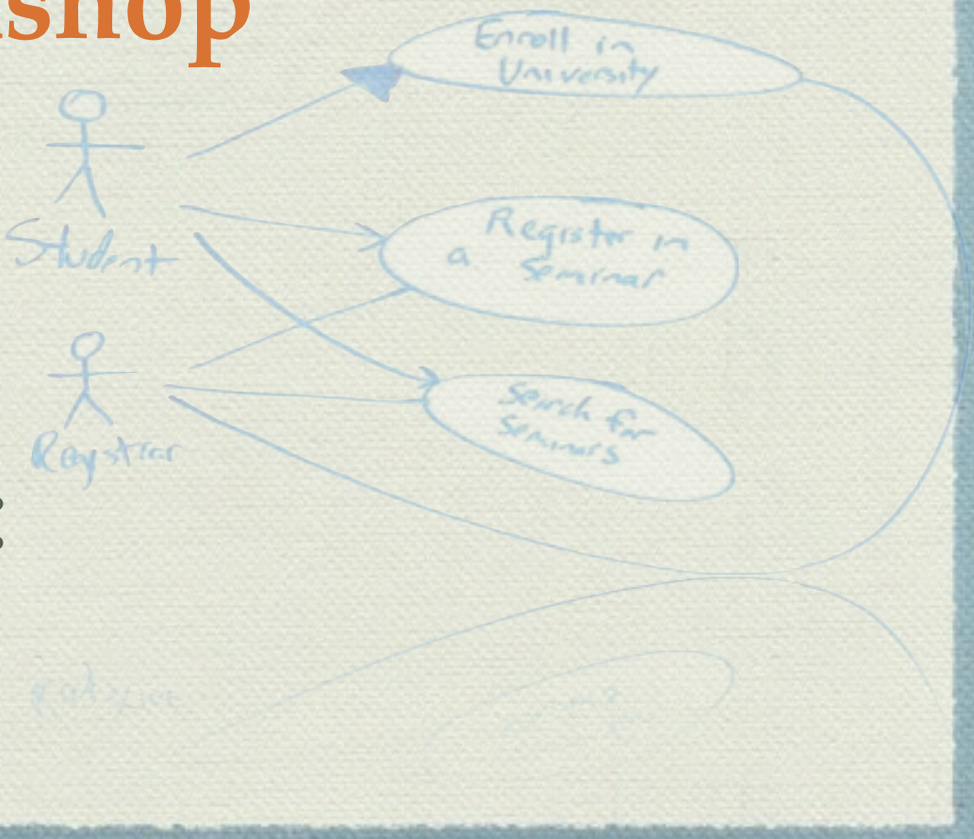
Software Engineering Workshop



Workshop 2

Working with Requirements: Use Case Diagrams

Slides prepared by Marwah Alaofi

- Anyone isn't assigned to a group?

-

- Have not contacted your supervisors?!

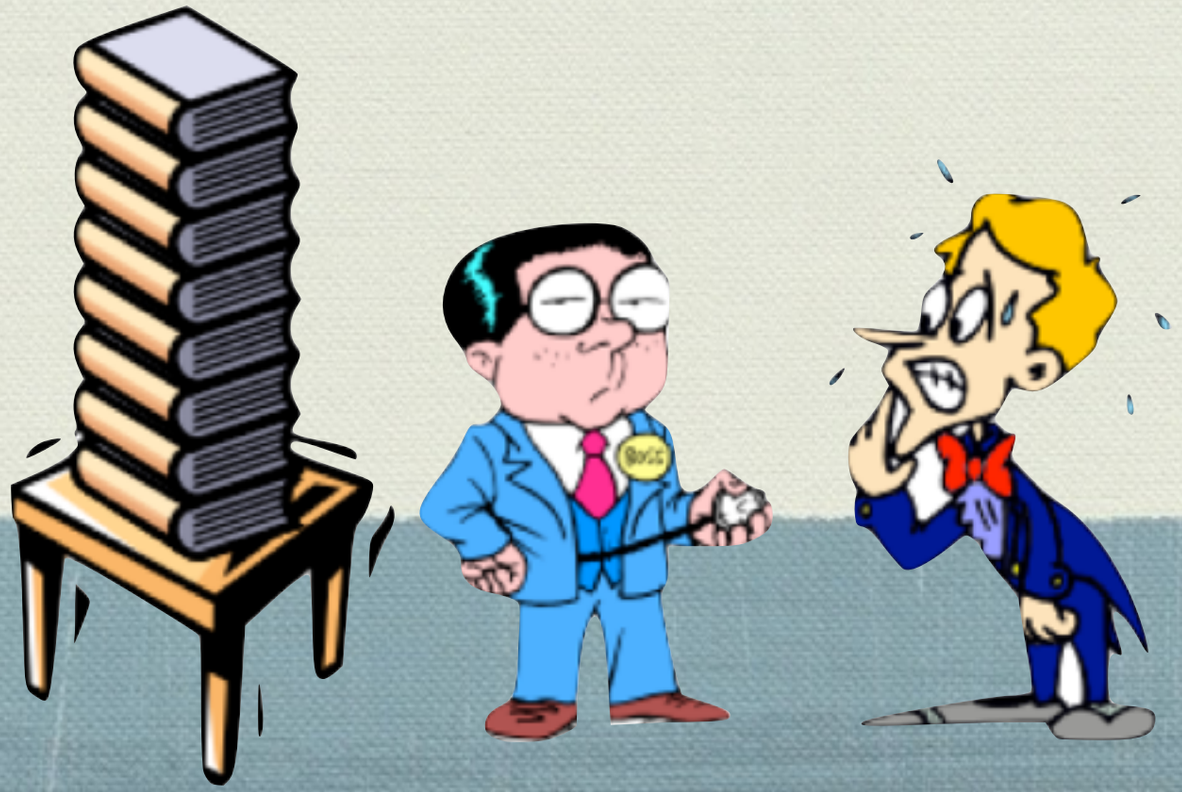
Quick Review

- What's the UML?
- Why do we need it?
- Why do we have many diagrams?

http://www.seas.harvard.edu/news-events/publications/qa/fred-brooks-jr

"The hardest single part of building a software system is deciding what to build. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later." **Frederick Brooks says.**

"Get your team up to speed on these requirements so that you can all start designing the system."



What's next?!

- How do you take this huge set of loosely defined requirements and distill it into a format for your designers without losing important detail?
- Use cases are an excellent starting point!

In today's workshop you'll learn..

- The role of the use case diagram and where to use it in the software development process.
- Different graphical notations used with use case diagram.
- The usefulness of use case diagrams.

Use Case Diagram

- Provides a complete, black-box, outside-in view of system functionality.

- Shows all users of the IT system and all tasks that users can perform with the system.

- Usually applied during requirements activities to capture functional requirements. Nonfunctional requirement?!

- Should be the first serious output from your model after a project is started. *How could you begin to design a system if you don't know what it will be required to do?!*

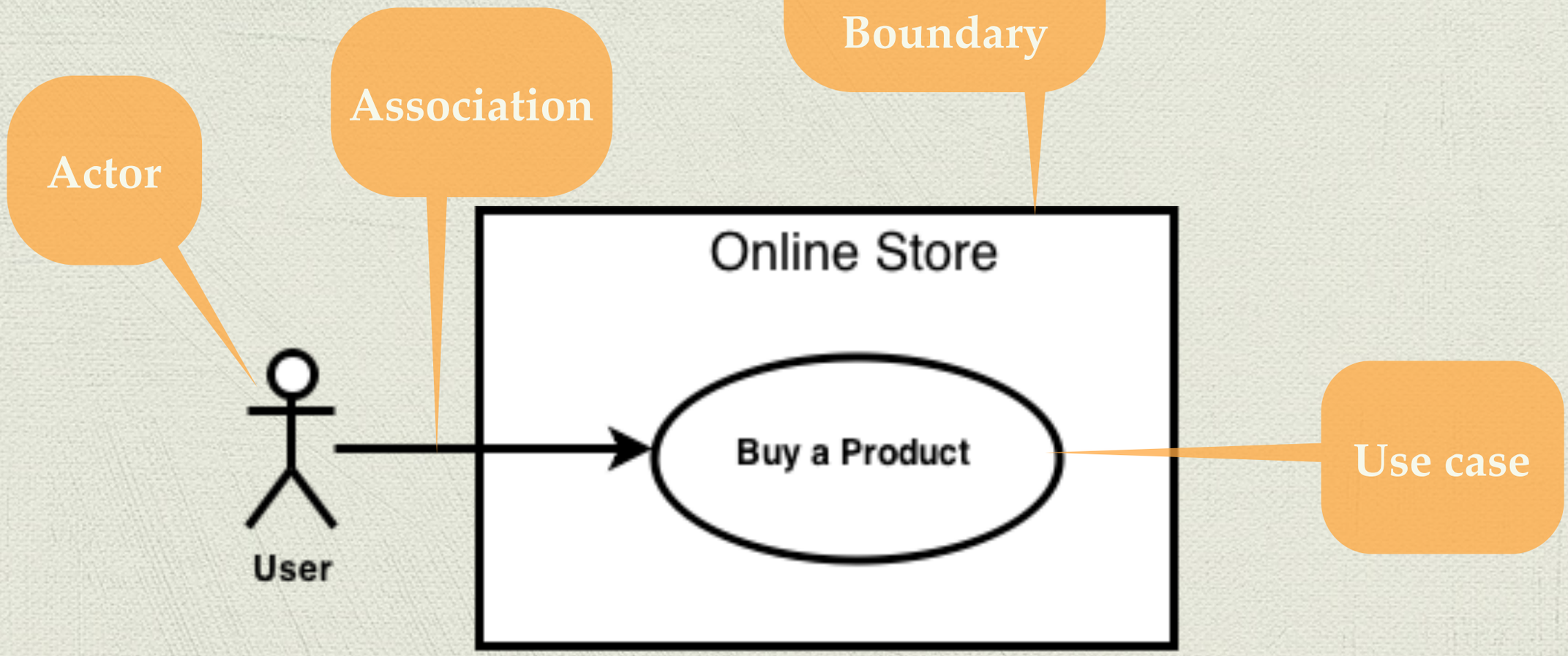
Use Case Diagram (Cont.)

- Describes the system functionality.
- Describes what actors that interact with the system.
- Describes any associations between use cases, actors, use cases and actors.
- Answers the question of “What does the system/software do? And who/what interacts with it?”

The graphical notations of the use case diagram

Online Store:

Requirement: The online store shall allow customers to buy products.



1. Use cases

- A use case is a functional requirement that is described from the perspective of the users.

- It is shown as an ellipse labeled with the name of the use case.

- Always named with the goal of the actor.

1. Use cases (Cont.)

- Each use case is composed of one or more behavior sequence/scenario.
- A scenario is a sequence of steps describing an interaction between a user and a system.

Example: "Buy a Product" use case

1. **Customer browses catalog and selects items to buy.**
2. **Customer goes to check out.**
3. **Customer fills in shipping information**
4. **System presents full pricing information including shipping**
5. **Customer fills in credit card information.**
6. **System authorizes purchase**
7. **System confirms sale immediately**
8. **System sends confirming e-mail to customer.**

Other scenarios for "Buy a Product"

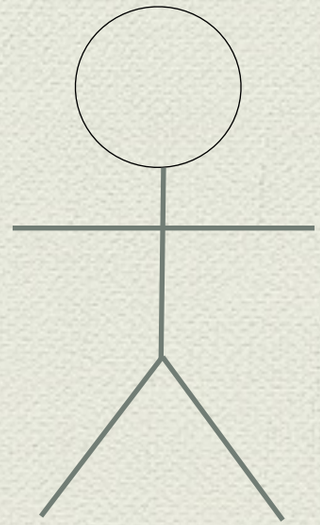
- The credit card authorization might fail, and this would be a separate scenario.
- In another case, you may have a regular customer for whom you don't need to capture the shipping and credit card information, and this is a third scenario.

What makes a good use case

- A rule of thumb!
  - A use case is something that provides some measurable result to the user or an external system.
  - Don't define each step in a use case as a use case! e.g. “fills in shipping information” is NOT a good use case.
- Natural language is used.

2. Outside your system: Actors

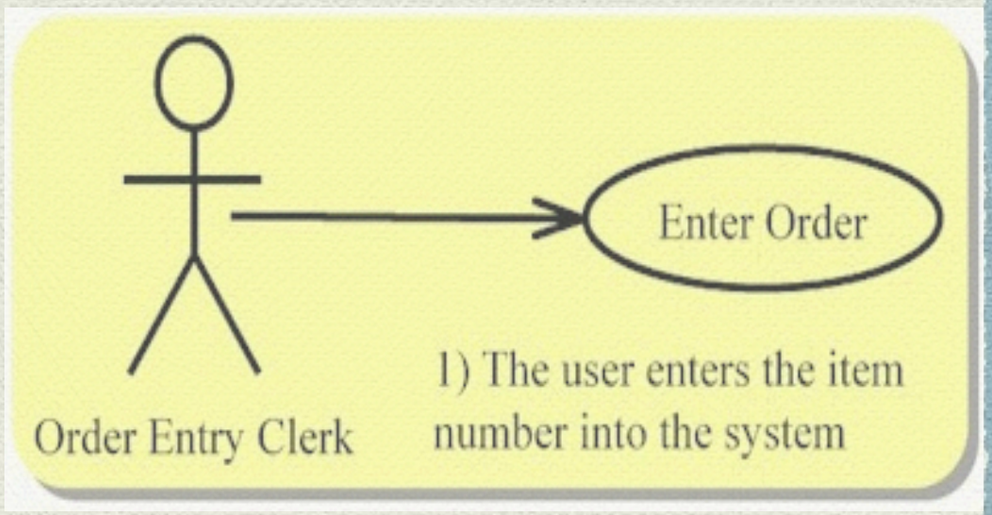
- Actors are entities which require help from the system to perform their task or are needed to execute the system's functions.
- *Interact* with the system but not part of it!
- An actor is drawn in UML using either a "stick man" or a stereotyped box and labeled with an appropriate name *(abstract name)*.



Actor name

2. Outside your system: Actors (Cont.)

- A use case is started by an actor.
- The actor is the source of the incoming event that is the first line in the use case.
- This actor is the primary or active actor.



2. Outside your system: Actors (Cont.)

- Answer the question of "who and what interacts with the system?"
- It might be a person, a system, or some external entity. Like?
- Linked to one or more system use cases.
- It is always *outside* the system being modeled.

Do you think that time could be an actor?

3. Associations(Relationships)

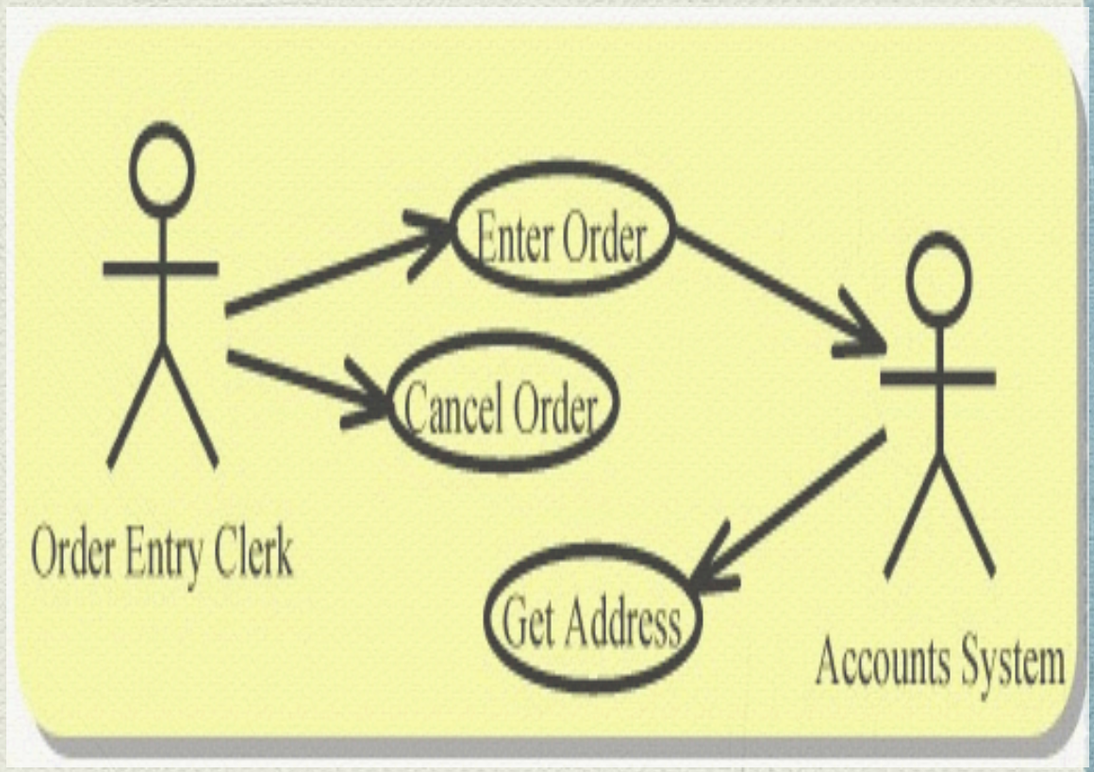
There are several types of relationships that may appear on a use case diagram:

1. An association between an actor and a use case.
2. An association between two use cases.
3. A generalization between two actors.
4. A generalization between two use cases.

Actor - use case relationship

- The association between the actor and a use case indicates that the actor uses the use case.

- The direction of the arrow is determined based on the type of the actor.

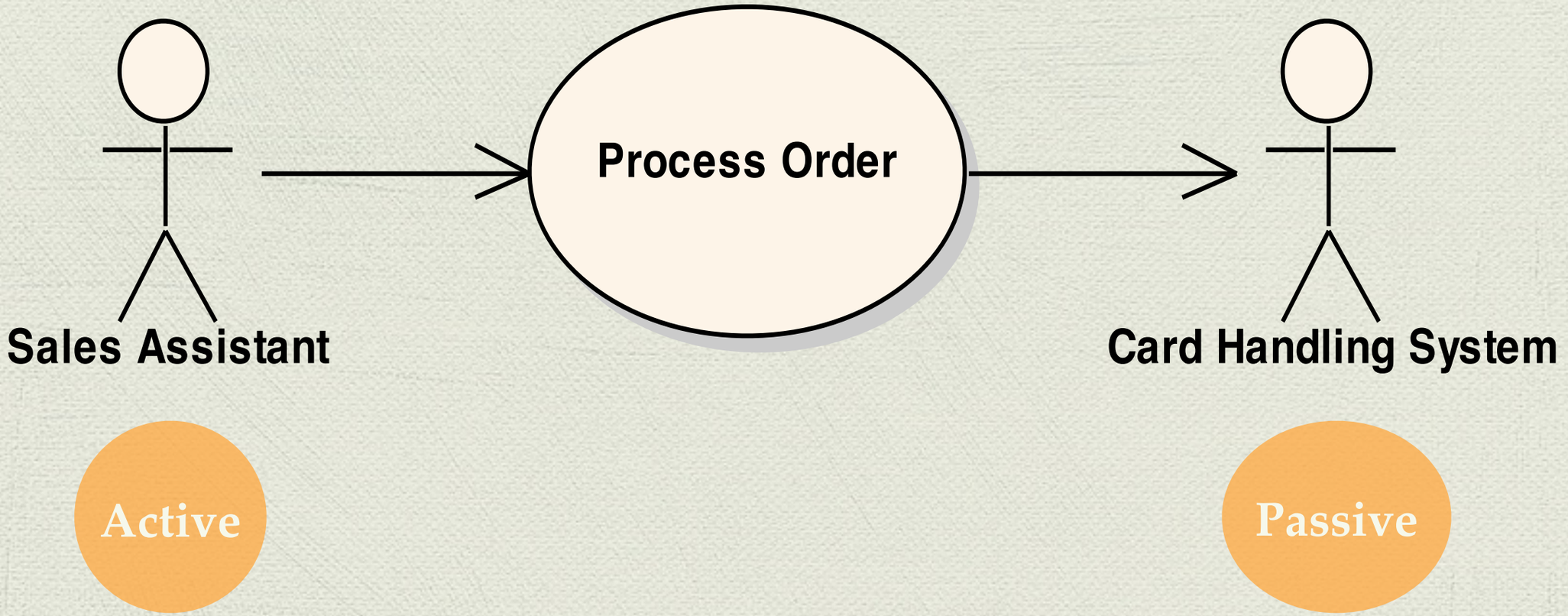


Active and Passive Actors

- An active actor is one that starts a use case by supplying the interaction across the system boundary that starts the use case.

- A passive actor interacts with the system as part of the use case but does not start it.

Example



Use a directed association from an active actor to a use case and a directed association from a use case to a passive actor.

4. System Boundary (Optional)

- A rectangle surrounding use cases.
- Anything outside your system should be outside the system boundary.
- It's good practice to name your box with the name of your system being developed.

How much it helps!

- Both time and money are saved!
  - Use cases are a means to bring gaps and the lack of understanding in the user's requirements to the forefront at the beginning of a project.
- It can help manage a project's workload
  - Once priority and risk are assigned to a use case!
- Serves as basis for testing
  - *What better way to test your system than by using the use cases that originally captured what the user wanted in the first place?*

5 minute BREAK !!

Example: University Record System

System Description

A university record system should keep information about its students and academic staff.

Records for all university members are to include their id number, given name, email, address, date of birth, and telephone number.
Students will also have a list of subjects they are enrolled in.
A student cannot be enrolled in any more than 10 subjects.
Academic employees will have a salary, and a list of subjects they teach. An academic can teach no more than 3 subjects.

Example: University Record System

The system should be able to handle the following commands:

1. Administrators can add and remove university members (students, and academic staff).
2. Administrators can add and delete subjects.
3. Students can enroll and un-enroll in subjects.

References

- Alhir, S. (2003) *Learning UML*. Sebastopol: O'Reilly Media, Inc.

- Fowler, M. (2004). UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional.

- Miles, R and Hamilton, K. (2006) Learning UML 2.0. Sebastopol: O'Reilly Media, Inc.

- UML Training Course from CRaG System, http://www.cragsystems.co.uk.