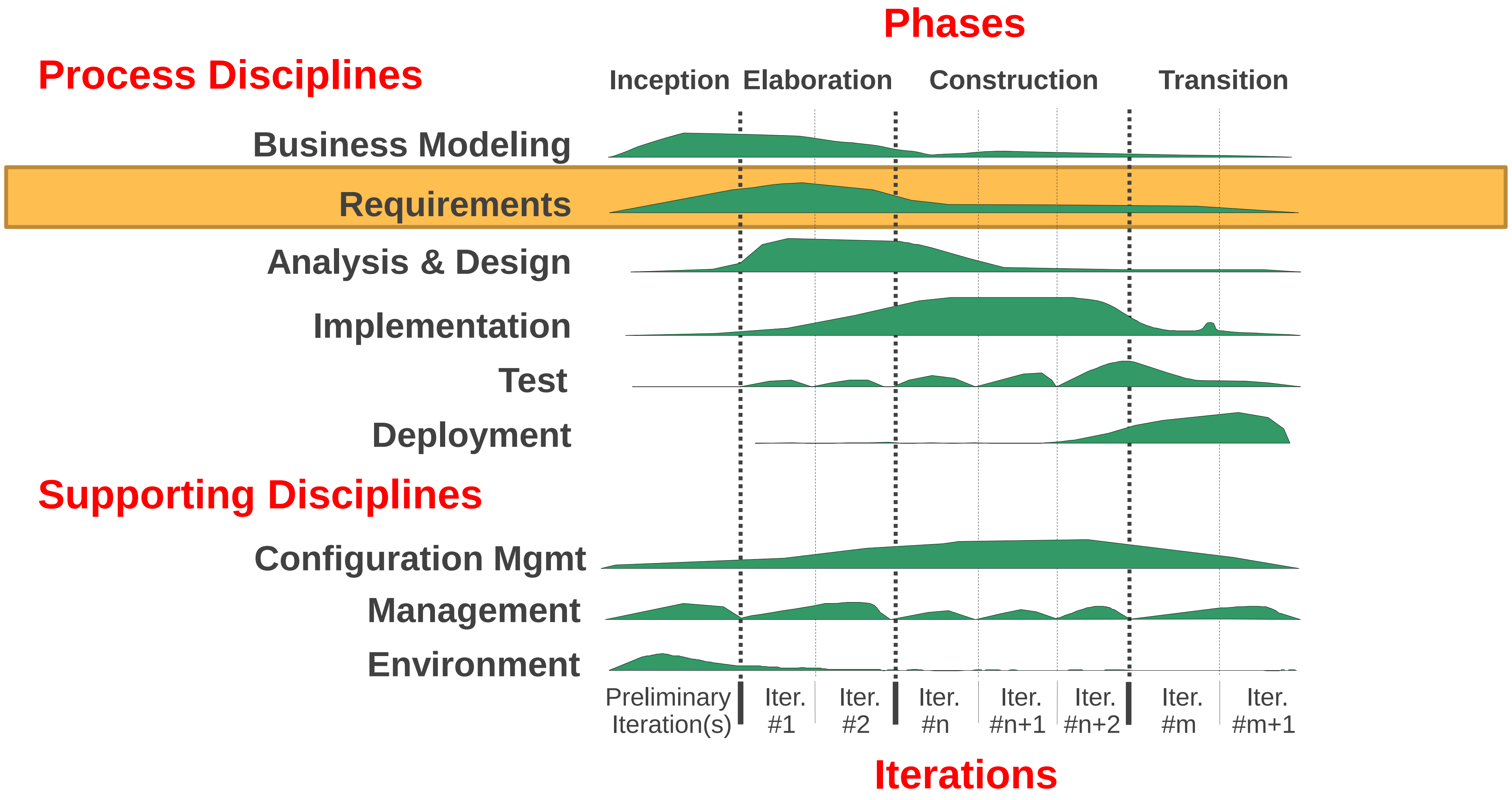
Bring ideas to life
VIA University College

Requirements

SWE 1

Unified Process



What are Requirements?

Requirements comes after Project Description and before Domain Model!!

Capabilities and **Conditions** to which **the system**, and the project, **must conform!**

- Only focus on **WHAT** the system must do at the end
- Absolutely **not HOW** it is done!!!! – The customer doesn't care!!

You will properly find it difficult to distinguish between *what* and *how*

How is a matter of **design** – not specification

How to find the right requirements?

- Writing Use Cases with customers
- Requirement workshops with both developers and customers
- Focus groups with proxy customers
- Demo of results of each iteration to the customers

The interaction with the customer is essential!!

Types and Categories of Requirements FURPS+

FURPS+ can be used to categorise requirements and work as a checklist

- **Functionality** (features, capabilities, security etc.) These are the functional requirements!
- **Usability** (human factors, help system, documentation)
- **Reliability** (frequency of failure, recoverability, predictability)
- **Performance** (response times, throughput, accuracy, availability, resource usage)
- **Supportability** (adaptability, maintainability, internationalisation, configurability)

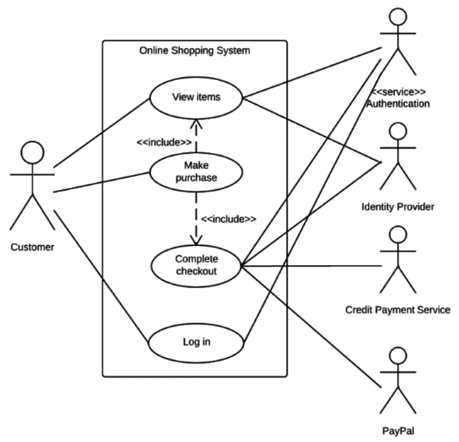
Quality requirements

the **'+'** is sub factors like

- **Implementation** (resource limitation, languages and tools, hardware ...)
- **Interface** (constraints because of interfacing with external systems)
- **Operations** (system management in its operational setting)
- **Legal stuff** (licensing etc.)
- Etc.

Have you remembered to look at all these??

Use Cases - Definitions



Actor

- A relevant **role** that a person, a system or an organisation can play when interacting with the system
- Will never be part of the system delivery

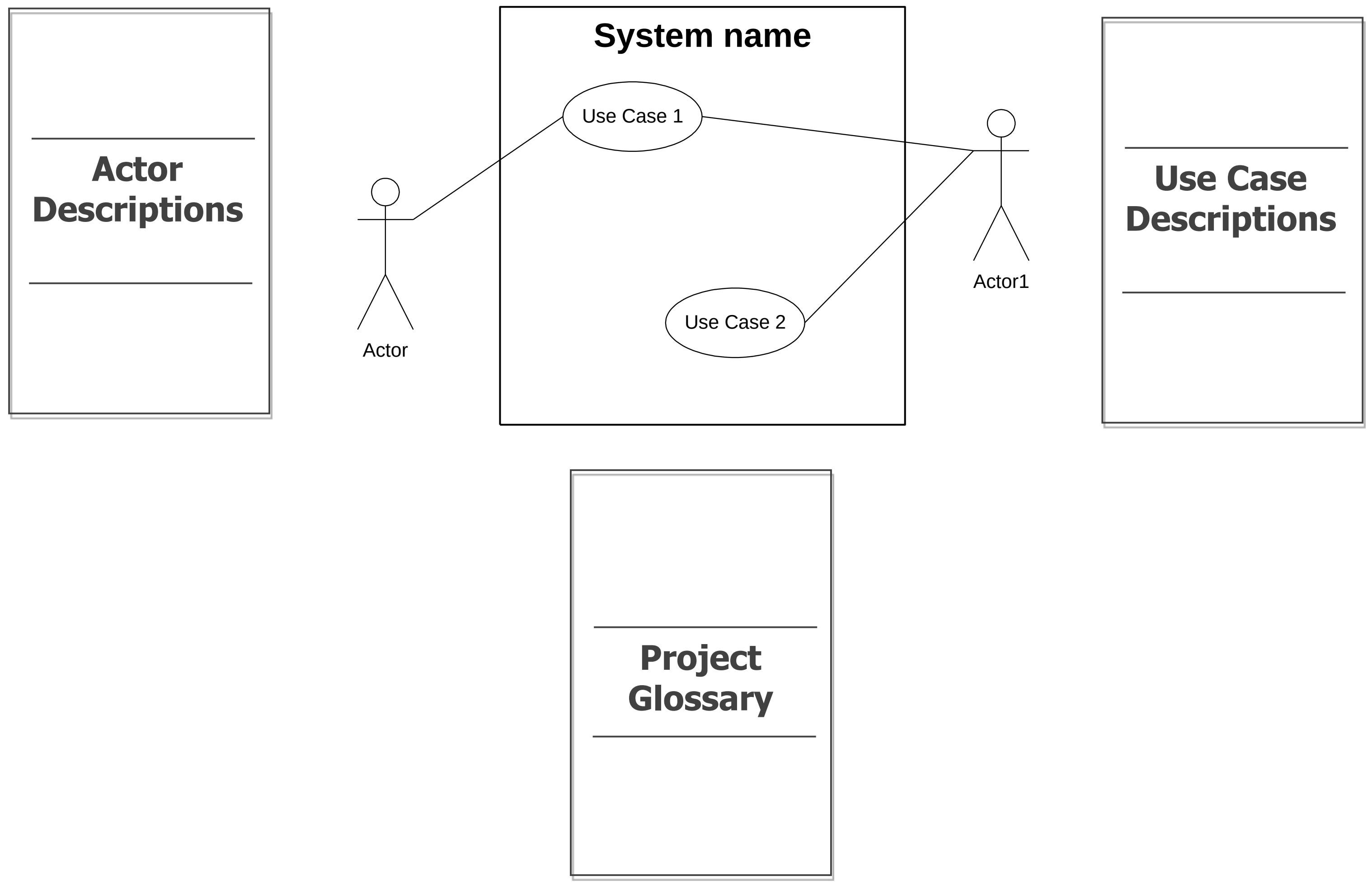
Use Case

- A collection of related scenarios that describe how actors use the system
- Both success and failure scenarios must be described!

Scenario

- A specific sequence of actions/reactions and interactions between actors and the system
- Describes one single story using the system, or one path through the Use Case

Use Case model elements



What is Actors?

Anything with a behaviour relevant for the system

Three types of Actors

1. **Primary Actor**
   - Has user goals fulfilled using services of the System
   - Important to find to find the user goals they have

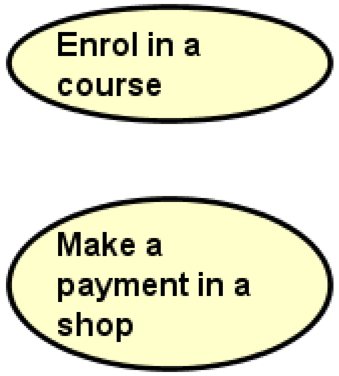
2. **Supporting Actors**
   - Provides a service or information needed to fulfil a user goal
   - Important to find for clarifying external interfaces and protocols

3. **Offstage Actors**
   - Has an interest in the behaviour of the Use Case – are not primary or supporting
   - Eg. Government, tax etc.

Use Case

A Use Case is a ***set of scenarios*** tied together by a ***common user goal***
Examples of a common user goals:

1. "Enrol in in a course"
2. "Make a payment in a shop"



Each user goals encapsulates a set of actions/reactions executed in a defined order!

Use Case – Brief format

Use cases are **informal** stories written in **one paragraph** – normally the ***sunny scenario***

Ex: Enrol in a course:

> "A student arrives at the Study Administration with a list of course names he/she wants to be enrolled in. The Study secretary uses the administrative system to check if the student has the right prerequisite for following each course. If the prerequisites is OK then the secretary use the administrative system to enrol the student in the course. The system validates and records the data. The student receives a list of all the courses that the student now are enrolled in."

This is how Use Case descriptions normally start out

Use Case – Casual format

Use cases **informal paragraphs**, that cover **various scenarios – including abnormally the scenarios**

- Describe all the scenarios where something goes wrong
- Describe how the system reacts to these faulty scenarios seen from the actor

Ex: Enrol in a course:

- What happens if the student do not have the prerequisite to follow a course
- What happens if a student already have attended the course

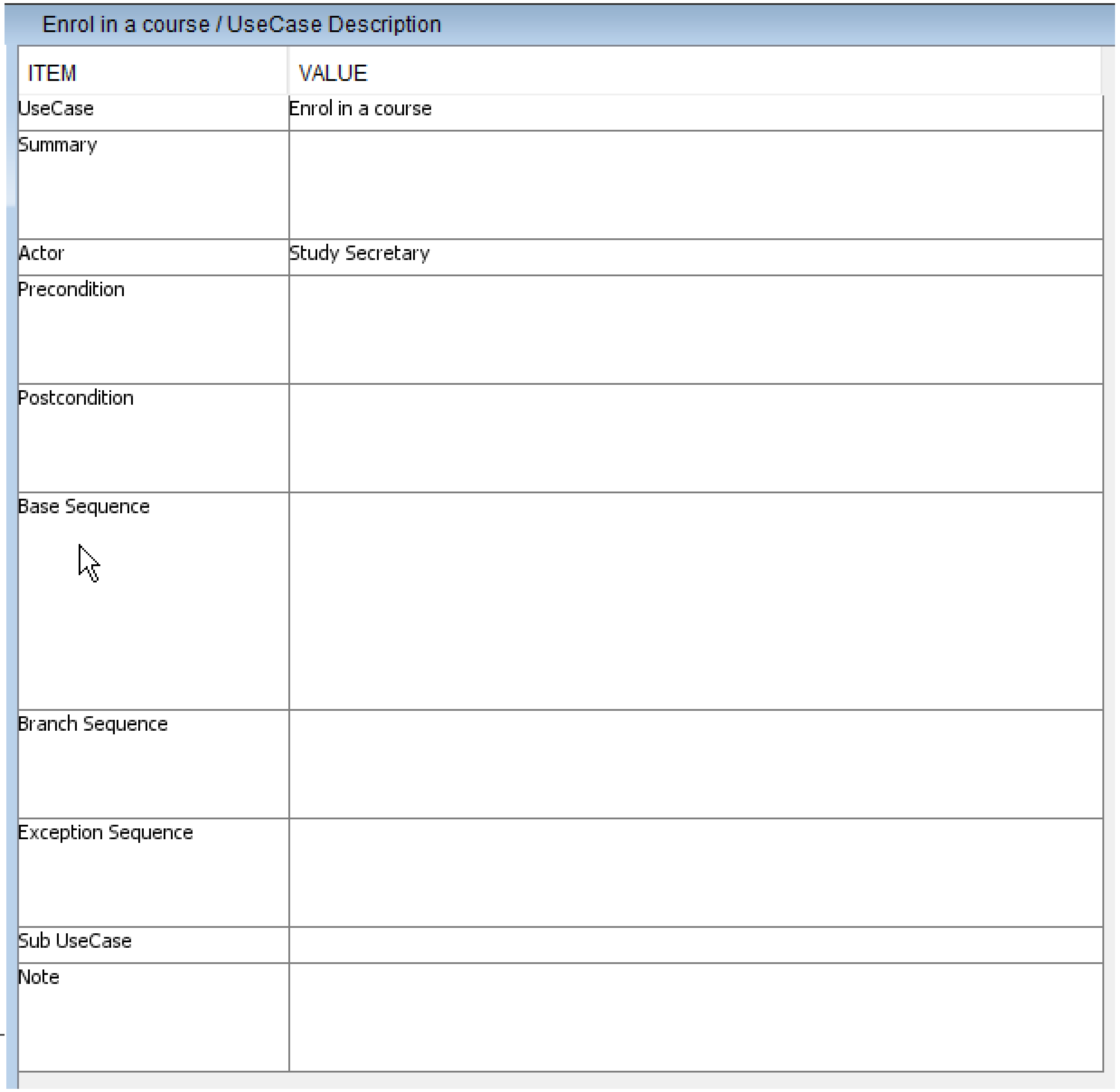
Use Case – Fully Dressed

All steps and variations (exceptions/sub-flows) are described in details

- The format is typically a table with supporting sections
  - Pre/Post-conditions
  - Actors involved
  - Base/Main sequences
  - Branch/Exception sequences
  - Etc.

Example from Astah ->

See example from **[Larman, 2005]** next slide



Enrol in a course / UseCase Description

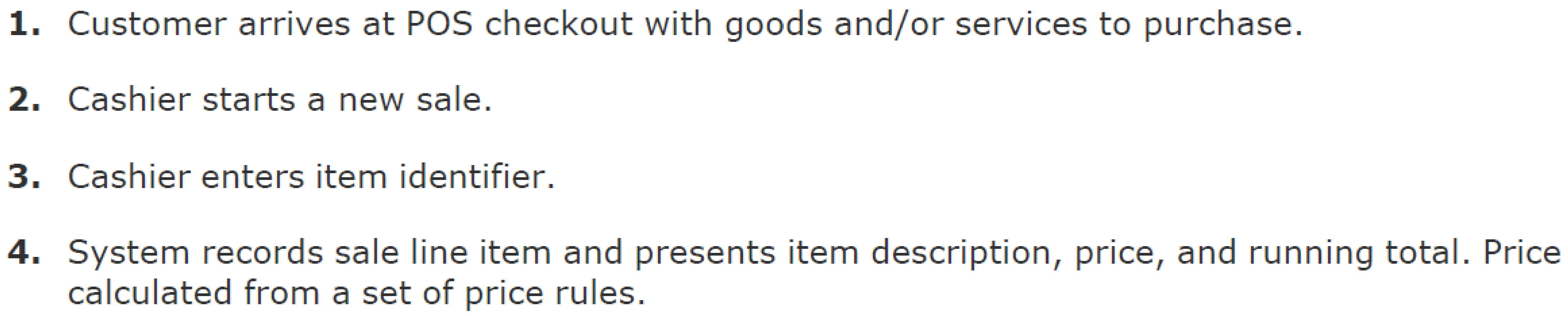
ITEM	VALUE
UseCase	Enrol in a course
Summary	
Actor	Study Secretary
Precondition	
Postcondition	
Base Sequence	
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Use Case Section	**Purpose/Guidelines**
Use Case Name	Must be unique – start with a verb
Scope	The system under design (is often obvious)
Level	User goal or subsystem
Primary Actor	Who is necessary to start the Use Case
Stakeholders and Interests	Who cares about this Use Case, and what do they want?
Pre-conditions	What must be true for the Use Case to start (**and worth telling the reader**)
Success Guarantee/Post-conditions	What must be true on successful completion (**and worth telling the reader**)
Main Success Scenario	A typical, *"sunny scenario"* of the Use Case
Extensions	Alternative scenarios of success or failure
Special Requirements	Related non-functional requirements
Technology and Data variations list	Varying I/O methods and data formats
Frequency of Occurrence	Influence implementation, timing and testing
Misc	Open issues etc.

See example of usage [Larman, 2005] p.68

At least two ways for writing Main Success Scenario and Extensions

One column format



1. Customer arrives at 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 calculated from a set of price rules.

Two column format

Actor Action	System Responsibility
1. Cashier starts a new sale	
2. Cashier enters item identifier	3. Records each sale line item and presents item description and running total

How to find the Use Cases

1. Identify all currently relevant Use Cases at a very high level (Low precision, high accuracy)
   - Brief format

2. Work out the details (add precision)
   - Fully dressed format

Notes about Use Cases

Keep the user interface out of Use Case descriptions in the beginning

Focus on the Actors intent!!

Write Black-box Use Cases

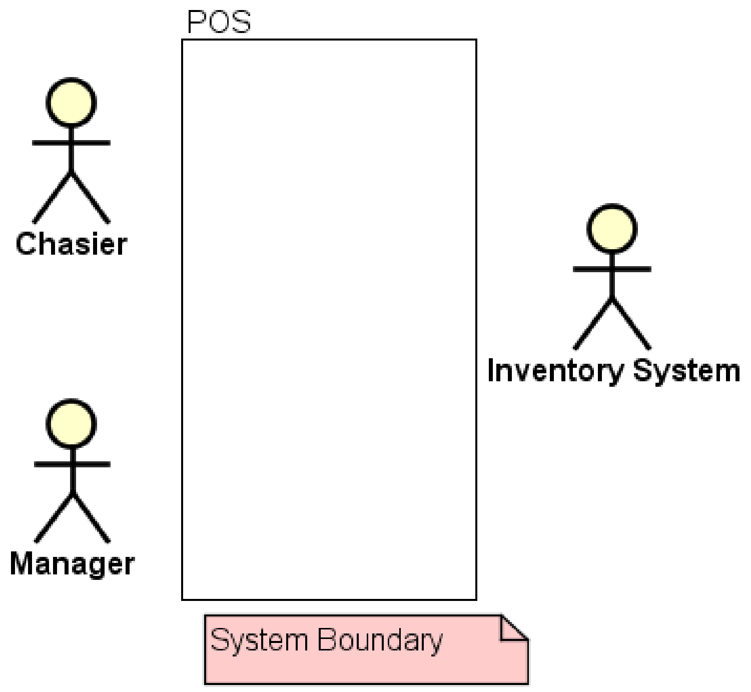
Do not describe how the system works internally

What suits one Actor might not suit the next

Guidelines for finding Use Cases

1. Find the system boundary
   - Everything inside the boundary is the system
   - Everything outside the boundary is NOT part of the system
   - Actors are always outside the boundary

2. Find Primary Actors and Actor Goals
   - Who starts and stops the system?
   - Who does user management?
   - Who does system administration?
   - Are there a **time** Actor?



What about the Customer?

Guidelines for finding Use Cases

Ask customer: “What are your goals when using the system?”

instead of

“What do you do with the system?”

Guidelines for finding Use Cases

Is it the right Use Cases we have found? Use these simple tests:

- The Boss Test
  - If the boss ask what you have been doing all day, then the answer must be meaningful
  - “Log-in” is not meaningful, so a login Use Case do not parse this test

- The Elementary Business Process (EBP) Test
  - EBP Definition: A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leave data in a consistent state. E.g. Approve Credit

- The Size Test
  - A Use Case is not a single action, it is typically many steps and will often require 3+ pages in a fully dressed description
  - These examples are failing the Size Test: Enter item ID, Log-in etc.

Activity Diagrams and Use Cases

Use Activity Diagrams to show what is happening between Actors and Use Cases
- Mainly used when it is difficult to describe the details clearly in the Use Case description

What is a good Specification

- Is unambiguous/evident
- Is complete
- Is consistent
- Is correct
- Tells **what** but **not how**
- Can be understood of the end-user/customer
- Can be tested
- Changeable
- Traceable

References

Writing Effective Use Cases: http://alistair.cockburn.us/get/2465

Exercise

Find Actors and Use Cases for your SEP1 project (or similar small project)

Draw a Use Case diagram with relevant Actors and Use Cases

Write Use Case descriptions for the Use Cases

- Write fully dressed descriptions for the Use Cases that gives most value to the involved Actors and brief descriptions for rest