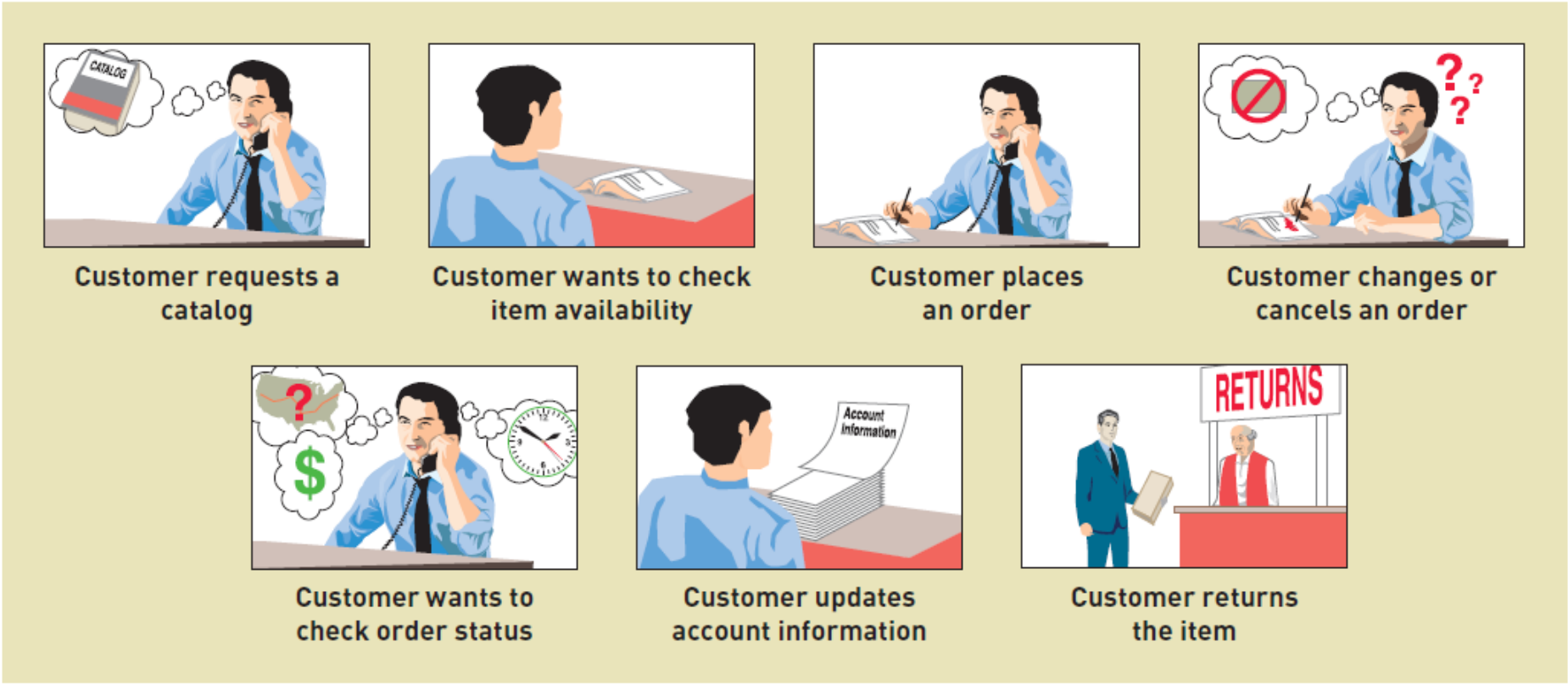


Customer tries on a shirt at Walmart

Customer buys a shirt

(the event that directly affects the system!)

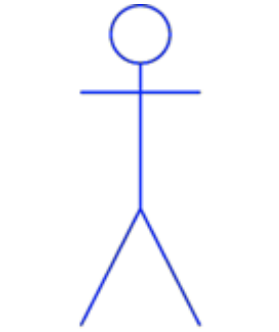
Tracing a sequence of transactions resulting in many events

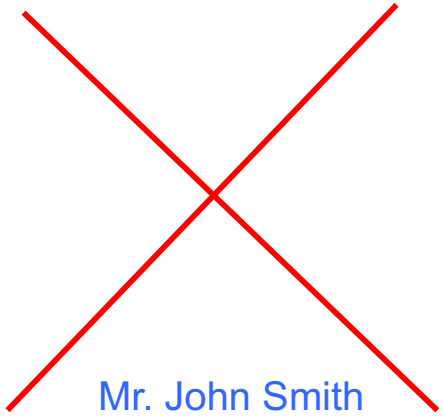


Use Case Diagram Elements

ACTOR

- A person or another software entity (such as a system timer) that initiates the functionality provided by a Use Case
- A coherent set of roles that users of Use Cases play when interacting with Use Cases
  - Roles not users or people
  - User may have more than one role
- Primary vs. Secondary (next slide)





Use Case Diagram Elements

Primary vs. Secondary ACTOR

- **Primary Actors:** The Actor(s) using the system to achieve a goal. The Use Case documents the interactions between the system and the actors to achieve the goal of the primary actor.
- **Secondary Actors:** Actors that the system needs assistance from to achieve the primary actor's goal.
- Example: A bank loan officer wants to review a loan application from a customer, and part of the process involves a real-time credit rating check.
- Use Case Name: Review Loan Application
- Primary Actor: Loan Officer
- Secondary Actors: Credit Rating System

Alistair Cockburn's book "Writing Effective Use Cases"

Use Case Diagram Elements

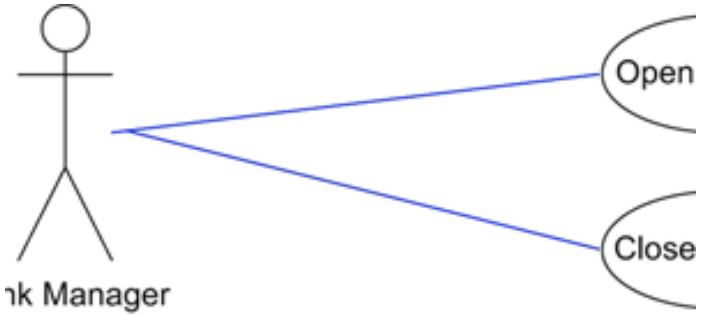
USE CASE

- A Use Case is an interaction between the system and a person or another system to achieve a result
- A required “bit” of functionality
- It yields an observable result of value to an actor
- Typically named with a verb then a noun eg. View Timetable

Use Case Diagram Elements

ASSOCIATION

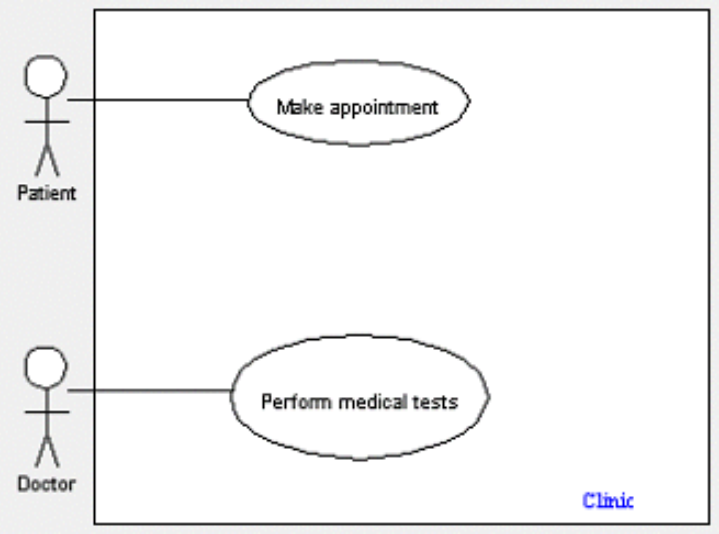
- The **Association** line indicates that a particular Actor makes use of the functionality provided by a particular **Use Case**.



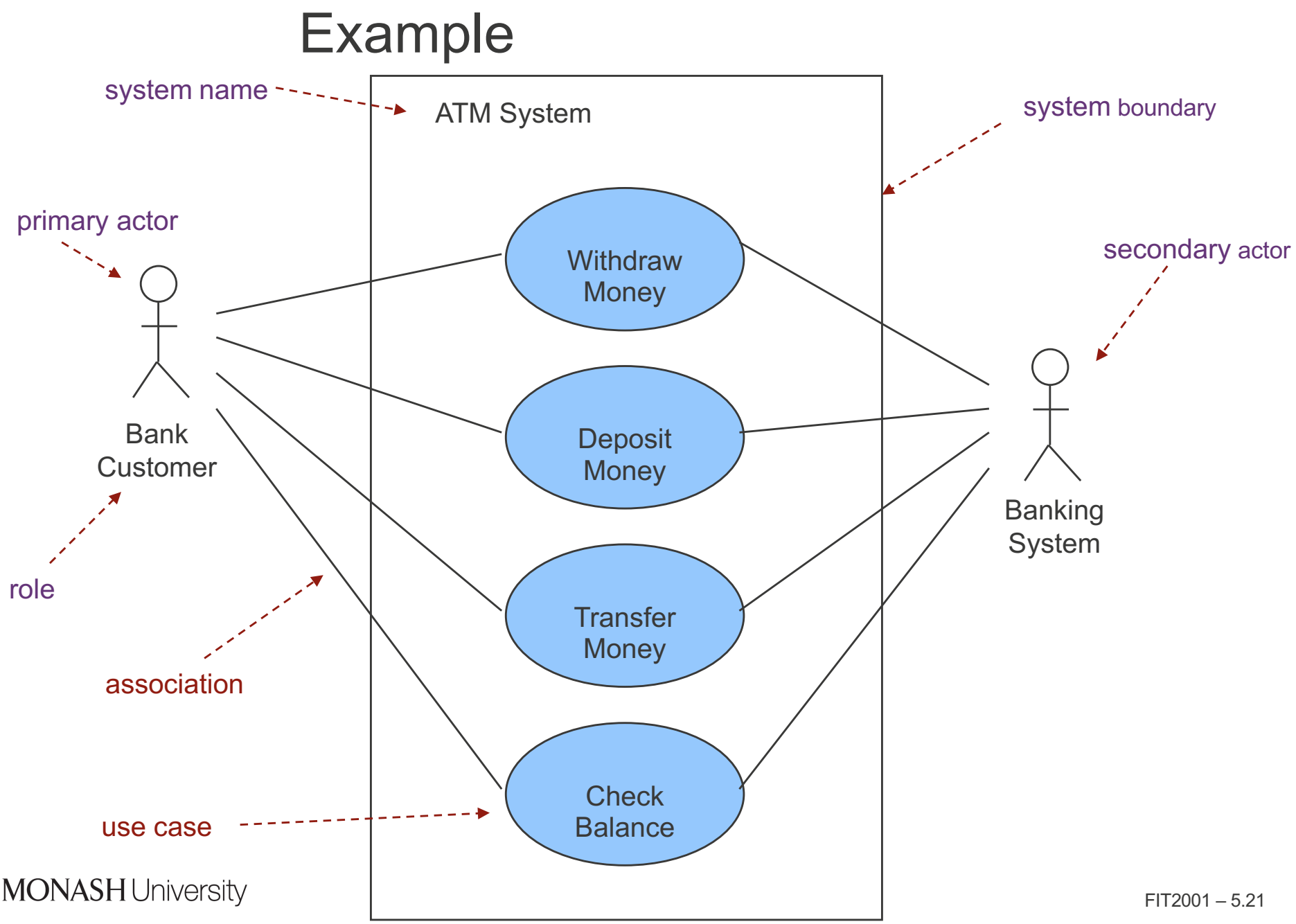
Use Case Diagram Elements

SYSTEM BOUNDARY

- Defines the scope of what the system will be.
- Shown as a rectangle spanning all the use cases.



Use Case Diagram Elements - Example



Use Case Diagram Elements

USE CASE RELATIONSHIPS

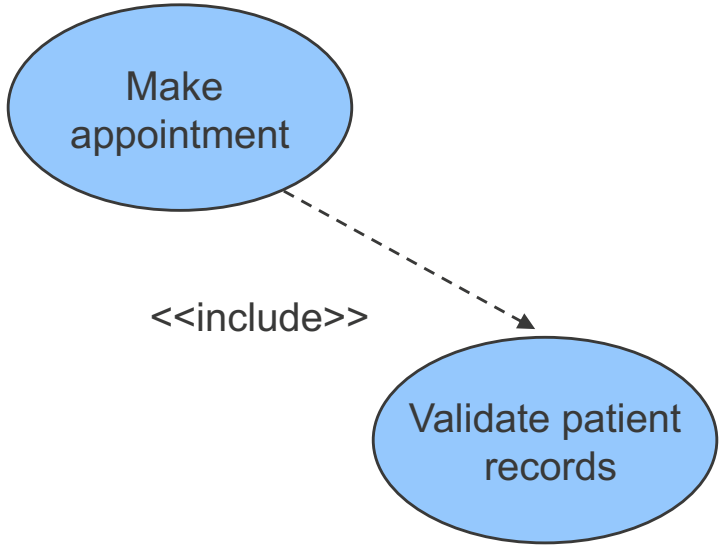
- A relationship between two use cases is basically a dependency between the two use cases.
- They can be:
  - **Include** - models encapsulated behaviors that can be inserted into a use case and possibly reused across multiple use cases.
  - **Extend** - models significant extensions and behaviors that can occur as additions to the use case model.
  - **Generalisation** models conceptual similarity between use cases.

Use Case Diagram Elements

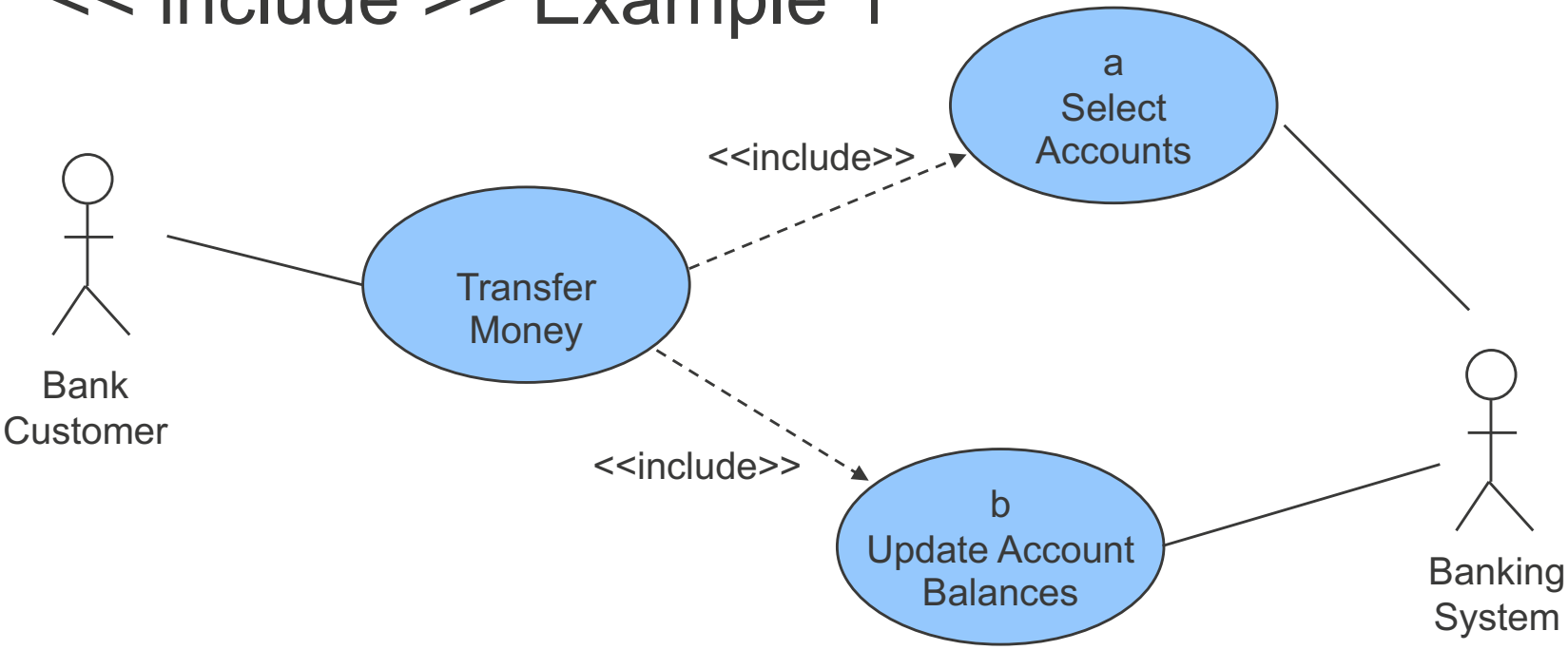
RELATIONSHIP << include >>

- Captures commonality among use cases
- Re-usability helps avoids repetition (saves time and money) – manages redundancy
- An include relationship is depicted with a directed arrow having a dotted shaft.

- Eg. The "Validate patient records" use case is contained within the "Make appointment" use case. Hence, whenever the "Make appointment" use case executes, the business steps defined in the "Validate patient records" use case are also executed.



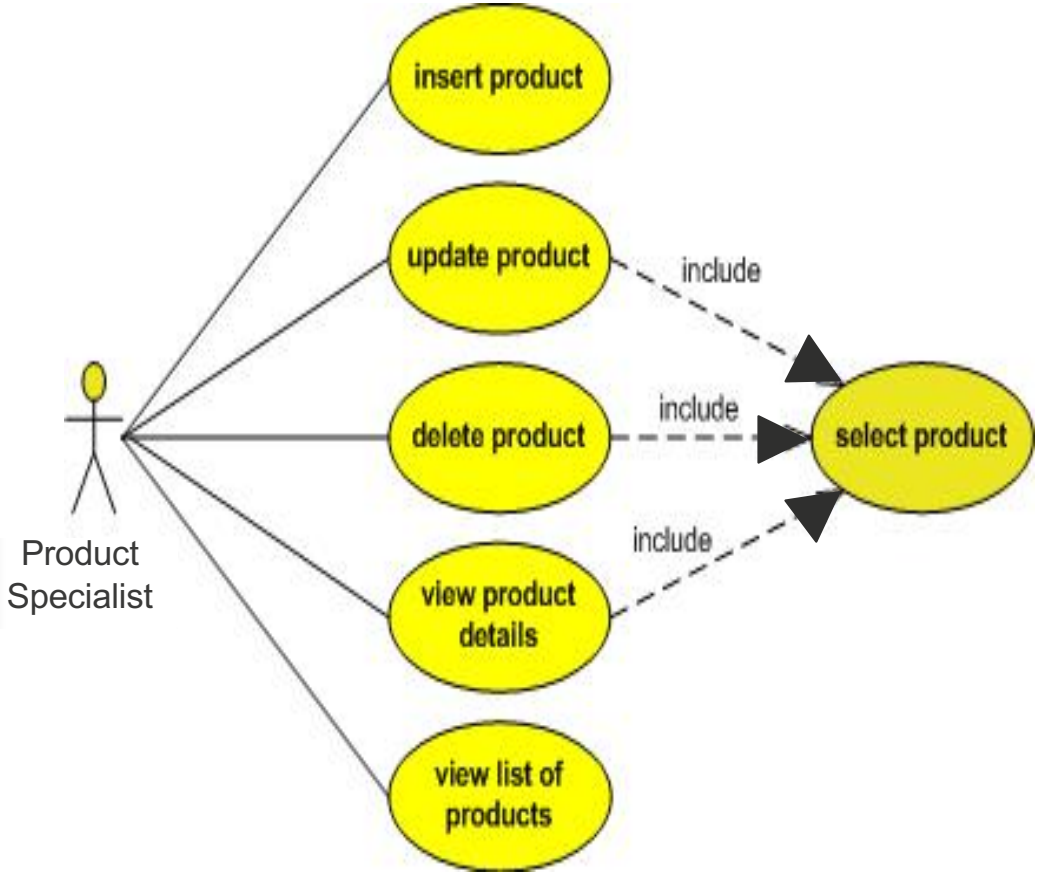
<< include >> Example 1

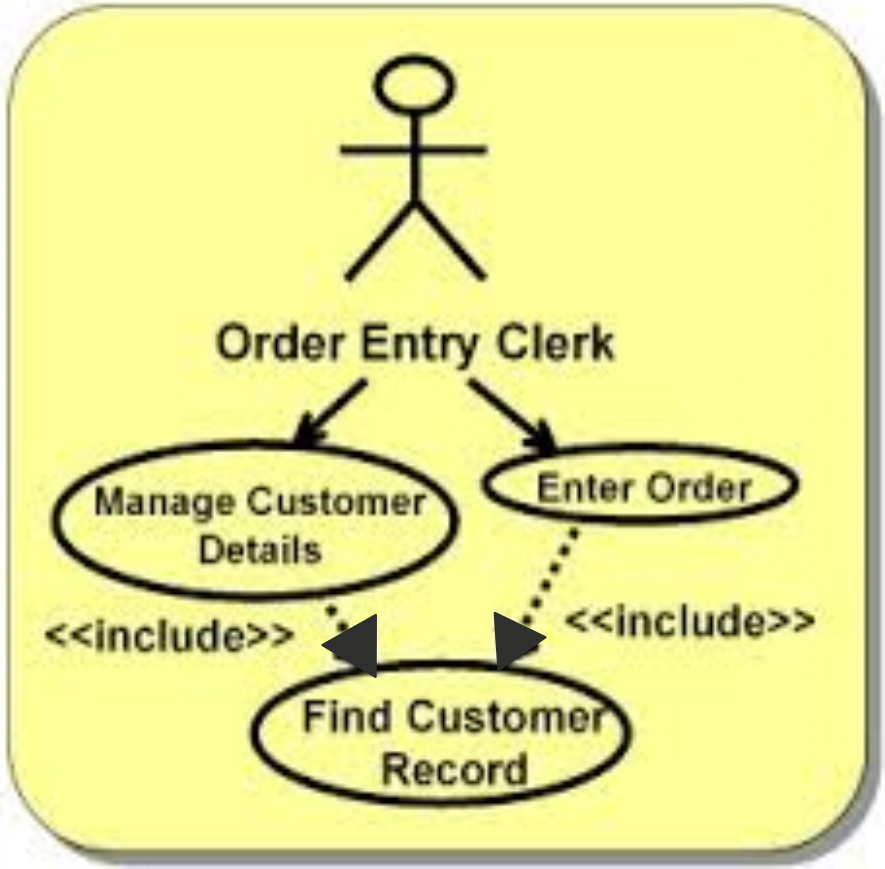


This is an **include dependency**.

It indicates that use case a and b are “included” in the base use case and will be invoked as part of ‘Transfer money’. They have been separated because of they are used by other use cases (reuse).

<< include >> More examples

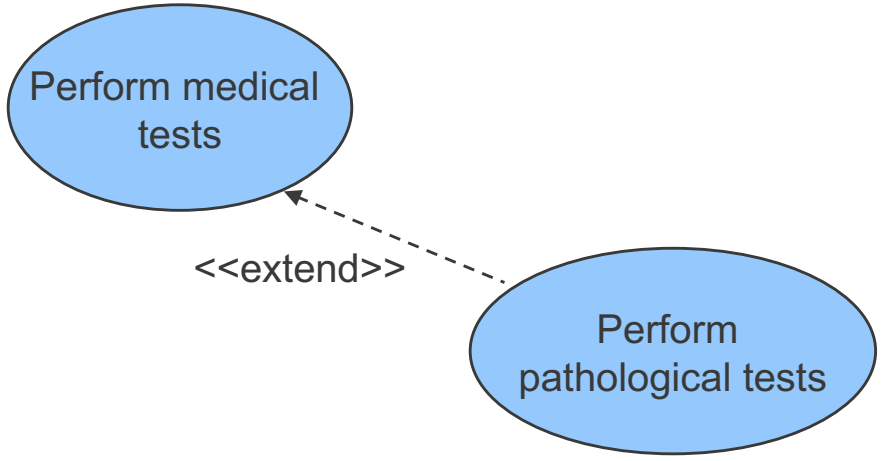




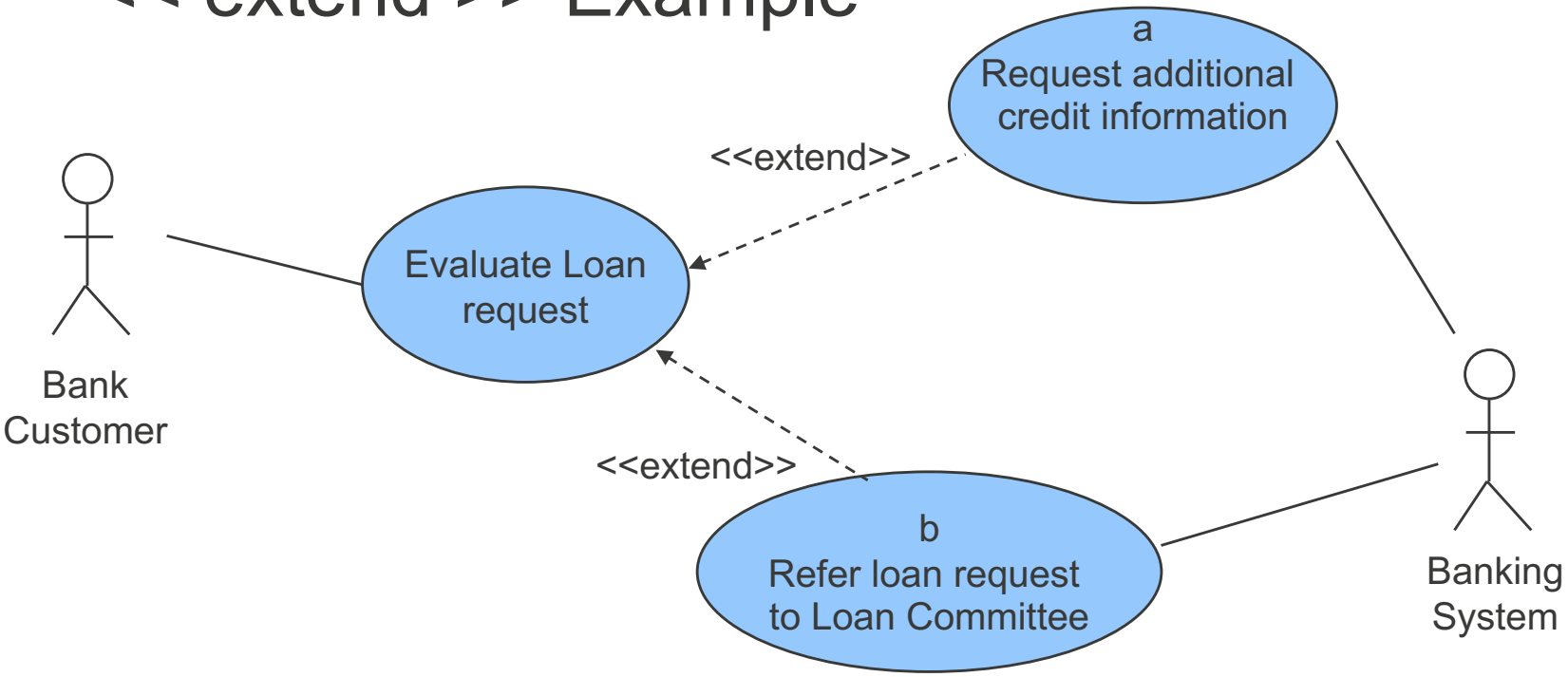
Use Case Diagram Elements

RELATIONSHIP << extend >>

- The child use case adds to the existing functionality and characteristics of the parent use case (optional/exception behaviour)
- The parent case can function without the child case.
- An extend relationship is depicted with a directed arrow having a dotted shaft, similar to the include relationship. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow.
- Eg. The "Perform pathological tests" (child) use case extends the functionality of the "Perform medical tests" (parent) use case. It is used only when required.



<< extend >> Example



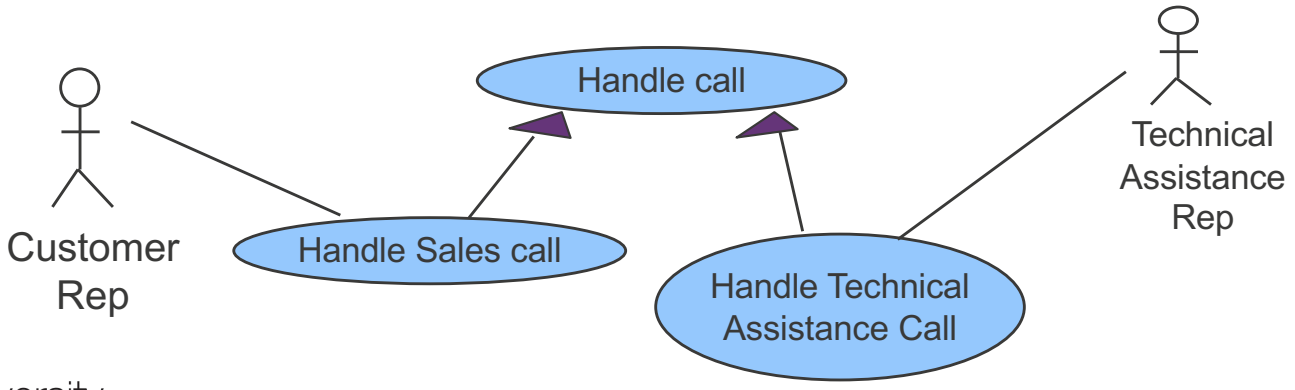
This is an **extend dependency**.

It indicates that use case a and b may or may not be invoked.

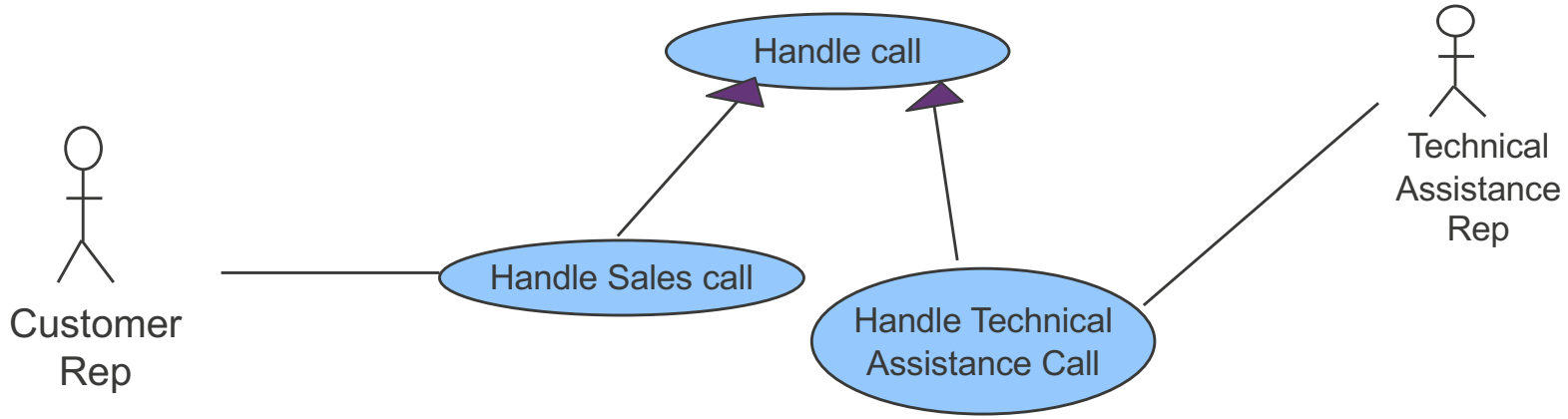
Use Case Diagram Elements

RELATIONSHIP – Generalisation

- If use cases have common behaviour, structure and similarities, their common parts can be factored out into a parent use case to optimise the model – the parent case is often abstract … not ever directly called by an actor.
- A generalization relationship between parent and child use cases is one in which the child is a more specialised form of the parent use case. The child inherits the behaviors of its parents and adds new behaviors, and specializes in behaviors inherited from the parent.
- A generalised use cases is depicted drawing an open, solid arrow from the specialized (child) use cases to the general (parent) use case.



Generalisation Example 1



- The ‘Handle Sale Call’ or ‘Handle Technical assistance call’ substitutes ‘Handle Call’. They inherit the behaviour of ‘Handle Call’ and add their own behaviour. Handle call is an abstract use case.

‘Handle call’ – behaviour inherited by child use cases

1.Transfer call to available representative

2.Mark representative as busy

3.Start record call

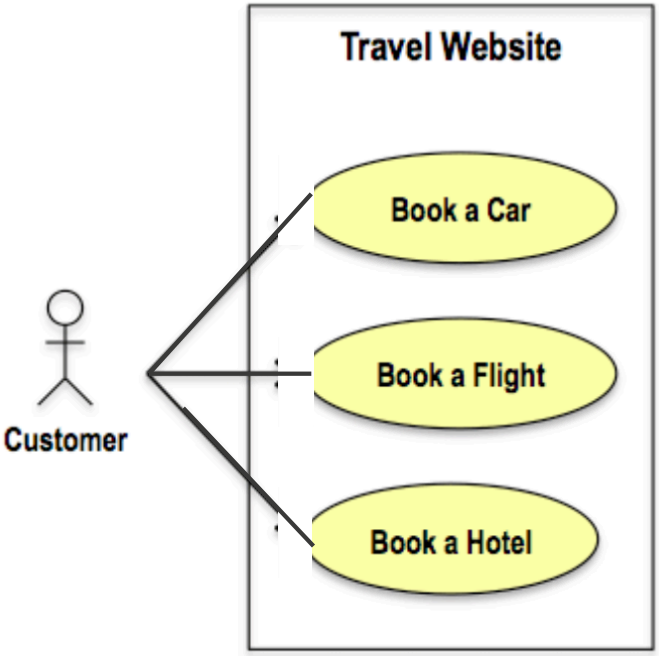
4.Stop record call

5.Log call details

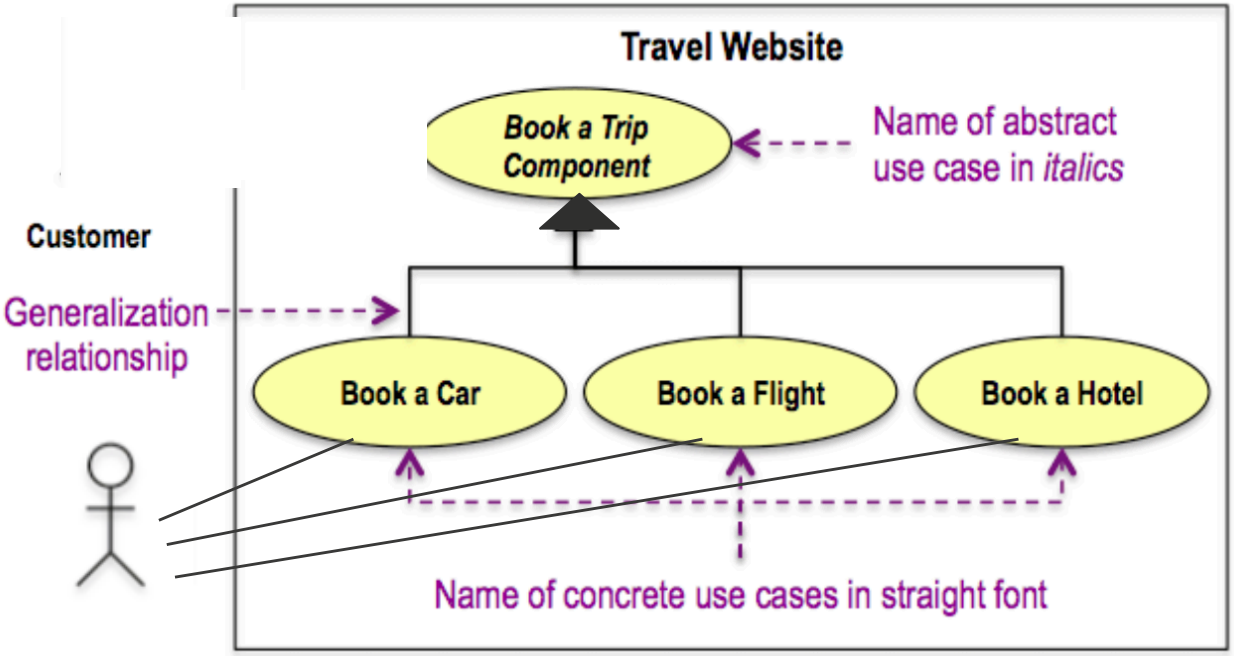
6.Mark representative as free

Generalisation Example 2

Start: Model concrete use cases to represent different actor goals.



Optimize: Generalize the concrete use cases to a generalization use case to represent the core booking process. Generalize the associations to an association on the generalization use case, so the specialization use cases inherit it.



Note: From a Customer's point of view, nothing has changed. Any Customer still sees only the concrete (public) specialization use cases, not the abstract (private) generalization use case. This is about **model optimization** and nothing else.

Ref: http://www.batimes.com/articles/generalization-and-use-case-models-part-2.html

What is a Use Case Description?

"The use case description (narrative, specification) provides the details of the functionality that the system will support and describes how the actors will use the system in order to obtain a specific result of value."

Use Case Template	'Create Order' Use Case
Use Case Name	*Create order*
Use Case Description	*Create order is the ability to request the purchase of a product*
Actor	*Order Creator*
Pre-conditions	• *Order Creator has been identified*
Basic Flow	1. *Order Creator selects 'order product' action* 2. *System requests customer/product identification information* 3. *Order Creator provides customer/product identification information* 4. *System requests mailing information* 5. *Order Creator provides mailing information* 6. *System verifies mailing information* 7. *System requests order be submitted* 8. *Order Creator submits order* 9. *System submits product order for processing* 10. *System confirms product order*
Post-conditions	• *Product order has been created*
Alternate Flows	• *Product is not in stock* • *Product has been discontinued* • *A customer's initial order is over $200*

Use Case Description – Template *

Number	*Unique use case number*	
Name	*Brief noun-verb phrase*	
Summary	*Brief summary of use case major actions*	
Priority	*1-5 (1 = lowest priority, 5 = highest priority)*	
Preconditions	*What needs to be true before use case "executes"*	
Postconditions	*What will be true after the use case successfully "executes"*	
Primary Actor(s)	*Primary actor name(s)*	
Secondary Actor(s)	*Secondary actor name(s)*	
Trigger	*The action that causes this use case to begin*	
Main Scenario	**Step**	**Action**
	Step #	*This is the "main success scenario" or "happy path."*
	…	*Description of steps in successful use case* "execution"
	…	*This should be in a "system-user-system, etc." format.*
Extensions	**Step**	**Branching Action**
	Step #	*Alternative paths that the use case may take*
Open Issues	*Issue #*	*Issues regarding the use case that need resolution*

*Adapted from A. Cockburn, "Basic Use Case Template"

Use Case Description Example

Number	1
Name	Withdraw Money
Summary	User withdraws money from one of his/her accounts
Priority	5
Preconditions	User has logged into ATM
Postconditions	User has withdrawn money and received a receipt
Primary Actor(s)	Bank Customer
Secondary Actor(s)	Customer Accounts Database

Continued …

Use Case Description - Example

Trigger	User has chosen to withdraw money	
Main Scenario	**Step**	**Action**
	1	System displays account types
	2	User chooses account type
	3	System asks for amount to withdraw
	4	User enters amount
	5	System debits user's account and dispenses money
	6	User removes money
	7	System prints and dispenses receipt
	8	User removes receipt
	9	System displays closing message and dispenses user's ATM card
	11	User removes card
	10	System displays welcome message
Extensions	**Step**	**Branching Action**
	5a	System notifies user that account funds are insufficient
	5b	System gives current account balance
	5c	System exits option
Open Issues	1	Should the system ask if the user wants to see the balance?

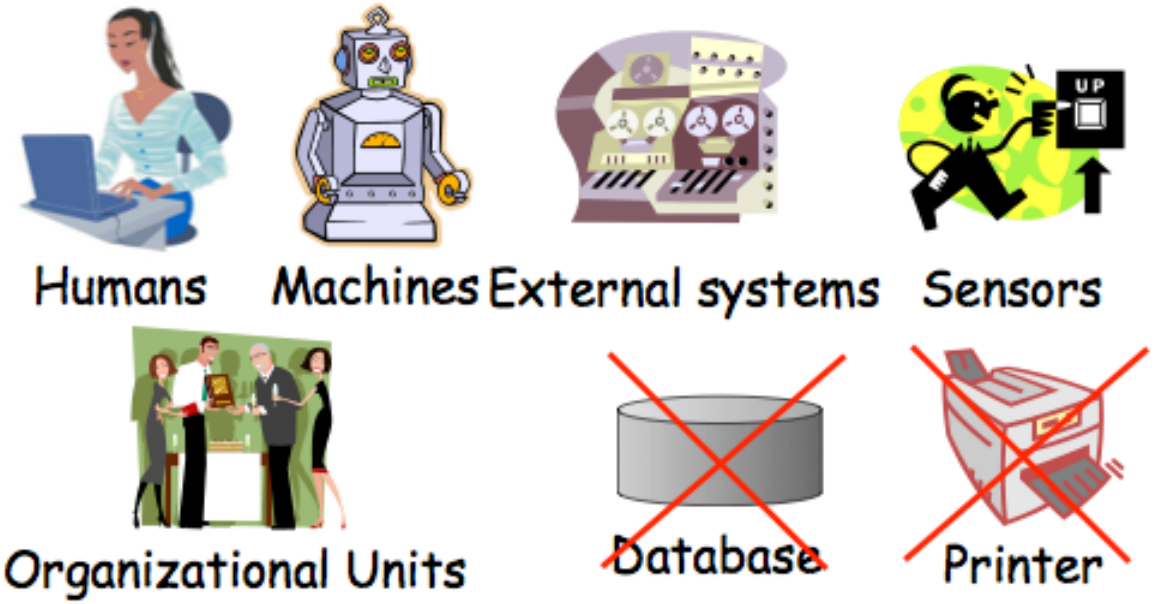
Writing Use Case Descriptions

Don’t focus on perfection – be productive

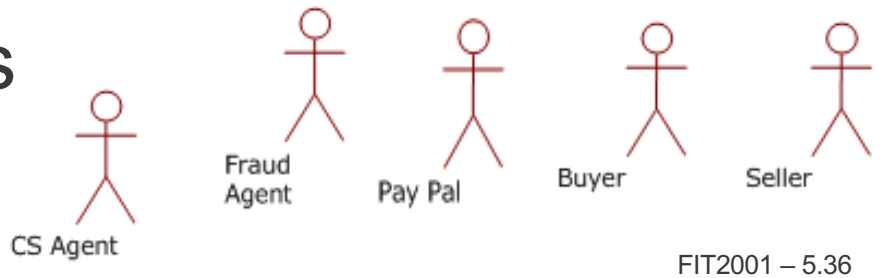
- Not about getting it right the first time
- An iterative process where you work and refine.

1. Identify the Actors

- Any "object" or person that has behavior associated with it. Users and Systems.



- Eg. Who are the actors that interact with ebay



2. Identify the Goal

- From the high-level scenario
- By brainstorming
- By asking

"What does this Actor want to do?"
"What will the actor use the system for?"
"Will the actor create, store, change, remove or read information in the system?"

3. Define the Pre-conditions

- Something that must happen before the Use Case can start; something that must be in place before the Use Case can start
- Identify Pre-Conditions by asking:
  "What must be in place for the Use Case to begin?" "How do you know you need to do this?"
- Examples:
  User account exists, User has enough money in their account, There is enough disk space

4. Define the Post-conditions

- The result, or successful outcome of the Use Case
- Identify the Post-Conditions by asking: "What is the successful result of this process or Use Case?
- Examples:
  Money was transferred to the user account,
  User is logged in, The file is saved

5. Describe the 'Main Flow'.1

- Primary Scenario / Happy Day Scenario
  - The simplest sequence – **everything goes right**
  - Starts with pre-conditions and ends with post conditions
- Describe the Main Flow by asking:
  - "What must happen to achieve the goal / outcome?" "What does the actor need to do next?" "What might happen next?" "What do you need to do to get from the trigger to the outcome?"

5. Describe the 'Main Flow'.2

- Example:
  1. Admin enters Course name, code & description
  2. System validates Course code
  3. System adds course to the database and shows confirmation message

- Identify opportunities for reuse

5. Describe the 'Main Flow'.3 *Guidelines for effective writing*

- Only one side (actor or system) should do something in a single step
- Each step should lead to some progress
  - *NOT 'User clicks Return key'*
- Use simple sentences
  - Actor asks for money
  - System asks for amount
  - Actor gives amount
  - System gives the money
  - *NOT Get the amount from the user and give him the money*

6. Describe the Alternate Flows.1

- Describes the variations/exceptions to the Main Flow
  - An exception or error flow to any line item in your basic flow
  - An additional flow, not necessarily error based, but a flow that COULD happen
- Discover alternate flows by asking:

"What might affect this Use Case?

"What could go wrong?"

Listen for "sometimes", "maybe", and "it depends"

6. Describe the Alternate Flows.2

- Examples:
  - While a customer places an order, their credit card failed
  - While a customer places an order, their user session times out
  - While a customer uses an ATM machine, the machine runs out of receipts and needs to warn the customer

How do you know when you have identified all your Use Cases?

- When all actors are specified
- When every functional requirement has a use case which satisfies it
- The CRUD (Create, Read, Update, Delete) technique can help validate, refine or cross-check use cases

Use Cases and CRUD Technique

For each type of data (data entity or domain class), verify that a use case has been identified that creates a new instance, updates existing instances, reads or reports values of instances, and deletes (archives) an instance.

Data entity/domain class	CRUD	Verified use case
Customer	Create	Create customer account
	Read/report	Look up customer Produce customer usage report
	Update	Process account adjustment Update customer account
	Delete	Update customer account (to archive)

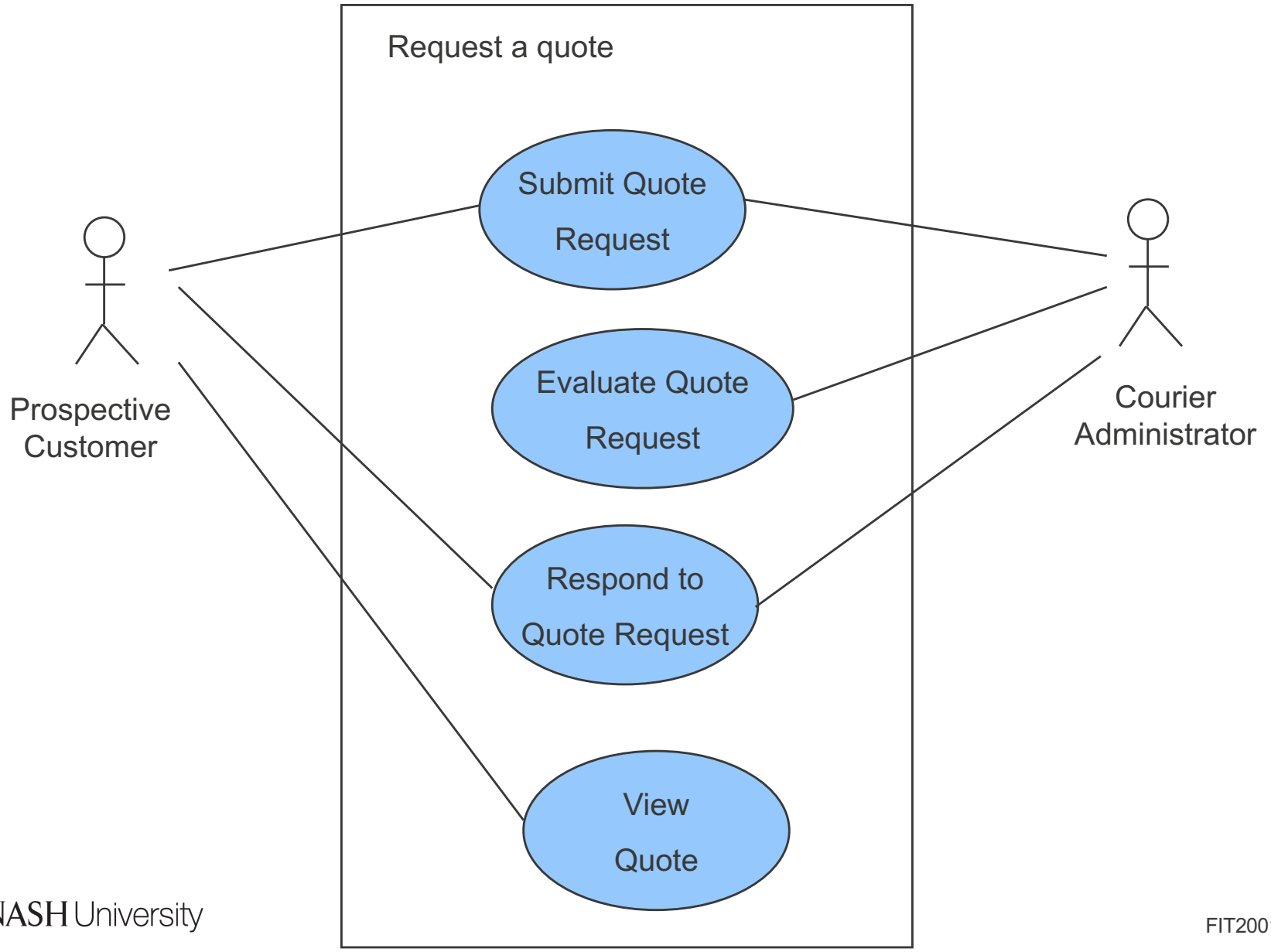
ON THE SPOT COURIER SERVICES

Request a Quote

We would like the customer to enter in the pick up and delivery address, the package details and the date and time of pickup and delivery. If the package does not meet our size requirements or the we are unable to do the pick up or delivery on the specified dates, we let them know that we cannot provide a quote, otherwise we provide a quote.

Create a Use Cases for the Request a Quote function for On the Spot Courier Services

Use Case modelling - Example



Workshop Preparation

- Watch Seminar 5

Thanks for watching

See you next week

Resources:

Prescribed text:

– Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 3 (pp. 73-88)

Use Case resources:

– Getting started with Use Case Modelling:

http://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

Additional example to help you …

Use case Example description

- ***Purpose:*** *Describe major services (functionality) provided by a hospital's reception.*
- **Hospital Reception** subsystem or module supports some of the many job duties of hospital receptionist. Receptionist schedules patient's appointments and admission to the hospital, collects information from patient upon patient's arrival and/or by phone. For the patient that will stay in the hospital ("inpatient") she or he should have a bed allotted in a ward. Receptionists might also receive patient's payments, record them in a database and provide receipts, file insurance claims and medical reports.

Use case diagram example: