Object Oriented Design

Requirements Analysis/Business Modelling

Dr. Seán Russell

School of Computer Science,
University College Dublin

Lecture 05

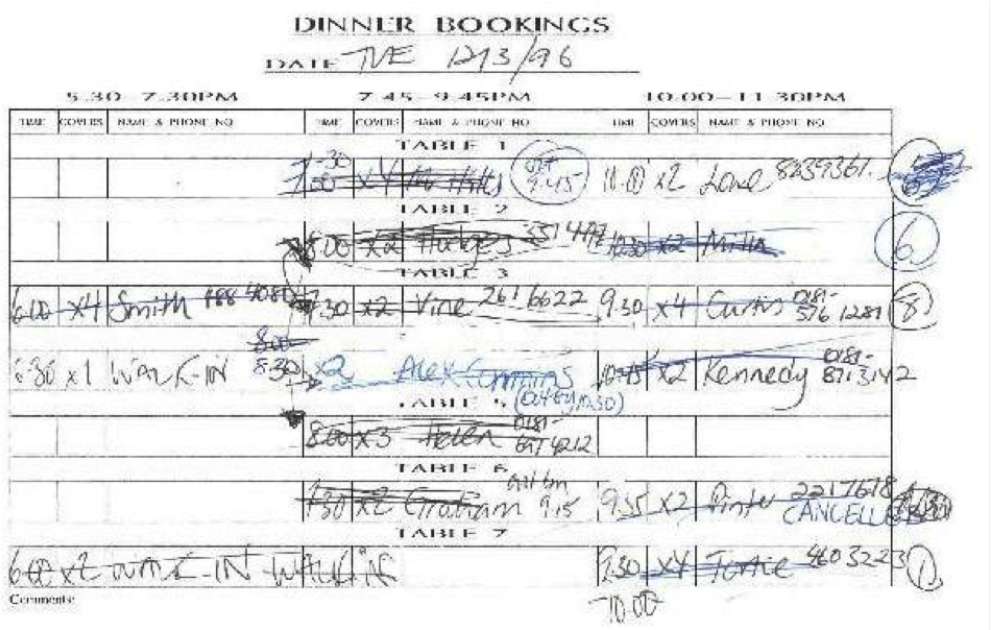
Table of Contents

Case Study

- In the next lectures we will consider a case study
  - This lecture will cover the Requirements analysis
  - There will be a lecture covering Analysis
  - There will be a lecture covering Design
  - There will be a lecture covering Implementation
- The purpose is to learn the UML diagrams used in these workflows as well as some experience with the process
- This will represent a single iteration in the Unified Process

Section Contents

- Our system will support the operations of a restaurant
- The system will record and display reservations made at the restaurant as well as time and table assignments


DINNER BOOKINGS

DATE TUE 12/3/96

5.30 - 7.30PM | 7.45 - 9.45PM | 10.00 - 11.30PM

TABLE | COVERS | NAME & PHONE NO

TABLE 1

TABLE 2

TABLE 3

TABLE 4

TABLE 6

TABLE 7

Comments:

- Slow to update and difficult to read
- No backup of information
- Difficult to analyse data

- A replacement system should show the same information in roughly in the same format
- We should be able to alter data easily and updates should be shown immediately

- In the first iteration we will implement just core functionality
  - Record a booking in the system
  - Change the time/table of a booking
  - Cancel a booking

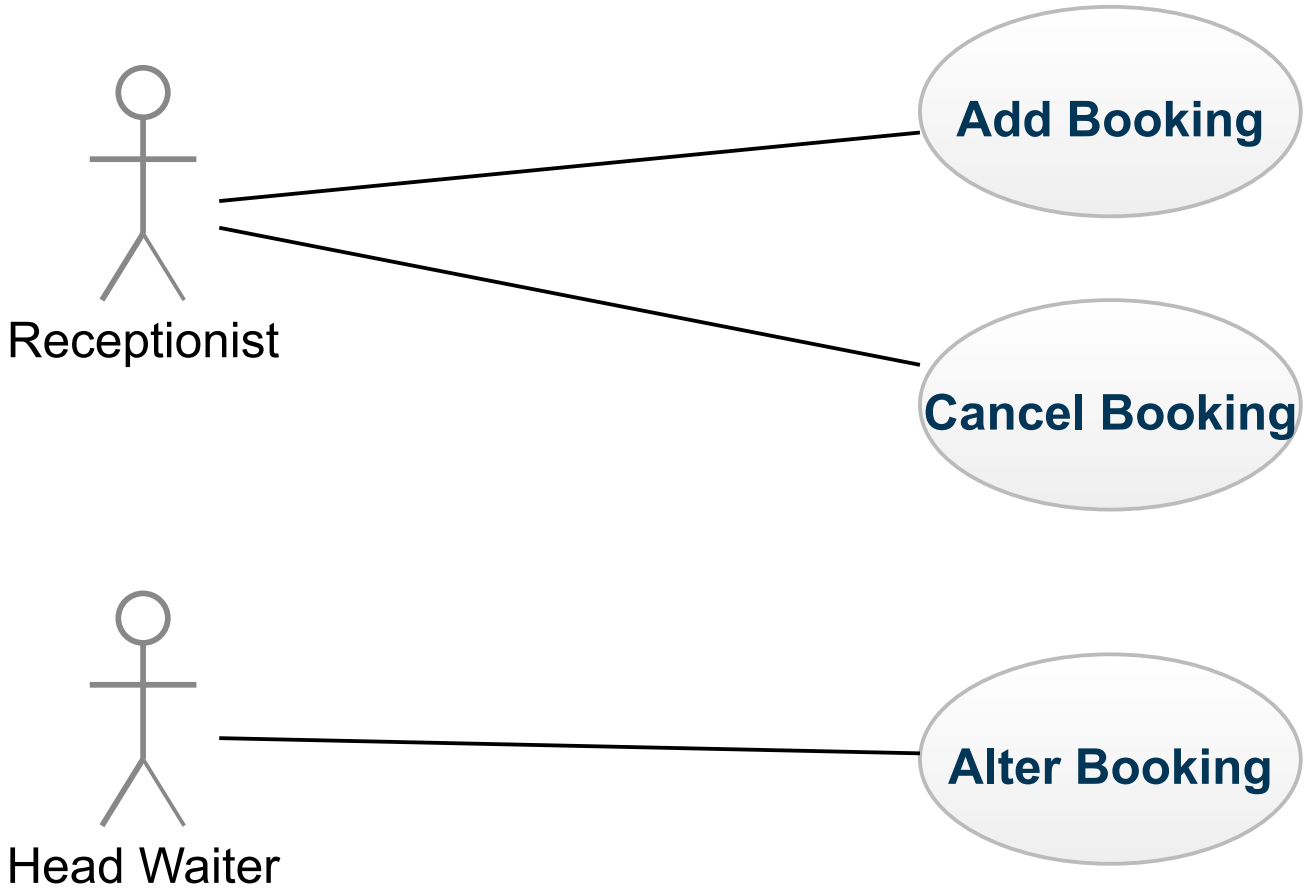
Section Contents

- CRUD is an acronym commonly used to describe the principle operations of databases
  - I.e. create, read, update and delete
- These can also be a useful description for what is happening in use cases

- Use case modelling is creating a structured view of a systems functionality
- We define the **actors** and **use cases**
  - An actor is a role that can be played in interaction
  - A use case describes a specific task that we can achieve
- Use case models should be understandable by anyone involved with the system

- Use cases are usually defined by consulting with users
- We will create use cases based on what restaurant staff should be able to do
  - Remember details about a booking ('Add booking')
  - Cancel a booking ('Cancel booking')
  - Change time/table of a booking ('Alter booking')

- Actors describe different categories of users within a system
- They often have different abilities in the system
- In the restaurant, we can identify the following user responsibilities:
  - Maintaining information on advanced bookings (‘Receptionist’)
  - Actions when the restaurant is open (‘Head waiter’)


Add Booking
Receptionist
Cancel Booking
Alter Booking
Head Waiter

Section Contents

- A use case describes a class of interactions between system and its users
- Each time we perform these steps is an **instance** of the use case
- Details will be often be different, also the steps and results can be different
- A complete description specifies all possible instances of the use case

- A use case description can be a lot of information
- UML does not define how this information must be formatted
- I will share a template that we can use in this class

- Name
- Description
- Actors
- Triggers
- Preconditions
- Postconditions
- Courses of events
- Inclusions
- Data Outcomes

- A course of events is a possible interaction between the user and system
- Usually shown like a dialogue
  - The user performs some action
  - The system responds in some way
- This is repeated until the use case is completed
- Sometimes named as **basic**, **alternative** or **exceptional** course of events

Name: Add Booking
Description: The creation of a new booking by the user. This includes entering customer and booking information as necessary.
Actors: Receptionist
Preconditions: None
Postconditions: All bookings in the system for the specified date will be displayed on the screen, including the newly created one.
Courses of Events: Shown later
Extension Points: None
Inclusions: None
Data Outcomes: **CREATE** - A new object will be added to the system representing the booking

1. The receptionist enters the date of the requested reservation
2. The system displays the bookings for that date
3. There is a suitable table available; the receptionist enters the customers name and phone number as well as the booking details (table, time, date, covers)
4. The system updates the display to show the current bookings for the specified date (including the newly created one)

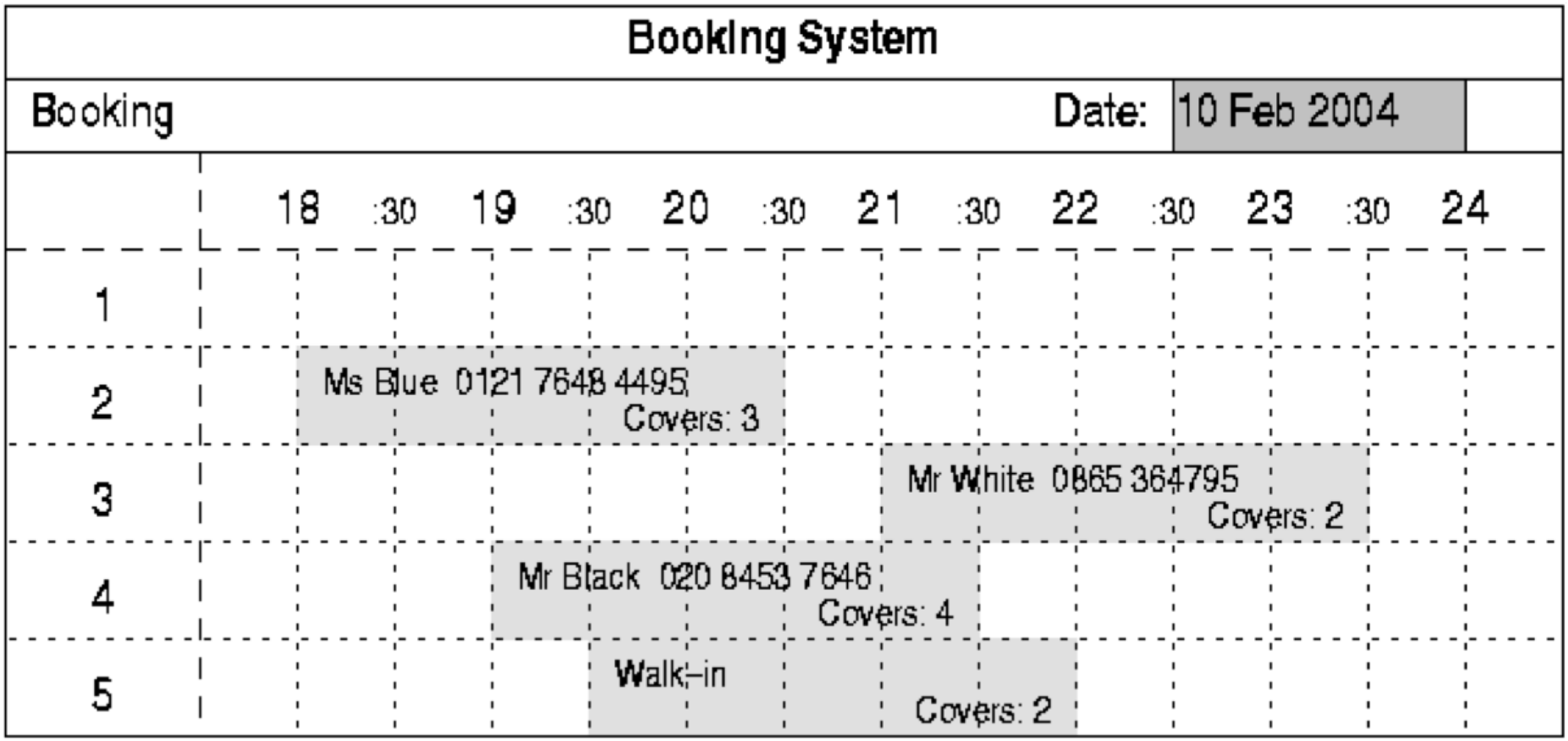
- In a course of events we only care about the interactions between the user and the system
- Including information about the users interactions with customers is not essential
  - We do not care how the user learned the customers name and phone number
- Being too specific in use cases can potentially make use cases less reusable

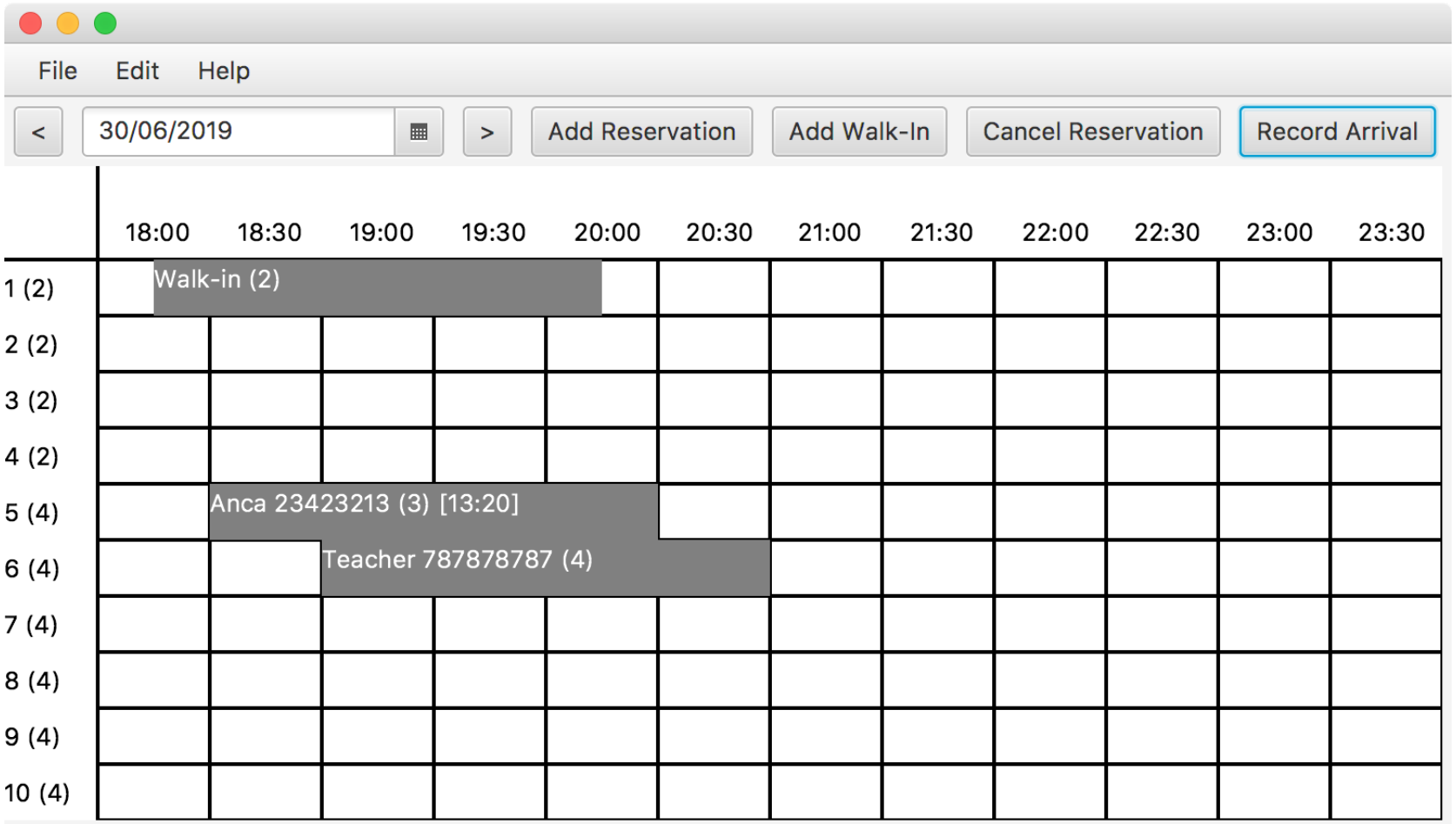
1. The receptionist enters the date of the requested reservation
2. The system displays the bookings for that date
3. No suitable table is available and the user performs no more actions

1. The receptionist enters the date of the reservation
2. The system displays the bookings for that date
3. The receptionist enters the customers name and phone number as well as the booking details (table, time, date, covers)
4. The system warns the user that the table selected has less places than the booking is made for and asks the user to confirm if this is ok
5. The user selects yes
6. The system updates the display to show the current bookings for the specified date (including the newly created one)

1. The receptionist enters the date of the reservation
2. The system displays the bookings for that date
3. The receptionist enters the customers name and phone number as well as the booking details (table, time, date, covers)
4. The system warns the user that the table selected has less places than the booking is made for and asks the user to confirm if this is ok
5. The user selects no
6. The system returns to the display to show the current bookings for the specified date

- Use case descriptions should not give details of **how** interaction is done
- This may restrict our choices of how to implement it later
- However, having a general idea of the user interface is a good idea
- The restaurant booking system will be similar in structure to the sheets it is replacing


Booking System
Booking
Date:
10 Feb 2004
18
:30
19
:30
20
:30
21
:30
22
:30
23
:30
24
1
2
Ms Blue 0121 7648 4495
Covers: 3
3
Mr White 0865 364795
Covers: 2
4
Mr Black 020 8453 7646
Covers: 4
5
Walk-in
Covers: 2


File
Edit
Help
<
30/06/2019
>
Add Reservation
Add Walk-In
Cancel Reservation
Record Arrival
18:00
18:30
19:00
19:30
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30
1 (2)
2 (2)
3 (2)
4 (2)
5 (4)
6 (4)
7 (4)
8 (4)
9 (4)
10 (4)
Walk-in (2)
Anca 23423213 (3) [13:20]
Teacher 787878787 (4)

Section Contents

Name: Alter Booking
Description: The modification of details of a booking (time or table number)
Actors: Receptionist
Preconditions: The booking to be modified must exist
Postconditions: The specified booking will be moved to a new location to represent the change.
Courses of Events: Shown later
Extension Points: None
Inclusions: None
Data Outcomes: **UPDATE** - The information inside the specified booking will be changed

1. The head waiter enters the current date
2. The system displays the bookings for that date
3. The head waiter selects the relevant booking from the display
4. The system highlights the booking to indicate it is selected
5. The head waiter updates the time/table for a selected booking
6. The system records change and updates the display

1. The head waiter enters the current date
2. The system displays the bookings for that date
3. The head waiter selects the relevant booking from the display
4. The system highlights the booking to indicate it is selected
5. The head waiter updates the time/table for a selected booking
6. The system alerts that this overlaps with another booking and cancels the change

- Every course of events has started with an actor entering a date and the system displaying the bookings
- It is better to define this shared behaviour in one place
- It can be then used later in the other use cases

1. The user enters a date
2. The system displays the bookings for that date

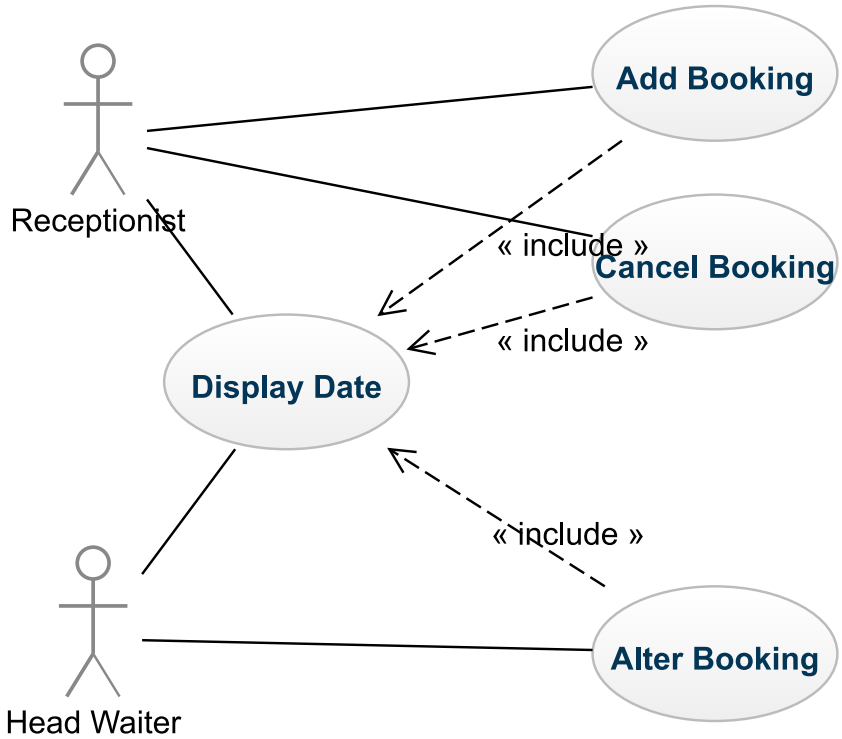
Name: Display Bookings
Description: Showing all of the bookings made for the specified date.
Actors: Receptionist/Head Waiter
Preconditions: None
Postconditions: All bookings in the system for the specified date will be displayed on the screen.
Courses of Events: Shown previously
Extension Points: None
Inclusions: None
Data Outcomes: **READ** - The details of the existing bookings for a date will be displayed

- One use case may be performed as a part of another
  - This is called **use case inclusion**
- It must be made clear in the use case diagram and in use case descriptions

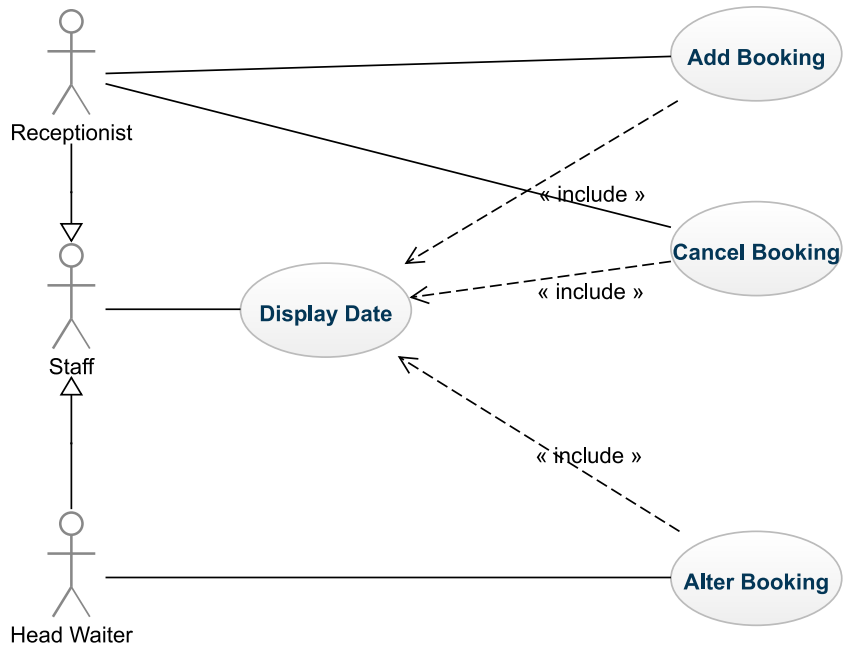
Add booking: Basic course of events (revised)

1. The receptionist performs the 'Display bookings' use case
2. There is a suitable table available; the receptionist enters the customers details and booking details
3. The system records the new booking

- The includes relationship is as a dashed arrow
- This is known as a **dependency**
- It is also labelled with a **stereotype** specifying the kind of relationship



- The first use case showed which actors could perform which use cases
- But now, both actors can perform display bookings
- This can be shown on the use case diagram using **generalisation/specialisation**



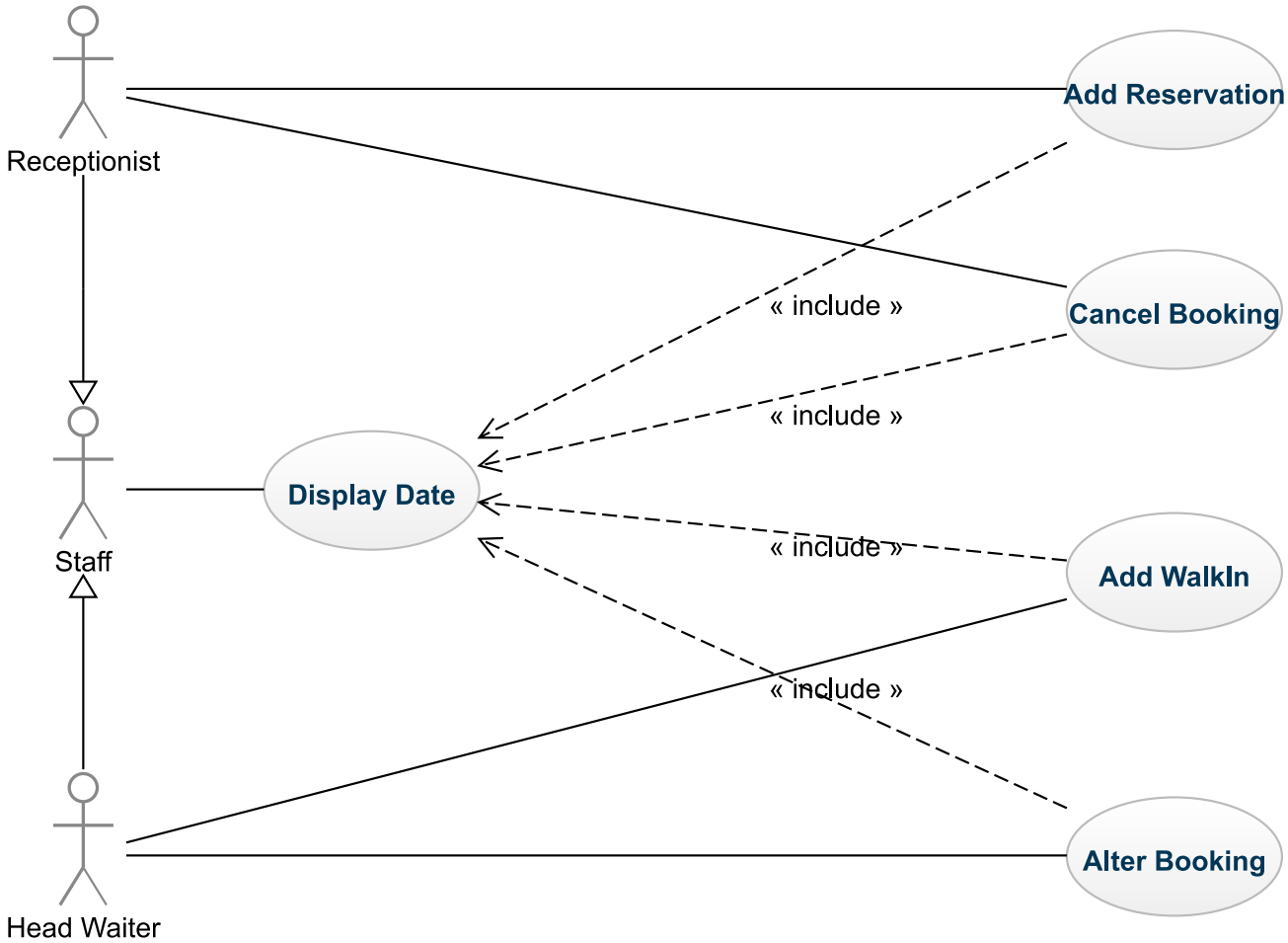
Section Contents

1. The receptionist performs the 'Display bookings' use case
2. The receptionist selects the required booking
3. The system highlights the booking to indicate it is selected
4. The receptionist cancels the booking
5. The system asks the receptionist to confirm the cancellation
6. The receptionist answers 'yes',
7. The system records the cancellation and updates the display

1. The head waiter performs the 'Display bookings' use case
2. The head waiter selects the required booking
3. The system highlights the booking to indicate it is selected
4. The head waiter changes the time/table of the booking
5. The system records the alteration and updates the display

- Some customers arrange their booking in advance, we remember their name and number so we can contact them
- Other customers simply arrive at the restaurant, we do not need to contact them in the future
- This mean we need to allow for the creation of a "walk-in" booking
- This use case will be slightly different to the "add booking" use case
  - We should rename that use case "add reservation"

1. The receptionist enters the date of the requested reservation
2. The system displays the bookings for that date
3. There is a suitable table available; the receptionist enters the booking details (table, time, date, covers)
4. The system records the new booking

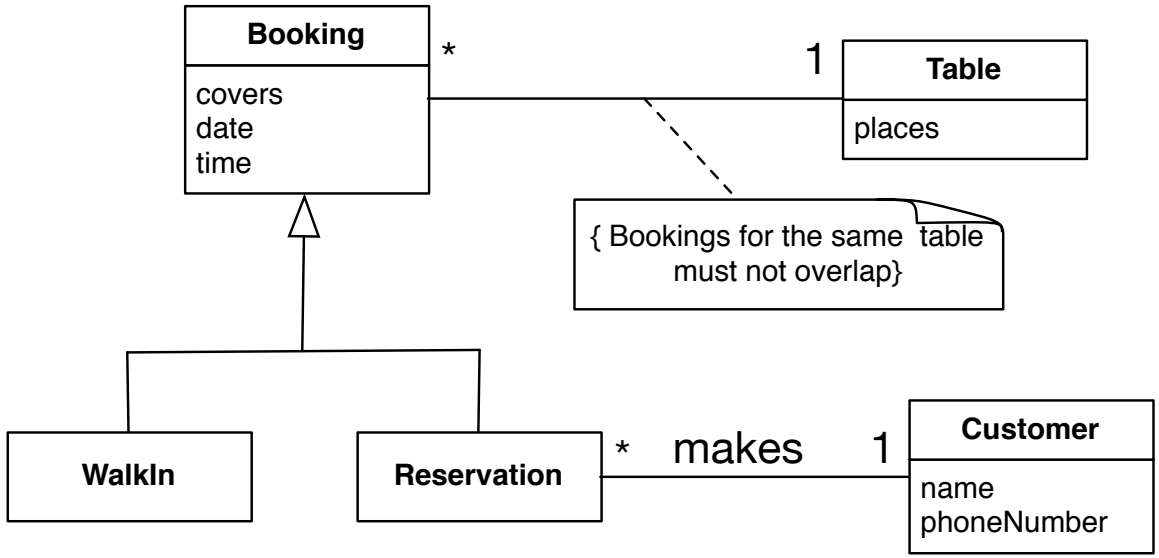

Receptionist
Staff
Head Waiter
Add Reservation
Cancel Booking
Display Date
Add WalkIn
Alter Booking
« include »
« include »
« include »
« include »

Section Contents

- Domain modelling is usually completed at the same time as use case modelling
- The concept is to produce a **class diagram** showing the important business concepts and their relationships
  - This is known as a domain model

- Key Concepts:
  - Customers
  - Reservations
  - Table
  - Walk-in

- Key Concepts:
  - Customers
  - Reservations
  - Table
  - Walk-in



Section Contents

- One useful outcome of domain modelling is a detailed consideration of the concepts and vocabulary that clients use
- It is very easy to use terms ambiguously or inconsistently when writing informally about a system
- It is often useful to summarise a system's core vocabulary in a series of definitions collected together into a **system glossary**
- Ideally, once a glossary has been created, all the system documentation should be edited to make consistent use of the defined terms

- **Booking** An assignment of a table to a party of diners for a meal
- **Covers** The number of diners that a booking is made for
- **Customer** The person making the reservation
- **Diner** A person eating at the restaurant
- **Places** The number of diners that can be seated at a particular table
- **Reservation** An advance booking for a table at a particular time and date
- **Walk-in** A booking that is not made in advance

Section Contents

- CRC modelling is an exercise for identifying classes/objects and their relationships and responsibilities
- Usually completed as a group exercise, one CRC card is created for each class/object in the design
- Cards are split into 3 parts
  - The name of the class
  - The responsibilities of the class
  - The collaborators of the class

- Choosing classes and responsibilities to represent is not always easy
- The following are good starting points
- Identifying classes:
  - Search for **nouns** in specifications
- Identifying responsibilities:
  - Search for **verbs** in specifications