University of
South Australia

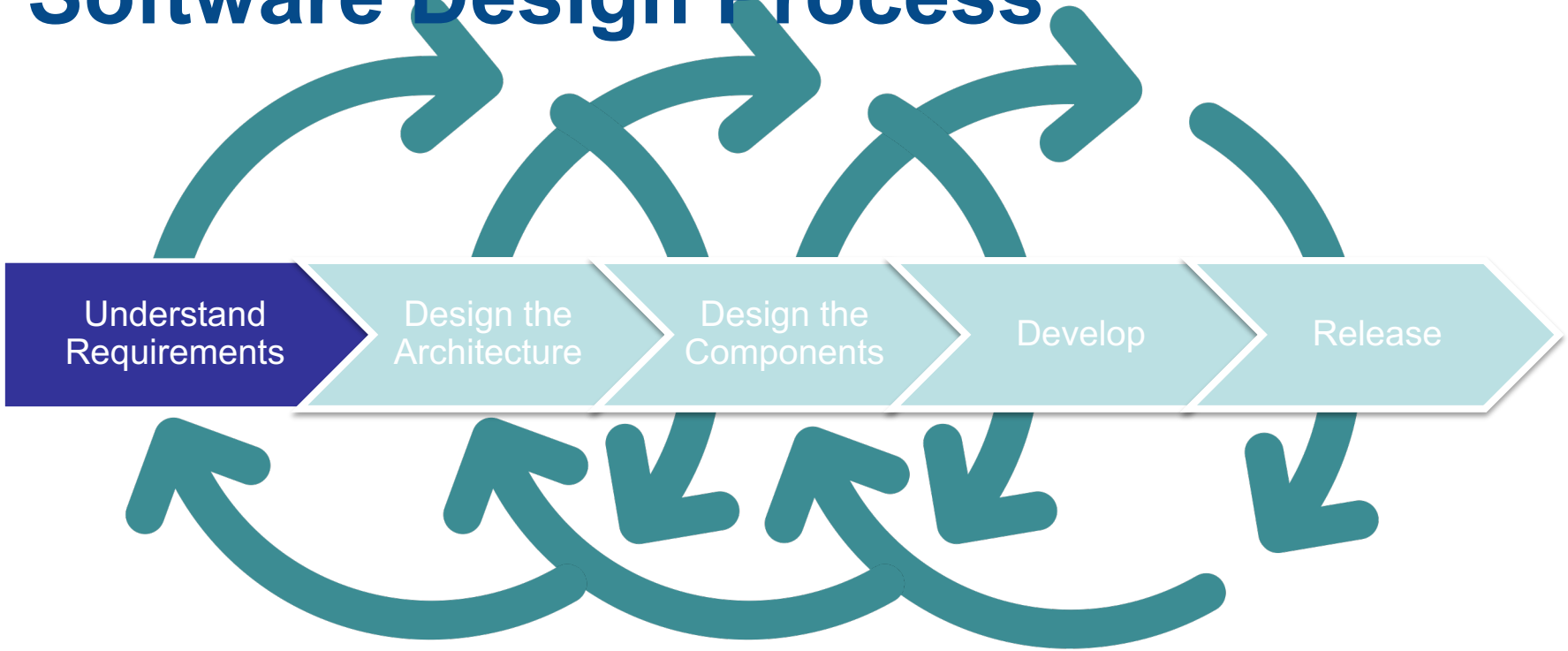
INFS 2044

Workshop 1a Answers

Preparation Already Done

- Revise SMART and FURPS+ topics from INFS1026 System Requirements and User Experience
- Bring a copy of the workshop instructions (this document) to the workshop

Software Design Process



Learning Objectives

- Identify deficient requirements and improve them
- Assess the completeness of a domain model
- Assess the completeness of a use case list

Task 1. Requirements Review

- Requirements form the basis for design
- How do we know we have good requirements?
- What are the properties of good requirements?
- What can we do to check?

S.M.A.R.T.

- Specific
- Measurable
- Achievable
- Realisable
- Traceable

Requirements Example

- R01: The system shall process payments.

- R01: The system shall authorize payments via credit card.

Requirements Example

- R02: The system shall be fast.

- R02: The system shall authorize credit card payment in no more than 30 seconds under normal conditions in 90% of cases.

Requirements Example

- R03: The system shall be compatible with every accounting system.

- R03: The system shall log sales transactions in the SAP ERP system.

Requirements Example

- R04: The system shall respond to each request in no more than 1 ps.

- R04: The system shall respond to each request in no more than 0.2 seconds.

Requirements Example

- R05: The cashiers should be friendly.

- R05: The system shall comply with ergonomic guidelines XYZ.

Requirements Example

- The system shall be no heavier than 0.2kg.

- R06: The system shall be no heavier than 0.2kg as per store fit-out policy XYZ.

Requirements Example

- R07: The system shall cost no more than AUD 2,000.

- Not a SYSTEM requirement. This is a business requirement or project requirement.

Requirements Example

- R08: The system shall have a *Payment* module.

- This is a solution masquerading as a requirement.

Requirements Completeness

- How do we know that we have captured all requirements?
- We cannot be sure
- Experience with similar systems can help
- Thinking about the different aspects of system requirements can help identify what may have been missed

FURPS+

- Function
- Usability
- Reliability
- Performance
- Security
- +: Design constraints, implementation-, interface-, physical-, and supportability requirements

Examples of requirements

1. … calculate taxes based on country-specific rules.
2. … display readable by normal-sighted persons at 1 meter distance
3. … respond to user input in no more than 0.2 seconds.
4. … continue processing sales if network unavailable.
5. … be unavailable for no more that 1 hour per week.
6. … under normal conditions process each message in no more than 1 second.

Examples of requirements

1. **F** calculate taxes based on country-specific rules.
2. **U** display readable by normal-sighted persons at 1 meter distance
3. **U** respond to user input in no more than 0.2 seconds.
4. **R** continue processing sales if network unavailable.
5. **R** be unavailable for no more that 1 hour per week.
6. **P** under normal conditions process each message in no more than 1 second.

Examples of requirements

7. … process no less than 100 requests per second.
8. … securely transmit data in flight.
9. … limit access to services to authenticated and authorized users.
10. … expire user sessions after 5 minutes of inactivity.
11. … includes only company-approved software.
12. … implemented in Python.
13. … enable configuration of business rules.

Examples of requirements

7. **P** process no less than 100 requests per second.
8. **S** securely transmit data in flight.
9. **S** limit access to services to authenticated and authorized users.
10. **S** expire user sessions after 5 minutes of inactivity.
11. **+** includes only company-approved software.
12. **+** implemented in Python.
13. **+** enable configuration of business rules.

Review a Use Case: NFRs

- Review the use case *Make Booking.*
- Identify non-functional requirements pertaining to this use case.

Use Case UC01: Make Booking

1. *User enters date range.*
2. *System presents available rooms, their descriptions, and daily rates.*
3. *User selects room.*
4. *System presents total price.*
5. *User enters contact details and payment details.*
6. *System verifies payment, records payment confirmation, and issues booking confirmation.*

NFR Considerations (Examples)

- Usability
  - Compliance with UX guidelines, device and platform standards
  - Response times
- Reliability
  - Availability of the system
- Performance
  - Transaction throughput, concurrent user sessions

NFR Considerations (Examples)

- Security
  - Securely process payments
  - Encrypt data in flight
  - Do not store payment details
  - Limit access to stored data to authenticated and authorized users
- Other
  - User interface multiple languages
  - Payment in different currencies
  - Process payment via payment provider XYZ

Task 2. Review Domain Model

- How do we know if the domain model is correct?
- Replay the use case narratives on the domain model to see if the model includes all concepts, attributes, and relationships required by the use case.
- Any missed elements indicate that the domain model may be incomplete.

Review Model for Room Booking

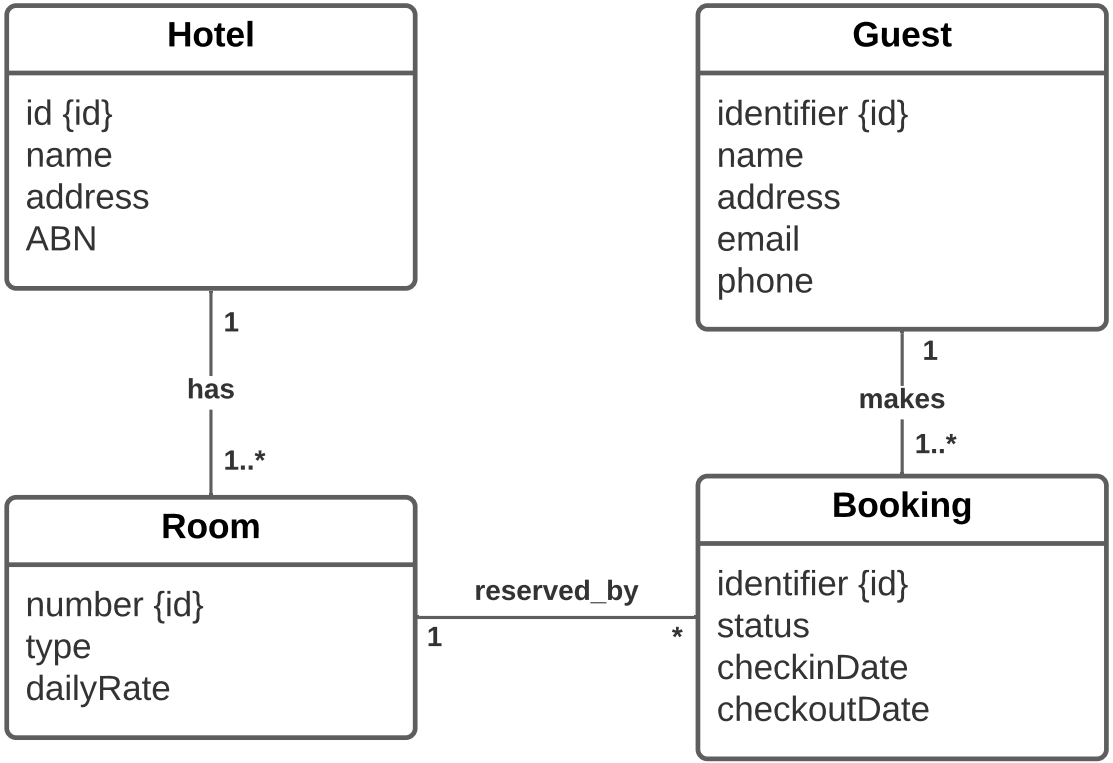
- Simulate the use case narrative for the *Make Booking* on the Domain Model.

- Does the domain model capture all relevant concepts, attributes, and relationships that are needed for simulating the use case narrative?
- If not, what has been missed?

Use Case UC01: Make Booking

1. *User enters date range.*
2. *System presents available rooms, their descriptions, and daily rates.*
3. *User selects room.*
4. *System presents total price.*
5. *User enters contact details and payment details.*
6. *System verifies payment, records payment confirmation, and issues booking confirmation.*

Room Booking Domain Model



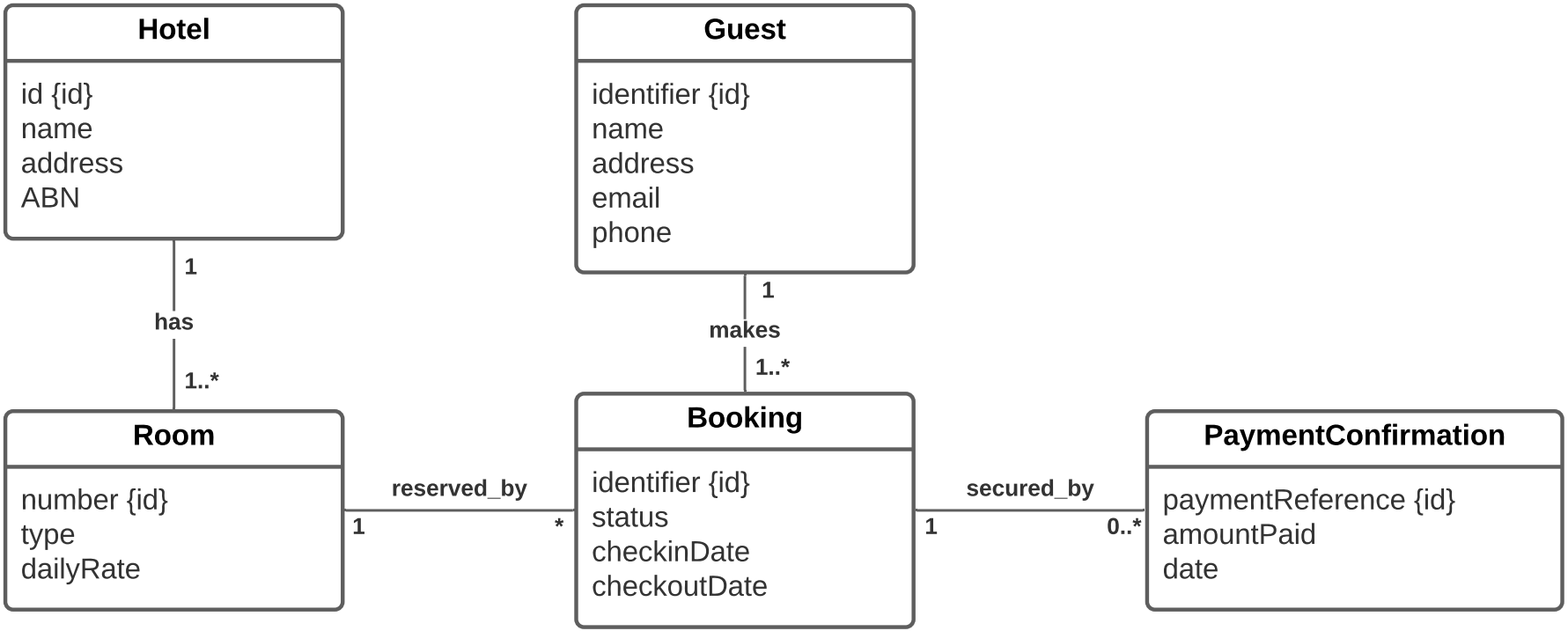
Domain Model Deficiencies

- **Payment Confirmation** requires to be stored but is not included in the domain model
- **Booking Confirmation** is not included
  - This is an omission if the system shall store them (e.g. in a log for debugging / auditing purposes)
  - If Booking Confirmations are transient objects, then not showing them in the domain model is okay.

Amend the Domain Model

- Add the missed elements to the domain model.

Revised Domain Model



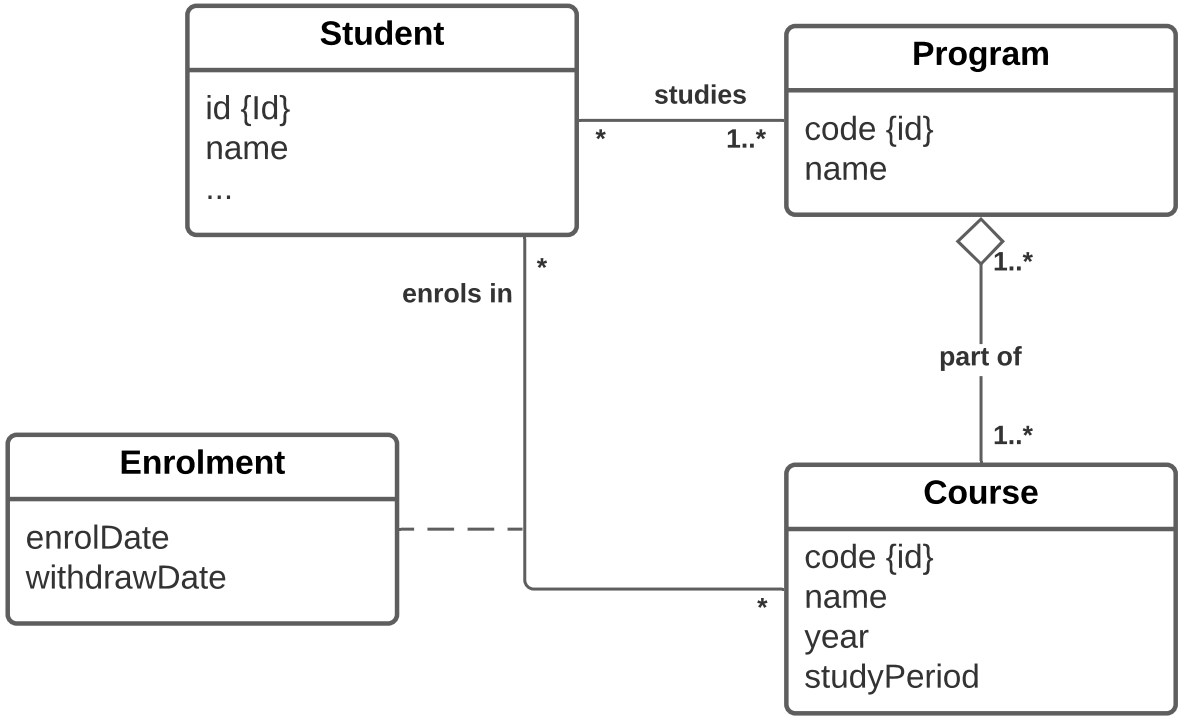
Task 3: CRUD Analysis

- A technique for identifying missing use cases
- Compare expected operations on domain model with actual operations carried out in use cases
- Identified omissions may indicate missed use cases

Enrolment System: Use Cases

- Suppose two use cases were identified
  - Search Courses
  - Enrol in Course

Enrolment System: Domain Model



Enrolment System: CRUD Table

	Student	Program	Course	Enrolment
Search Courses		R	R	
Enrol in Course	R		R	C

Enrolment System: CRUD Analysis

	Student	Program	Course	Enrolment
Search Courses		R	R	
Enrol in Course	R		R	C

- Enrolments can only be created
- Should the system be able to
  - Read enrolments (Use case “List enrolled courses”)
  - Update enrolments (Use case “Change enrolment”)
  - Delete enrolments (Use case “Withdraw from Course”)
  - Add/modify/delete Students, Programs, Courses, etc??

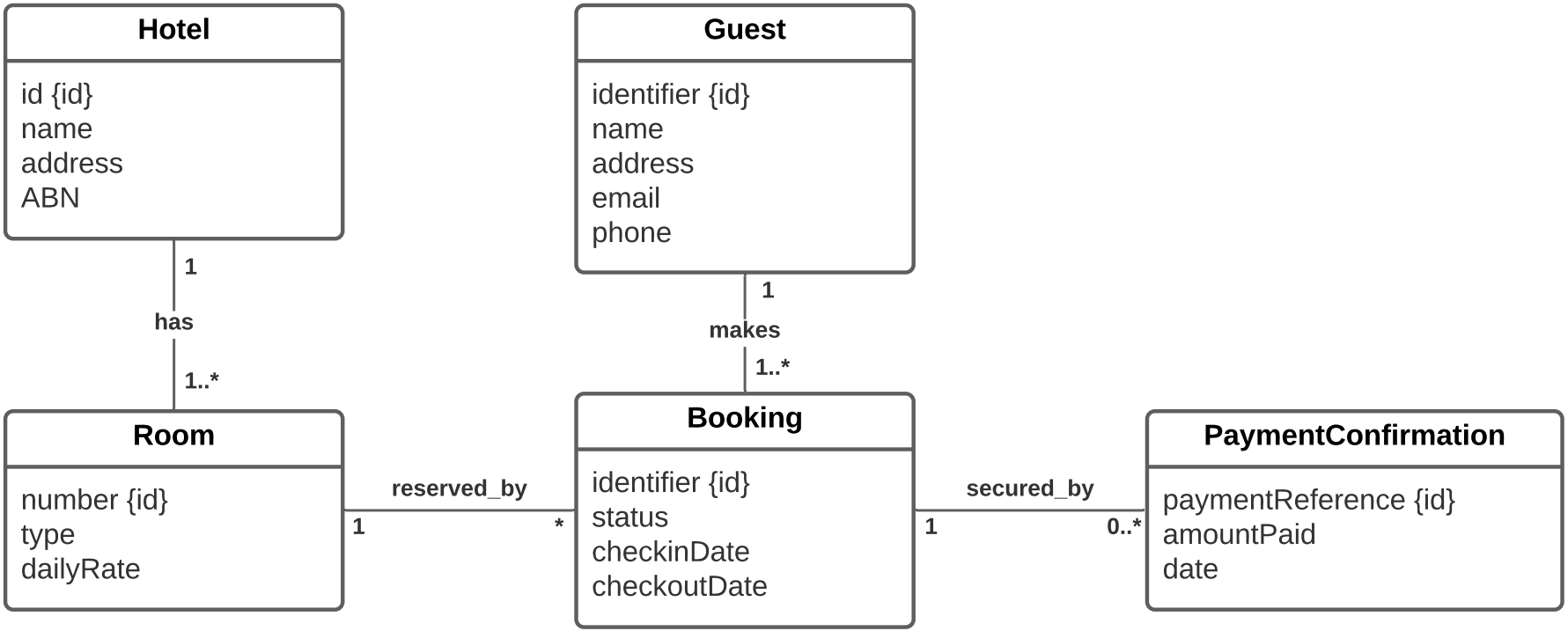
CRUD Analysis for Booking System

- Tabulate the CRUD operations that each use case carries out on the relevant elements in the domain model.
- Make reasonable assumptions where the use case narrative is not given.

Booking System: Use Cases

Identifier	Use Case Name
UC01	Make Booking
UC02	Edit Booking
UC03	View Bookings
UC04	Find Hotel
UC05	Find Rooms

Booking System: Domain Model



CRUD Table for Booking System

	Hotel	Room	Booking	Guest	Payment-Confirmation
UC01 Make Booking		R	C	CR	C
UC02 Edit Booking		R	RU		C
UC03 View Bookings	R	R	R	R	R
UC04 Find Hotel	R				
UC05 Find Rooms	R	R	R		

CRUD Analysis for Booking System

	Hotel	Room	Booking	Guest	Payment-Confirmation
UC01 Make Booking		R	C	CR	C
UC02 Edit Booking		R	RU		C
UC03 View Bookings	R	R	R	R	R
UC04 Find Hotel	R				
UC05 Find Rooms	R	R	R		

Cancel Booking

CRUD Analysis for Booking System

	Hotel	Room	Booking	Guest	Payment-Confirmation
UC01 Make Booking		R	C	CR	C
UC02 Edit Booking		R	RU		C
UC03 View Bookings	R	R	R	R	R
UC04 Find Hotel	R				
UC05 Find Rooms	R	R	R		

(Add / Edit / Remove) Hotel, Room?

CRUD Analysis for Booking System

	Hotel	Room	Booking	Guest	Payment-Confirmation
UC01 Make Booking		R	C	CR	C
UC02 Edit Booking		R	RU		C
UC03 View Bookings	R	R	R	R	R
UC04 Find Hotel	R				
UC05 Find Rooms	R	R	R		

(Edit / Delete) Guest?

CRUD Analysis for Booking System

	Hotel	Room	Booking	Guest	Payment-Confirmation
UC01 Make Booking		R	C	CR	C
UC02 Edit Booking		R	RU		C
UC03 View Bookings	R	R	R	R	R
UC04 Find Hotel	R				
UC05 Find Rooms	R	R	R		

Edit / Delete use cases NOT applicable!

Booking System: Revised Use Cases

Identifier	Use Case Name
UC01	Make Booking
UC02	Edit Booking
UC03	View Bookings
UC04	Find Hotel
UC05	Find Rooms
UC06	Cancel Booking
UC07	Edit Guest

Identifier	Use Case Name
UC08	Delete Guest
UC09	Add Room
UC10	Edit Room
UC11	Remove Room
UC12	Add Hotel
UC13	Edit Hotel
UC14	Remove Hotel

Revised CRUD Table: Booking System

	Hotel	Room	Booking	Guest	Payment-Confirmation
UC01 Make Booking		R	C	CR	C
UC02 Edit Booking		R	RU		C
UC03 View Bookings	R	R	R	R	R
UC04 Find Hotel	R				
UC05 Find Rooms	R	R	R		
UC06 Cancel Booking			U		C
UC07 Edit Guest				U	

Revised CRUD Table: Booking System

	Hotel	Room	Booking	Guest	Payment-Confirmation
UC08 Delete Guest				D	
UC09 Add Room	R	C			
UC10 Edit Room		U			
UC11 Remove Room		D			
UC12 Add Hotel	C				
UC13 Edit Hotel	U				
UC14 Remove Hotel	D				

You Should Know

- Identify poor requirements using SMART
- Identify missed requirements using FURPS+
- Validate domain models by replaying use cases
- Validate use cases using CRUD

Activities this Week

- Attend second Workshop session
- Complete Quiz 1

University of
South Australia