Software Engineering

Requirement Analysis: Quick Recap

Requirements Analysis: Quick Recap

- Requirements define the characteristics of the product (be careful with functional requirements and non-functional requirements)
- The requirements analysis phase identifies requirements and collect them into a requirement document
- UML Approach: Use Case Diagram and Sequence Diagram

UML Diagrams

- **Static**
  - Package Diagram
  - Use case diagram
  - Class diagram and Object diagram
  - Component diagram
  - Deployment diagram
- **Dynamic**
  - Interaction diagrams
    - Sequence diagram
    - Collaboration diagram
  - State diagram
  - Activity diagram

UML Diagrams: Main Uses

- **Static**
  - **Package Diagram**: Help you organize your model
  - **Use case diagram**
    - Business Modeling: Processes
    - Requirements: Functional Requirements
  - **Class diagram and Object diagram**
    - Business Modeling: Organization, Entities
    - Analysis And Design: Logical Architecture

UML Diagrams: Main Uses

- **Static**
  - **Component diagram**
    - Design:
      - Physical Architecture
      - DB Structure
      - Structuring in Files
  - **Deployment diagram**
    - Design: Deployment of Executable in Complex Networks

UML Diagrams

- **Dynamic**
  - **Interaction diagrams**
    - Business Modeling, Requirements: Interaction User-system
    - Analysis And Design: Interaction Among Elements of The System
    - Requirements: Functional Requirements (Sequence Diagram)
  - **State diagram**
    - Analysis and Design: Specification of the Behaviour of the System
  - **Activity diagram**
    - Business Modeling: Workflow
    - Analysis and Design: Specification of the Behaviour of the System

UML Diagrams

- Static
  - Use case diagram
  - Class diagram and Object diagram
  - Component diagram
  - Deployment diagram
- Dynamic
  - Interaction diagrams
    - Sequence diagram
    - Collaboration diagram
  - State diagram
  - Activity diagram

Software Engineering

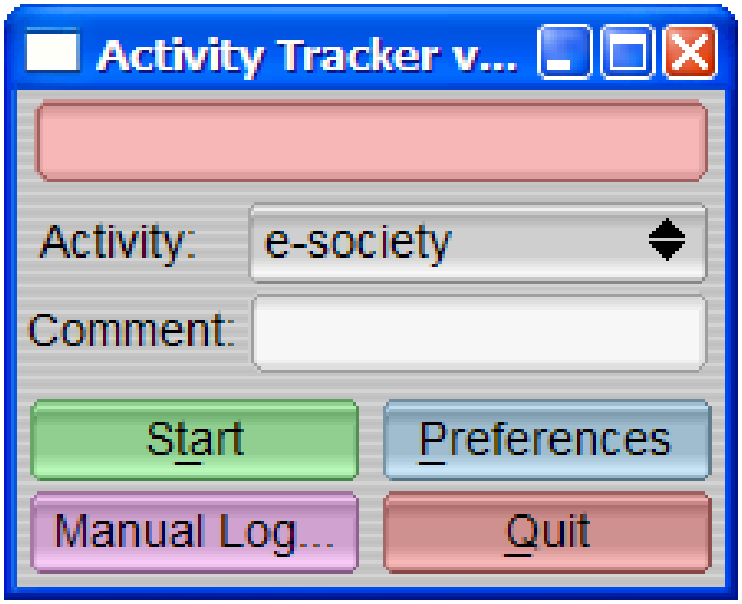
Use Case Diagram

Use Case Diagram

- Use Case
  - Set of **scenarios** describing a coherent unit of functionality **as seen by a user of the system**.
- Actor:
  - Set of **roles** that users can play
- Use Case Diagram
  - A graph collecting the use cases, the actors taking place in the use cases, and the relationship among use cases and actors.

Example: Time Tracker

We have been contacted by a small firm. They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports.



The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects. This information is then used to charge clients.

Step 1: Identify Actors

An actor is someone or something that must interact with the System Under Development

Step 1: Time Tracker

We have been contacted by a small software firm.

They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports by an administrator.

The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects.

Step 1: Identify Actors

An actor is someone or something that must interact with the System Under Development

Step 2: Identify Use Cases

A use case is a pattern of behavior the system exhibits

Each use case is a sequence of related transactions performed by an actor and the system in a dialogue

Strategies for identifying Use Cases

- Examine Actors to determine their needs
- Find verbs and actions in the description

Step 2: Time Tracker

We have been contacted by a small software firm.

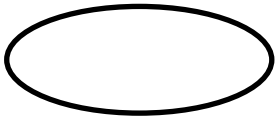
They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports by an administrator.

The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects.

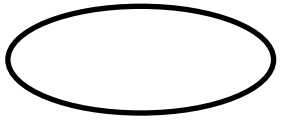
Step 2: Identify Use Cases

A use case is a pattern of behavior the system exhibits

Each use case is a sequence of related transactions performed by an actor and the system in a dialogue



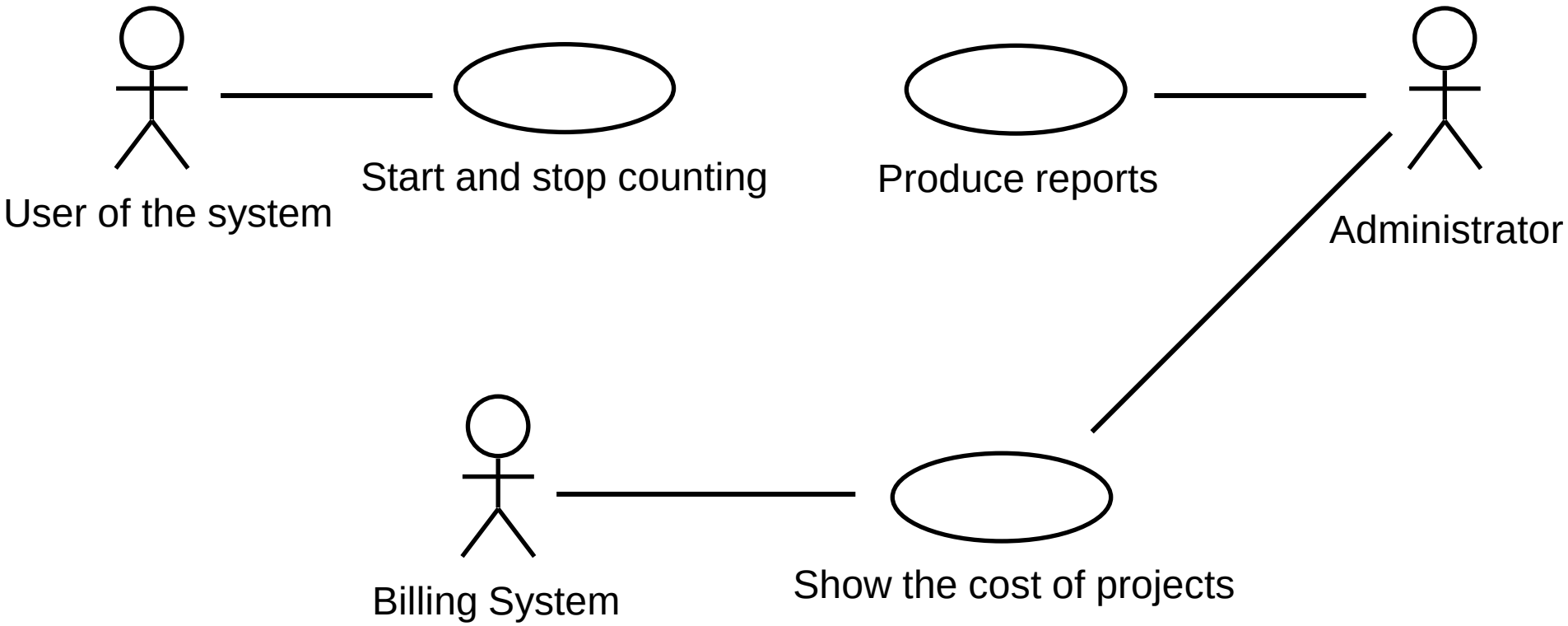
Start and stop counting



Produce reports

Step 3: Use Case Diagram

Use case diagrams are created to visualize the relationships between actors and use cases



Use Case Diagram: Elements

- Actor is a coherent set of roles that users of an entity can play
  - Actors need not be human
  - Actors represent roles and not instances (e.g. a person can play different roles when using a system)
- Use Case represent a coherent unit of functionality of the system

Software Engineering

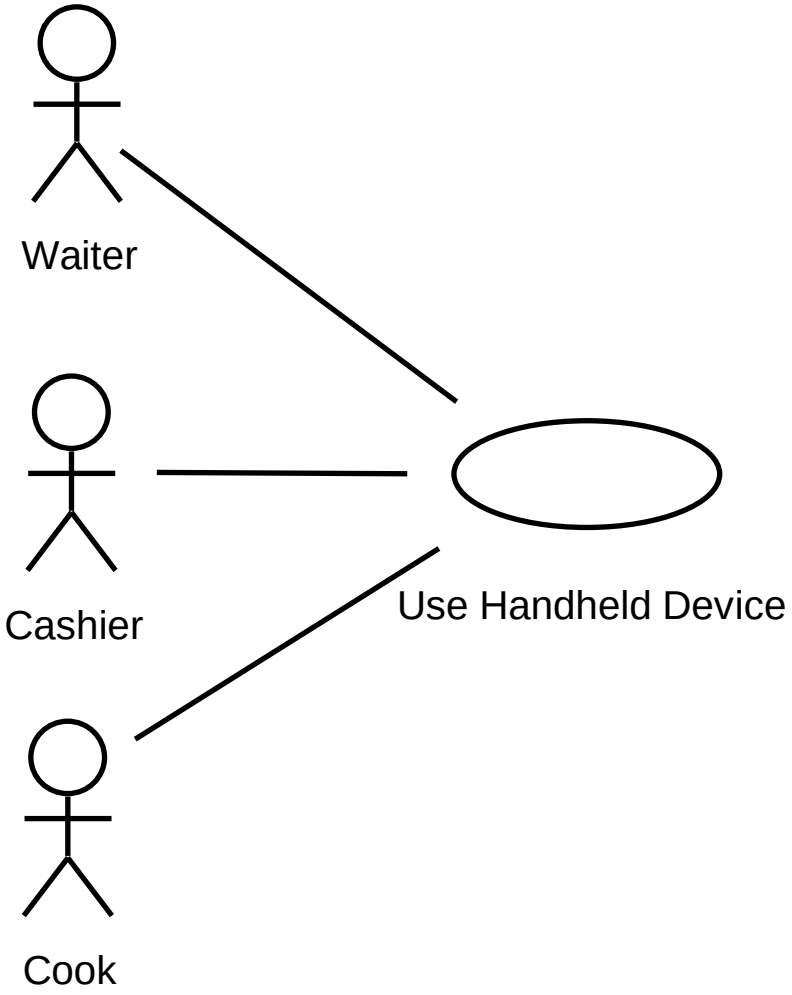
Restaurant Management System

Restaurant Management System

Restaurant Management System

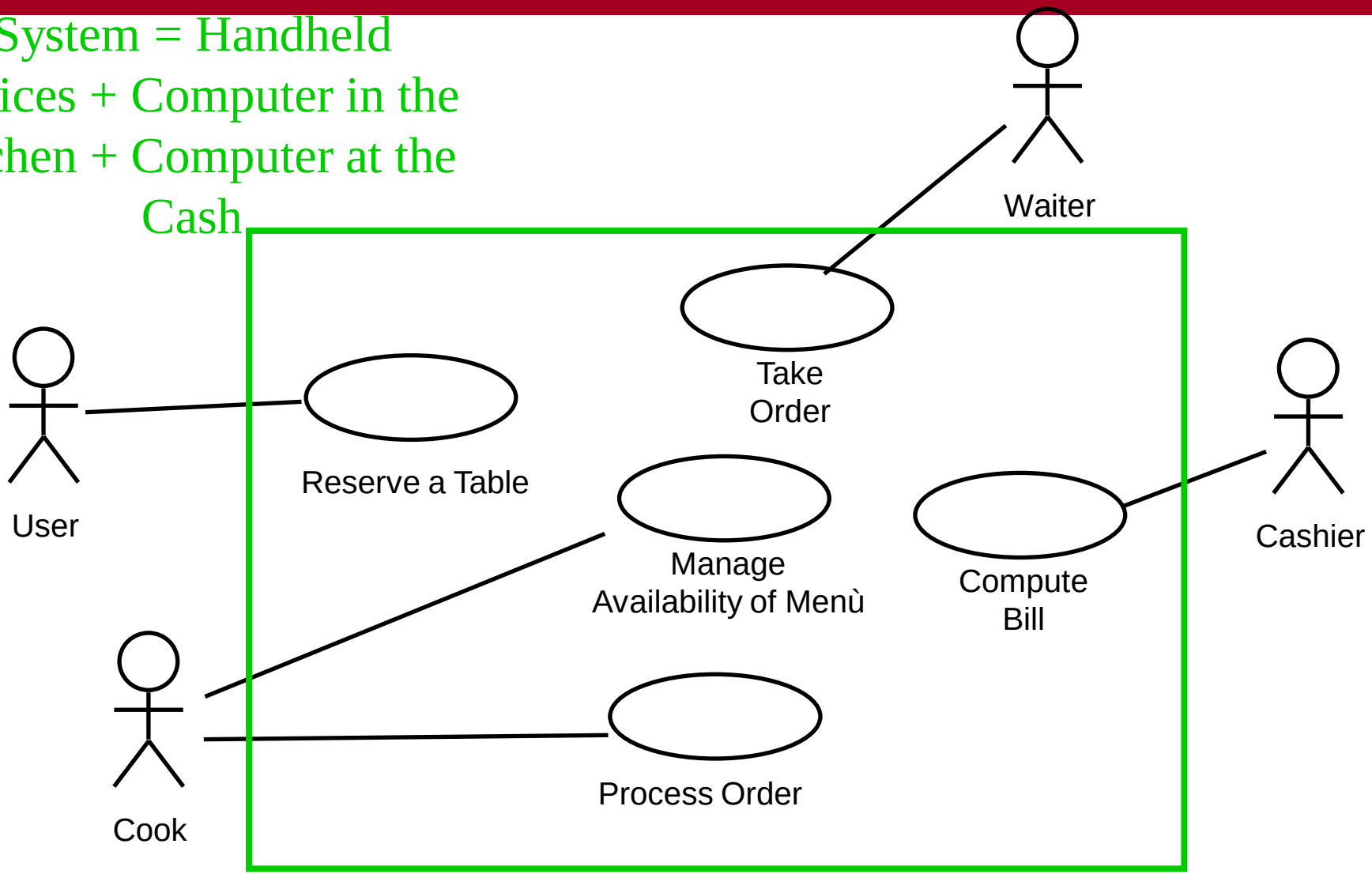
We have been asked to build a system to automate the ordering and billing activities of a restaurant. The system is distributed: waiters and waitresses are provided with handheld devices to take orders. The handheld devices communicate orders to the kitchen and to the cashier. The handheld devices receive real-time information about availability of the different items in the menu. Once placed, orders can be changed by the customers, within a time frame from the order (5 minutes) or after the time-out, if the corresponding order has not yet been processed by the Cook. The system computes bills and is also used to manage reservations of tables. Reservations can either happen by phone or via the internet.

Restaurant Management System



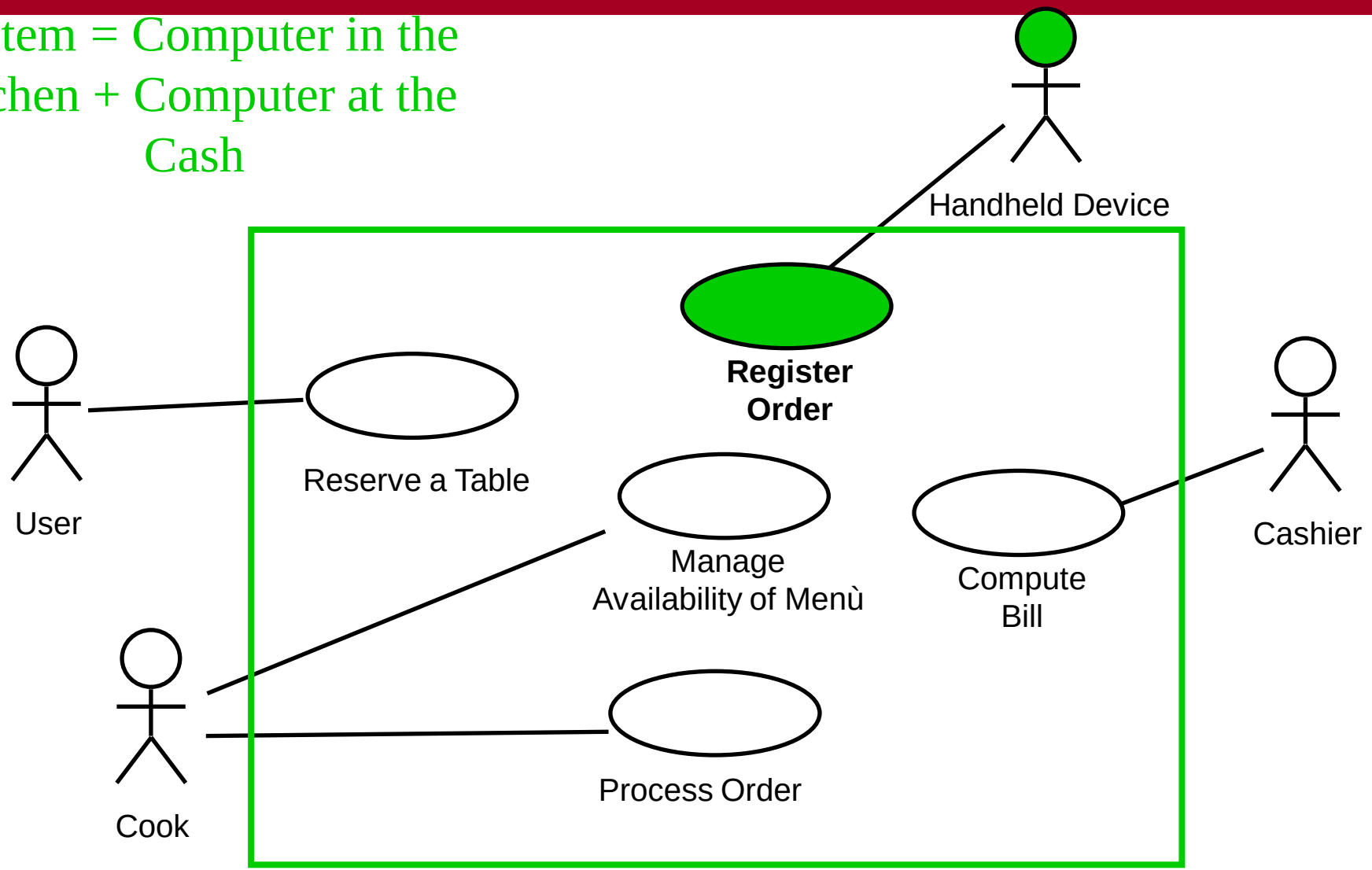
Resturant Management System

System = Handheld devices + Computer in the kitchen + Computer at the Cash



Resturant Management System

System = Computer in the kitchen + Computer at the Cash



Software Engineering

Describe Use Cases

Describe Use Cases

- A flow of events document is created for each use cases
  - Written from an actor point of view
- Details what the system must provide to the actor when the use cases is executed
- Typical contents
  - How the use case starts and ends
  - Normal flow of events
  - Alternate flow of events
  - Exceptional flow of events

Describe Use Cases

In theory, no particular format for the description of a use case.

In practice, follow a convention.

Title.

Summary.

Description.

Exceptions.

Extension Points.

Describe Use Cases: Example

Title: Buy Ticket

Summary:

This use case describes how tickets can be bought using the reservation system.

Description

Step 1. The User chooses a show and a date. **[exception 1]**

Step 2. The System shows the seats available.

Step 3. The user provides name and a credit card number **[extension 1]**

…

Exceptions

[exception 1] If the name of the show and the dates do not correspond, an error message is issue.

…

Extension Points

[extension 1] At step 3, "loyal customers" simply need to enter their PIN.

…

Software Engineering

Making Use Cases more coherent

Some open issues…

- Use case coherence

  As use cases are informal, there is a tendency to forget details when writing use cases.

- Use case proliferation

  By describing requirements through scenarios, use cases tend to proliferate (and use case diagrams grow too complex).

Software Engineering

Use Case Diagram: Relationships

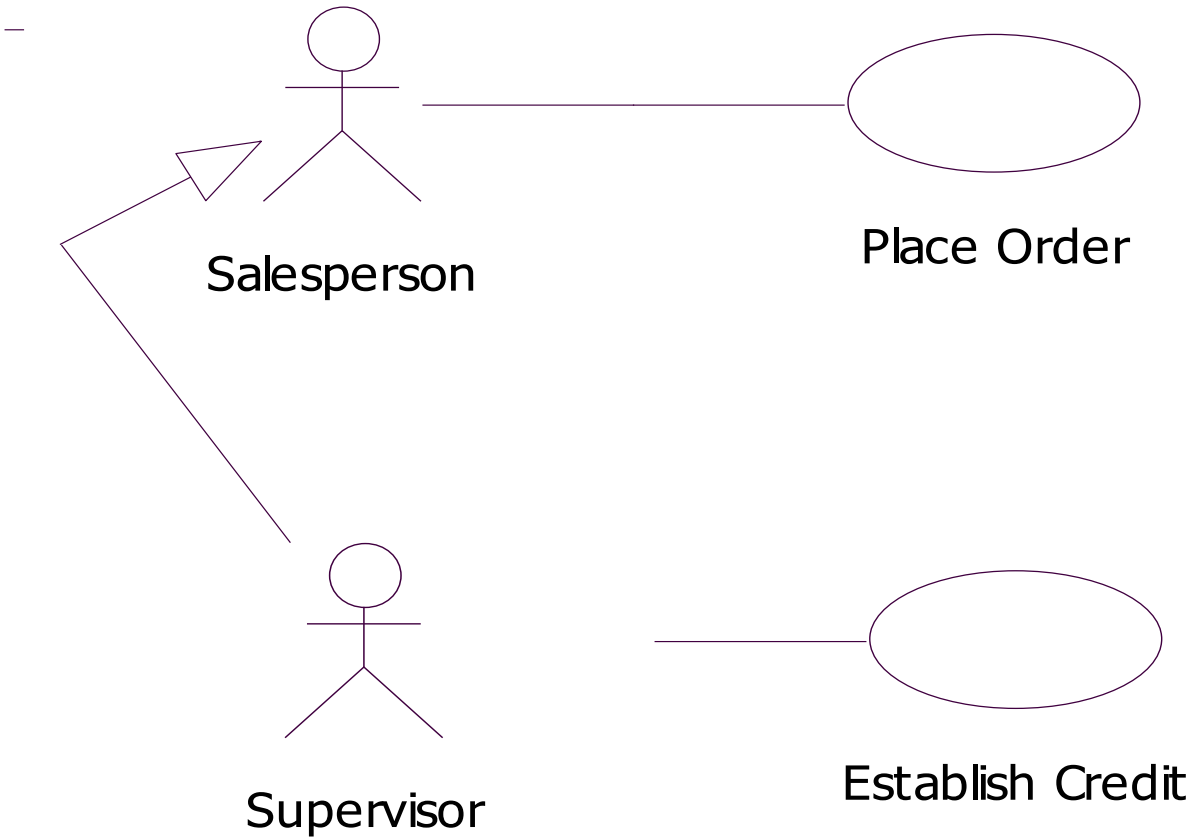
Use Case Diagram: Relationships

- **Actors Relationships**

Association:
The participation of an actor in a use case. **This is the only relationship between Actors and Use Cases**

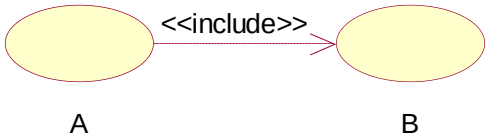
Generalization:
A generalizes B implies that A can perform the use cases of B.

Actors Relationships: Example



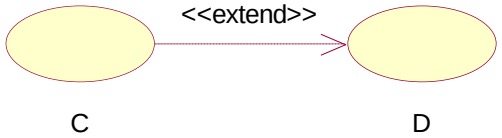
Use Case Diagrams: Relationships

- **Use Case Relationships**



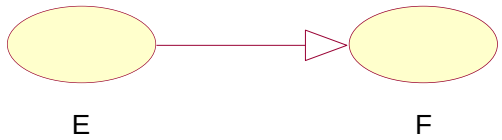
Include:

a step of A is the execution of B



Extend:

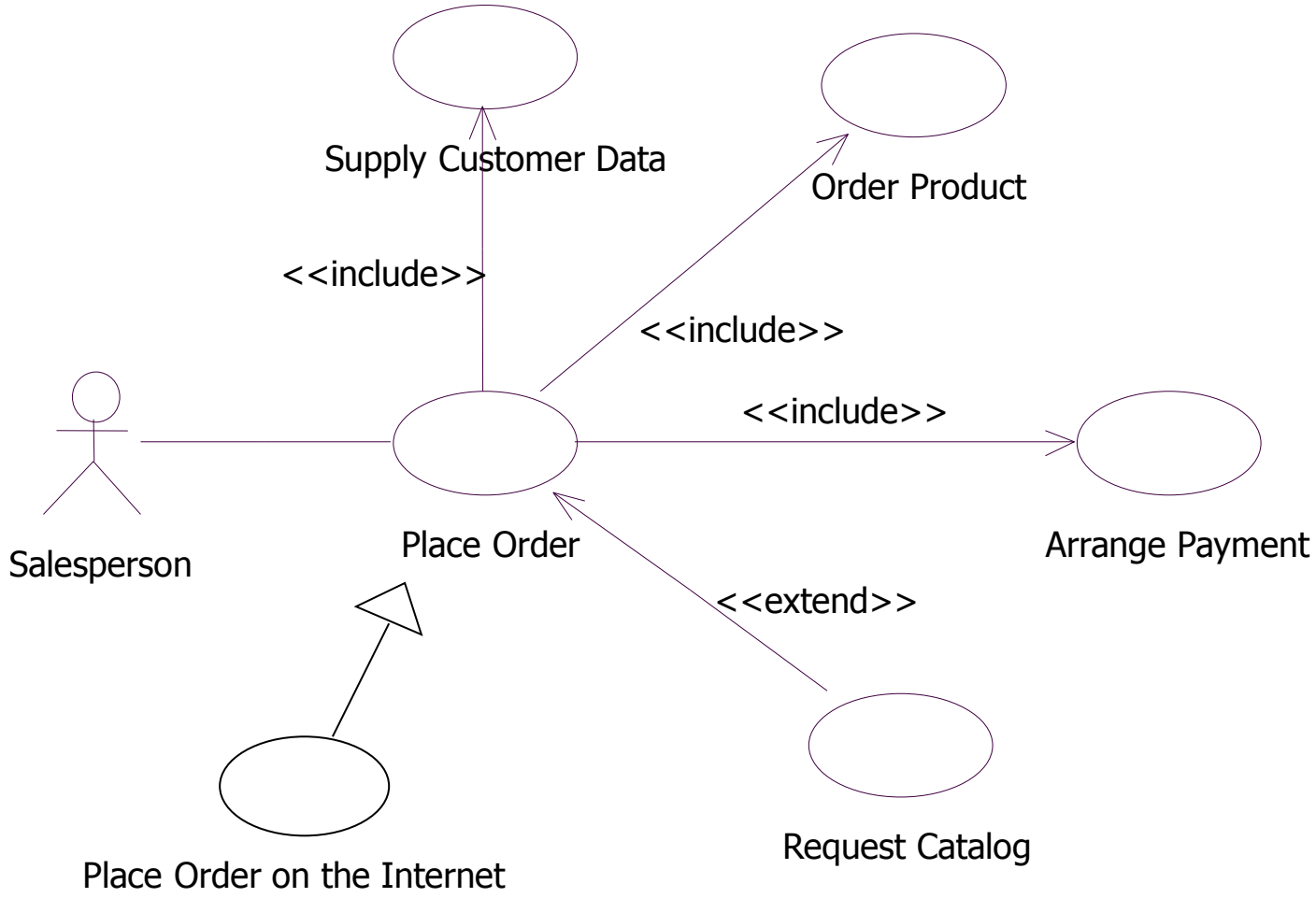
C does a little more than D does



Generalize:

E does something more specific than F

Use Case Relationships: Example



Hint 1: Actors

- Actors are external to the system:
  - They help setting the boundaries of the system…
  - … as a consequence: the “system” itself can’t be actor!
- Actors need not be human:
  - Braking System, Hydraulic System, etc are OK!

Hint 2: Actors

Sometimes you need to represent functionality that take place on a regular basis (e.g. backup)

Who is the actor for such a use case?

- **Solution 1.** Who benefits from the execution of the regular action (System Administrator)
- **Solution 2.** What triggers the action, namely: “Time” or “Clock”.

… both 1 and 2 are ok!

Exercise:

Reservation System

We want to build a system for the electronic reservation of seats of a group of movie theatres. The users can either buy tickets for a particular show or buy subscriptions, also by phone. Payments can be performed in cash or by credit card. A supervisor can print the list of seats available for a particular show and see the status of sales performed so far.

A secretary updates the list of shows (by adding/deleting/...)