# 2DV603
## Software Engineering
## Assignment 1 — Requirements Engineering

Jonas Sjöberg
860224-xxxx
Linnaeus University
js224eh@student.lnu.se
https://github.com/jonasjberg
http://www.jonasjberg.com

| | |
|---|---|
| Written during: | February 7, 2018 – February 11, 2018 |
| Teacher responsible: | Mauro Caporuscio |
| | Diego Perez |

**Abstract**

The first assignment in the course *2DV603 — Software Engineering* at Linnaeus University during the spring term of 2018. The goal of this assignment is to create a system to manage front-desk activities of a fictional hotel, following a given scenario.

# Contents

**1 Domain Analysis Document** **2**
1.1 Existing Front-desk Systems . . . 2
1.2 Hotel Reservations Domain Analysis . . . 3
1.2.1 Initial Observations . . . 3
1.2.2 Conclusions . . . 5

**2 Requirements Document** **5**
2.1 Scenarios . . . 5
2.1.1 Potential guest inquires about future patronage . . . 5
2.1.2 Guest checks in at the hotel . . . 6
2.1.3 Guest cancels a reservation . . . 6
2.2 Use Cases . . . 6
2.2.1 Use Case — Find Vacant Rooms . . . 6
2.2.2 Use Case — Find Vacant non-smoking Rooms . . . 7
2.2.3 Use Case — Reserve a Specific Room . . . 8
2.3 Use Case Analysis . . . 9
2.4 Refined Problem Statement . . . 10
2.5 Features in the First Release . . . 10

**3 Time Log** **10**

**References** **10**

# 1 Domain Analysis Document

**Our Goal** is to create a system to manage front-desk activities at the "Linnaeus Hotel".

## 1.1 Existing Front-desk Systems

A very brief look into prior work led to this [1] example of a hotel management system that seemingly includes most features that are arguably required in order to meet the requirements in this assignment.

A partial screenshot is shown in Figure 1.

Features are grouped by into these main groups:

- Booking Engine
- Rooms and Rats
- Booking Channels (OTAs)
- Guest Communication
- Booking Management
- Service and Support
- Payment Processing and Invoicing

It is obvious that most of these features stem from the basic features we likely need to implement in our management system. Their system has probably grown from a set of basic requirements and use cases, developing and adding new features over time after customer feedback and accumulation of domain-specific knowledge and experience.

I find that the functionality for exporting various data to PDF documents, display in a web front end, e-mailing to customer, etc. are typical examples of components that can be easily added when the underlying software architecture is sound.

Another key point is that the domain-specific functionality will not likely change all too much, and the malleable parts should be made user-adjustable to fit individual user requirements.

Initial development likely deals mostly with core functionality like domain logic and transforming pure data. And as development goes on, the core will likely not see too many invasive changes (assuming a "good", flexible design) and the development effort moves to different means of user interaction, such as data exporting.

## 1.2 Hotel Reservations Domain Analysis

### 1.2.1 Initial Observations

Initial assumptions about the domain are based on the description and information given in the assignment instructions.

- Apparent main uses of the system:
  - Add, retrieve, modify room reservations (Scenario 2.1.1)
  - Add, retrieve, modify guest information (Scenario 2.1.1)
  - Provide information in "real-time" when staff interact with guests (Scenario 2.1.2)
  - Respond to queries for vacant rooms (Scenario 2.1.1)
- Notable characteristics of Hotel Rooms
  - Uniquely identified by the room number
  - Always either occupied or vacant at a specific point in time
  - Mutually exclusive smoking or non-smoking
  - Some adjoin other with internal doors between them
  - "Attractiveness" (room with a view, room size, etc.) is summarized with a quality level
    - Priced based on quality level (attractiveness)
    - Quality levels determine max daily rate
  - Contains a fixed number of beds with specific sizes
- Notable characteristics of Reservations:
  - Always involve rooms
  - Spans a period of time
- Notes on the Data Storage:

Compare Our Plans

| Booking Engine | Starter | Pro |
|---|---|---|
| Commission & fees | 0% | 0% |
| Booking engine on your website | ✓ | ✓ |
| 20+ languages supported | ✓ | ✓ |
| WordPress plugin | ✓ | ✓ |
| Booking engine on your Facebook page | ✓ | ✓ |
| Customized input fields | ✓ | ✓ |
| Alerts on new bookings | ✓ | ✓ |
| Review-your-booking feature | ✓ | ✓ |
| Multiple booking engines | | ✓ |

| Rooms & Rates | Starter | Pro |
|---|---|---|
| Create multiple room types | ✓ | ✓ |
| Offering add-ons & extras | ✓ | ✓ |
| Categorizing room types | | ✓ |
| Block out individual rooms/units | ✓ | ✓ |
| Rate manager | ✓ | ✓ |
| Add surcharges & taxes | ✓ | ✓ |
| Set restrictions (closed, length of stay, etc.) | ✓ | ✓ |
| Seasonal rates | ✓ | ✓ |
| Discount/coupon codes | ✓ | ✓ |

| Booking Channels (OTAs) | Starter | Pro |
|---|---|---|
| Booking.com, Expedia, Agoda, Hostelworld, HotelSpecials | | ✓ |
| TripAdvisor Instant Booking, Review Express | ✓ | ✓ |
| Google Hotel Ads | | ✓ |
| iCal import/export (Airbnb, 9flats, HomeAway, etc.) | | ✓ |
| SiteMinder integration | | ✓ |
| MyAllocator integration | | ✓ |

| Guest Communication | Starter | Pro |
|---|---|---|
| Customized email and SMS templates | ✓ | ✓ |
| Automated messages based on check-in, check-out, etc. | ✓ | ✓ |
| Sending text messages (per SMS) | $0.15 / £0.10 / €0.10 | $0.08 / £0.05 / €0.05 |

| Booking Management | Starter | Pro |
|---|---|---|
| Calendar overview | ✓ | ✓ |
| Modify & cancel bookings | ✓ | ✓ |
| Group & split bookings | ✓ | ✓ |
| Reuse contact details for returning guests | ✓ | ✓ |
| Schedule cleaning & housekeeping tasks | ✓ | ✓ |
| Printable booking PDFs | ✓ | ✓ |
| History log | ✓ | ✓ |
| Statistics | ✓ | ✓ |
| Export booking information | ✓ | ✓ |
| Callback URL | ✓ | ✓ |

| Service and Support | Starter | Pro |
|---|---|---|
| Anytime, anywhere access | ✓ | ✓ |
| Free 24/5 support by email, live chat & phone | ✓ | ✓ |
| Encrypted communication & automated backups | ✓ | ✓ |
| Multiple user accounts | | ✓ |
| Restricted user access | | ✓ |

| Payment Processing & Invoicing | Starter | Pro |
|---|---|---|
| Take payment at the time of booking | ✓ | ✓ |
| Store credit card details for later processing | ✓ | ✓ |
| Create invoices and cash receipts | ✓ | ✓ |



Figure 1: Cropped screenshot of features provided by a hotel reservation management system[1]

- `CRUD` database
  - Store basic information about guests
    - Name
    - Address
    - Telephone number
    - Credit card (security requirements)
    - Passport number (security requirements)
  - Store payment data
- Provides interface to front end systems
  - Query for relevant data for billing guests
- Must not be insanely slow — probably not an issue in practice

### 1.2.2 Conclusions

Also, we might assume that the very central core of the system will not be very complicated. It is purely data storage, retrieval and update ("CRUD").

Programming is transforming data. This domain will likely not require all too specific means of transforming data. Even so, compared to say an online library that might require knownledge of standards for identification such as ISBN-numbers, DOIs, etc. Most such systems I've looked at have utilized some sort of automated metadata extraction system, text search, metadata normalization, comparison, etc. All very difficult problems that require very advanced techniques.

I would argue that this domain might not be "computationally" difficult to deal with. High risk tasks such as secure payment systems, conformant storage of user information, etc., can is best solved by integrating third-party software.

My theory is that the most difficult part of working in this domain has to do with human-interaction and usability — hotel personnel will spend countless hours with the software and will definitely have a lot of ideas on how it could be improved to streamline their workflow.

However, we must be provided with this information and *understand it very well*, as to facilitate us to translate a personal, subjective account of practical user interaction to requirements/features and finally code.

# 2 Requirements Document

## 2.1 Scenarios

These three scenarios are derived from the assignment instructions. [2]

### 2.1.1 Potential guest inquires about future patronage

- A travelling salesman needs somewhere to stay for one night a couple of weeks from now. This person contacts the hotel by phone to inquire about vacancy over a couple of days.
- The hotel personnel answers any questions and uses the management system to find out if any rooms are available in the requested time frame. The hotel employee relays information about which rooms are available.
- After hearing about the rooms, the salesman asks about room sizes.

- The employee looks up relevant characteristics and attributes and relays the information.
- After brief contemplation, the salesman decides to make a reservation for one of the vacant rooms.

#### 2.1.2 Guest checks in at the hotel

- A travelling salesman arrives at the hotel after having sold nothing and has no patience for delays and pleasantries. He walks up to the hotel front desk and declares that he has made a reservation in advance and would like to check in.
- The receptionist scrambles to find which room number the guest ostensibly has reserved. The receptionist asks the salesman for personal information used when making the reservation.
- The salesman provides personal information to the receptionist.
- The receptionist enters the information into the management system. After narrowing down the search results to a single number, the receptionist can retrieve the room key and proceed to guide the guest to his room.
- The salesman is provided with his room key and instructions on where the room is located.

#### 2.1.3 Guest cancels a reservation

- A cat named Gibson wishes to cancel a reservation because of feline matters. Gibson logs in to the hotel website using a identifier and access key that was provided to him when the room was first reserved.
- The web portal loads in under 2 seconds, which is probably considered good enough for a modern website. Gibson cries in cat but acknowledges that everything is really slow now and the hotel web portal is probably comparably "performant".
- After clicking on clearly visible "cancel reservation" link, a message informs Gibson about the cancellation fee, calculated from the room price and time delta from planned residence and cancellation.
- Gibson sighs in cat and clicks the "I Agree" link, which leads to a confirmation page with a summary of upayments that will be billed.

### 2.2 Use Cases

#### 2.2.1 Use Case — Find Vacant Rooms

The UML use case diagram for Usecase 2.2.1 is shown in Figure 2.

**Actors**

- Employee — uses the system to gather information
- Customer — inquires about vacant rooms at a specific date

**Goals**

- Find rooms that are vacant at a certain date
- Respond to the customers inquiry

**Preconditions**

- The system is “live” and reflects the actual state of the world
- The employee is authorized to access information on room availability

**Summary** Demonstrates real-time use of the system to answer questions during interaction with a customer.

**Related** This use case is extracted from Scenario 2.1.1.

**Entry Conditions** The employee wants to get information to provide to a customer during an interaction.

**Steps**

1. Navigate to the room-search functionality
2. Enter the date of interest
3. Proceed to the results listing
4. Repeat steps 2–3 until all customer inquiries are resolved

**Exit Condition** The employee can answer the customers requests through information provided by the system.

### 2.2.2 Use Case — Find Vacant non-smoking Rooms

The UML use case diagram for Usecase 2.2.2 is shown in Figure 2.

**Actors**

- Employee — uses the system to gather information

**Goals**

- Find rooms for non-smokers that are vacant at a certain date

**Preconditions**

- The system is “live” and reflects the actual state of the world
- At least one room is marked as a “non-smoking” room
- The employee is authorized to access information on room availability

**Summary** Demonstrates room queries with a conditional.

**Related** This use case is extracted from Scenario 2.1.1. It is also related to Usecase 2.2.1.

**Entry Conditions** The employee wants to find rooms for non-smokers that are vacant at a certain date.

**Steps**

1. Navigate to the room-search functionality
2. Enter the date of interest
3. Enable filtering non-smoking rooms
4. Proceed to the results listing

**Exit Condition** The employee has gathered information on whether there are vacant non-smoking rooms at a given date.

### 2.2.3 Use Case — Reserve a Specific Room

The UML use case diagram for Usecase 2.2.3 is shown in Figure 3.

**Actors**

- Manager — uses the system to reserve a room
- Customer — wants to reserve a specific room

**Goals**

- Make a reservation of a specific room

**Preconditions**

- The system is "live" and reflects the actual state of the world
- The manager can make reservations of specific rooms

**Summary** Involves a special kind of interaction *that might require elevated privileges.*

**Entry Conditions** The manager wants to meet a less frequent, specific request.

**Steps**

1. Navigate to the room-search functionality
2. Enter the room number of the requested room
3. See if it is currently vacant
4. If not:
    - Reserve the room
5. If it is:
    - Attempt to exploit elevated privileges to invalidate the current vacancy
    - Reserve the room

**Exit Condition**

1. The requested room is either already reserved and the manager can not meet the customer request.
2. The requested room is reserved.

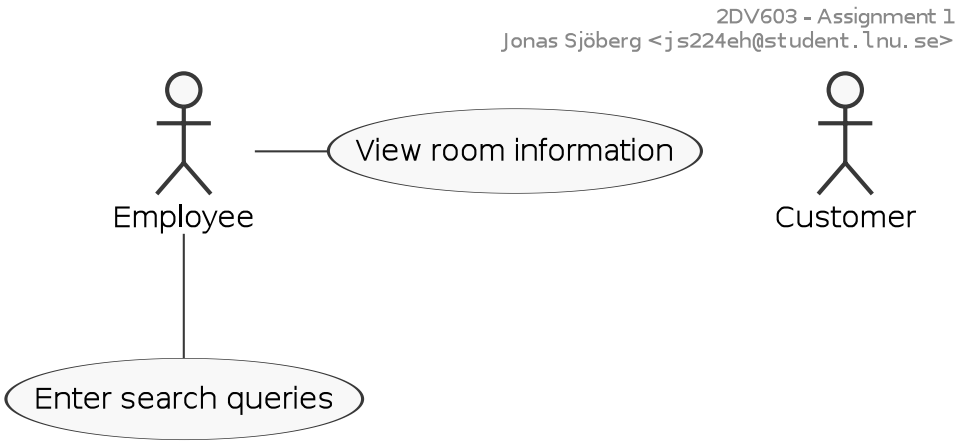


Figure 2: UML Use Case Diagram for Use Case 1 – Get information on room

The UML use case diagram for Usecase 2.2.1 and Usecase 2.2.2 is shown in Figure 2. The UML use case diagram for Usecase 2.2.3 is shown in Figure 3.

## 2.3 Use Case Analysis

The use cases indicate that all tasks either retrieve data on rooms/reservations, modify data on rooms/revervations or both.

There is also a need for different “priveleges” as indicated by the somewhat unclear description in the instructions that seems to indicate that the manager should have extra control over the system.

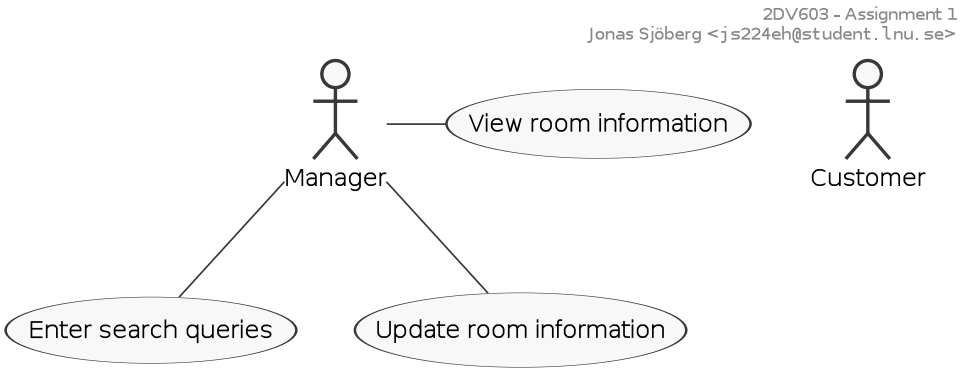


Figure 3: UML Use Case Diagram for Use Case 2 – Get information on room and make a reservation

| Date and time | Activity | Time Spent |
|---|---|---|
| 2018-02-07 | Initial skeleton document outline and notes | 2h |
| 2018-02-09 | Domain analysis | 1h |
| 2018-02-09 | Requirements Document | 1h |
| 2018-02-10 | Domain analysis, Use Cases, Requirements | 3h |
| 2018-02-11 | Everything else | 4h |

Table 1: Time Log for Assignment 1

### 2.4 Refined Problem Statement

The core problem is designing a "suitable" storage system for the relevant data. This system should provide some means of retrieving data, as well as retrieving data that matches some given conditional statements.

Then we want to wrap this storage backend with user interfaces that give all actors a controlled means of interacting with the data. This might be a web portal or a desktop application.

So, we need a SQL database at the very bottom. Then a thin layer of domain logic that abstracts the persistence layer. A classic Model View Controller architecture could be used to allow different Views as well as clearly defined boundaries between sub-systems.

### 2.5 Features in the First Release

The first release should aim to provide a robust storage backend and some prototypal user interface. Very early versions probably focus on the functional core and might use a mock database and view, so that these can easily be replaced later.

First release should focus on basic "CRUD" functionality with restrictions and domain rules designed from the requirements.

I would not think it productive to focus on user interfaces and usability before a central core has been implemented.

## 3 Time Log

An (approximate) time log for this assignment is shown in Table 1.

## References

[1] *Sirvoy hotel reservation system features*, [Online; accessed 10-Feb-2018], 2018. [Online]. Available: https://sirvoy.com/features-pricing/.

[2] M. Caporuscio, *Assignment 1*, [Online; accessed 7-Feb-2018], 2018. [Online]. Available: https://mymoodle.lnu.se/pluginfile.php/2998633/mod_assign/introattachment/0/2DV603%20-%20Requirements%20Engineering%20Assignment.pdf?forcedownload=1.